

March 2025

MARKTECHPOST



Scan here to Partner

OPEN-SOURCE AI



Journey & Insights
YAM MARCOVITZ
CEO Emcie

From Vision to Generative AI
Talk with OpenVino team

Exclusive Talk
ANITA LACEA
Microsoft

Interview with
HAMZA TAHIR
CTO ZenML

unsloth AI
How a Two-Person Team Is Revolutionizing LLM Training with GPU Kernel Magic

Exclusive Interview
ANIMESH SINGH
LinkedIn



MARKTECHPOST

Marktechpost AI Media Inc is an AI Research and Dev News Platform

- * Monthly Traffic on Marktechpost.com: 2 Million+
- * AI Community: 500k+ Members
- * Newsletter

EDITORIAL TEAM



Asif Razzaq

Editor at Marktechpost

As the CEO and Editor of Marktechpost Media Inc., a California-based AI digital publishing platform, Asif has dedicated his career to making AI and ML accessible to a broad audience through compelling and insightful media content. Under his stewardship, Marktechpost has evolved into a premier destination for AI enthusiasts, professionals, and researchers, boasting approximately 1.5 million monthly visitors.



Jean-Marc Mommessin

Editor at Marktechpost

Jean-Marc is a seasoned AI executive with a track record of driving growth for AI-powered solutions. He founded a computer vision company in 2006 and has since become a recognized leader in the field, frequently speaking at AI conferences. He holds an MBA from Stanford and a Master's in Engineering from Arts et Métiers. Jean-Marc has filed patents on agentic AI and has played key leadership roles, including leading sales at Exodigo.ai and other high-growth startups. He is currently based in Palo Alto, CA.



Tarry Singh

Partner at Marktechpost

Tarry Singh is the Board Director & CEO of DK AI Holding, as well as a co-founder and AI researcher for Real AI, an enterprise AI startup, and Earthscan.io, an energy-focused AI startup. These ventures are part of NVIDIA's Inception Program, highlighting top AI startups worldwide. With 30 years of expertise, Tarry has advised global executives, government leaders, and country states on building data-driven organizations from the ground up. A frequent speaker at global AI leadership summits, he leads high-impact workshops with a team specializing in Generative AI, NLP, Computer Vision, Robotics, and other AI fields.



Tallin Muskat

Founding Member at Marktechpost

Tallin Muskat is a visionary entrepreneur, recruiter, and Founding Member of MarkTechPost, known for her expertise in digital innovation and content strategy. With a strong background in technology and marketing, she identifies emerging talent while driving the growth of a dynamic tech community.

Table of Contents

Why Open Source AI? _____	05
Open Source AI Landscape _____	06
Why Open-sourcing AI for Production? _____	14
Direct Dialogue <i>Robert Nishihara (CEO Anyscale)</i> _____	15
Conversation with <i>Yam Marcovitz (CEO Emcie)</i> _____	18
Unsloth AI _____	24
Interview with <i>Hamza Tahir (CTO ZenML)</i> _____	30
Exclusive Talk <i>Anita Lacea (Microsoft)</i> _____	36
Achieving critical reliability in Instruction- Following with LLMs (<i>Parlant</i>) _____	42
From Vision to Generative AI (<i>Intel</i>) _____	46
Exclusive Interview <i>Animesh Singh (Linkedin)</i> _____	50
From GenAI Demos to Production _____	54
Deep Dive: Exploring _____ <i>Byron Hsu (xAI)</i>	62
Inside Vectara's AI Journey _____	65
AI PCS- A Big Moment for Open Source AI (<i>Darren Oberst</i>) _____	68
Goodbye Open AI (<i>Devvret Rishi</i>) _____	71
Verdict: A Library for Scaling Judge Time-Compute (<i>Nimit Kalra & Leonard Tang</i>) _____	76
Exclusive Interview <i>Boris Yangel (Head of AI R&D Nebius)</i> _____	80
Platforms for Open Source AI _____	82

Why OPEN SOURCE AI?



According to a McKinsey report this February, business leaders are almost evenly split in their preference between open-source and proprietary AI. Additionally, Deepseek has demonstrated that open-source AI matches the performance of its proprietary counterparts, much like a tsunami sweeping through the industry.

In the rapidly evolving world of artificial intelligence (AI), open source solutions have emerged as a powerful tool for businesses seeking to implement AI in production. These solutions offer not only a cost-effective alternative to proprietary software but also bring a wealth of resources and community-driven support. The growing adoption of open source AI in production environments is driven by several factors:

Cost: One of the primary attractions of open source AI is its **cost efficiency**. Organizations can save significantly on licensing fees, which can be prohibitive with proprietary AI solutions. Additionally, open source AI offers unparalleled **flexibility and customizability**. Companies can tailor the software to meet their specific needs without the restrictions typically imposed by closed systems.

The **community support** surrounding open source projects is another significant benefit. A vast network of developers contributes to continual improvements and innovations, ensuring that the software evolves quickly to meet the demands of modern AI applications. Furthermore, the **transparency** of open source AI builds trust, as users can verify the integrity and fairness of the algorithms themselves.

Challenges of Implementing Open Source AI in Production Despite its advantages, open source AI

presents certain challenges. Technical support can be sporadic, and organizations might need to rely on community forums or hire specialists for maintenance and troubleshooting. Integration with existing systems can also pose challenges, requiring skilled professionals to ensure smooth operation.

Moreover, the **quality and reliability** of open source projects can vary greatly. Organizations must carefully select projects with active maintenance and a robust user base to ensure they are adopting a reliable solution.

Real-World Applications : Many industries have successfully integrated open source AI into their production processes. For example, in healthcare, open source algorithms are being used to predict patient outcomes and optimize treatment plans. In the automotive industry, open source AI assists in everything from automated driving functions to predictive maintenance.

Looking forward, the influence of open source AI is only expected to grow. It plays a crucial role in democratizing AI technology, making powerful tools accessible to a broader range of businesses and developers. As the community continues to expand and technology advances, open source AI will remain at the forefront of innovation in production environments.

Open source AI offers a promising path for organizations looking to leverage AI technology in production. By understanding its benefits and navigating its challenges, businesses can effectively harness the power of community-driven AI to drive innovation and maintain competitive advantage in their industries.

OPEN SOURCE AI LANDSCAPE

Models

AI observability & Monitoring

LLM Hosting & deployment

Synthetic Training Data

AI Agent frameworks

AI observability & Monitoring

Vector Database

Security, Privacy & Governance



OPENSOURCE AI MODELS

Microsoft



Microsoft released 2 open source models : Phi-4-multimodal and Phi-4-mini, the newest models in Microsoft's Phi family of small language models (SLMs). These models are designed to empower developers with advanced AI capabilities. Phi-4-multimodal, with its ability to process speech, vision, and text simultaneously, opens new possibilities for creating innovative and context-aware applications. Phi-4-mini, on the other hand, excels in text-based tasks, providing high accuracy and scalability in a compact form.

Qwen



Qwen is a series of large language models (LLMs) independently developed by Alibaba Cloud.

Meta



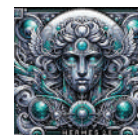
Meta is the front runner of open source LLMs and develops the Llama family of models.

Mistral



Mistral offers several open source model from Mistral Small, to Pixal (multimodal) to research models such as Codestral for coding

Open Hermes



OpenHermes Mistral is a state of the art Mistral Fine-tune, provided by OpenHermes which trained on additional code datasets. OpenHermes was trained on 1,000,000 entries of primarily GPT-4 generated data, as well as other high quality data from open datasets across the AI landscape.

NVIDIA



NVLM is a family of multimodal LLMs developed by NVIDIA

Nous



Nous Research developed Hermes 3 contains advanced long-term context retention and multi-turn conversation capability, complex roleplaying and internal monologue abilities, and enhanced agentic function-calling.

DeepSeek



DeepSeek-R1 is an open-source language model designed to perform a variety of text-based tasks in both English and Chinese. With 671 billion parameters and a Mixture of Experts (MoE) architecture, it selectively activates only a portion of its parameters during operation, optimizing performance and reducing computational costs.

Janus-Pro-7B, developed by DeepSeek, is an advanced multimodal AI model designed for both image generation and analysis

Gemma 2.0



Gemma 2.0 Flash is a part of Google's Gemini AI suite. It is an experimental model designed to enhance real-time interaction and multimodal capabilities. It introduces several notable features aimed at improving user experience and performance.

Imagen 3 is a highest quality text-to-image model, capable of generating images with even better detail, richer lighting and fewer distracting artifacts

VideoWorld



The video generation experimental model "VideoWorld" was jointly proposed by the Doubao Big Model team, Beijing Jiaotong University, and the University of Science and Technology of China. Unlike mainstream multimodal models such as Sora, DALL-E, and Midjourney, VideoWorld is the first in the industry to realize the world cognition without relying on language models.

Arcee



Arcee developed Arcee-Maestro-7B-Preview, and a fast and efficient Mistral-based DeepSeek distillation we call Arcee-Blitz.

Ai2



Allen Institute develops open source language models, including a family of 7B and 13B models trained up to 5T tokens. As an example, OLMo 2 outperforms other fully open models and competes with open-weight models like Llama 3.1 8B

Hugging Face



SmolLM by Huggingface is a series of state-of-the-art small language models available in three sizes: 135M, 360M, and 1.7B parameters. These models are built on a meticulously curated high-quality training corpus to run on device.

Stability.ai



Stable Diffusion open sources Stable Diffusion 3.5 Large and Stable Diffusion 3.5 Large Turbo,... These models are highly customizable for their size and can run on consumer hardware

LLM HOSTING & DEPLOYMENT

Ollama



Ollama is an open-source tool and allows users to run various open-source large language models (LLMs) locally on their own systems.

BentoML



BentoML is a open source Unified Serving Framework for AI/ML Systems. BentoML is a Python library for building online serving systems optimized for AI apps and model inference. It supports serving any model format/runtime and custom Python code, offering the key primitives for serving optimizations, task queues, batching, multi-model chains, distributed orchestration, and multi-GPU serving.

OpenVINO



OpenVINO is a open-source toolkit that helps developers optimize and deploy AI models running on intel Chips. It can be used to build applications for tasks like computer vision, speech recognition, and natural language processing. Developers can leverage models from PyTorch, TensorFlow, TensorFlow Lite, PaddlePaddle, and ONNX directly.

ZenML



ZenML is an extensible, open-source MLOps framework for creating portable, production-ready machine learning pipelines. By decoupling infrastructure from code, ZenML enables developers across your organization to collaborate more effectively as they develop to production.

Nebius



Nebius has open-sourced a Kubernetes operator called "Soperator" to manage Slurm clusters on Kubernete towards running AI loads.

AI OBSERVABILITY & MONITORING

Langfuse



Langfuse is an open-source LLM engineering platform (GitHub) that helps teams collaboratively debug, analyze, and iterate on their LLM applications. All platform features are natively integrated to accelerate the development workflow.

LangSmith



LangSmith is a platform for building, testing, and monitoring large language model (LLM) applications. It's designed to help developers ship applications quickly and reliably.

NannyML



NannyML is an open-source python library that allows you to estimate post-deployment model performance (without access to targets), detect data drift, and intelligently link data drift alerts back to changes in model performance

Truera



TruEra "TruLens," is open source software, that allows developers to evaluate and track the quality of their Large Language Models (LLMs) using a set of defined metrics and feedback functions.

SYNTHETIC TRAINING DATA

Synthea



Synthea is an open-source synthetic patient generator aimed at healthcare research and simulation. It enables the creation of vast and diverse virtual patient populations with intricate medical histories, demographics, and clinical data.

CTGAN



CTGAN is a collection of Deep Learning based synthetic data generators for single table data, which are able to learn from real data and generate synthetic data with high fidelity.

Human Data



Synner/Human Data Interaction Lab is a tool that helps users generate real-looking synthetic data by visually and declaratively specifying the properties of the dataset such as each field's statistical distribution, its domain, and its relationship to other fields

AI AGENT FRAMEWORKS

Human Data



LangChain is a composable framework to build with LLMs. LangGraph is the orchestration framework for controllable agentic workflows. Run.

LangGraph



LangGraph

Gain control with LangGraph to design agents that reliably handle complex tasks. Build and scale agentic applications with LangGraph Platform.

CrewAI



CrewAI is a framework that uses AI to organize multiple agents to work together to complete tasks. It's an open-source, Python-based framework that can be used for a variety of industries and applications.

AutoGen



AutoGen is an open-source programming framework for building AI agents and facilitating cooperation among multiple agents to solve tasks. AutoGen aims to provide an easy-to-use and flexible framework for accelerating development and research on agentic AI.

SuperAGI



SuperAGI is an open-source framework that allows developers to create, manage, and run autonomous AI agents. It combines AI agents with human workflows to automate tasks and improve efficiency.

LLMWare



LLMWare focuses their open source research efforts on the new "ware" ("middleware" and "software" that will wrap and integrate LLMs), as well as building high-quality, automation-focused enterprise models available in Hugging Face.

LlamaIndex



LlamaIndex is an open-source framework that lets users combine custom data sources with large language models (LLMs). It's used to build applications like chatbots, document extraction, and AI assistants.

Haystack



Haystack is an open source framework for building production-ready LLM applications, retrieval-augmented generative pipelines and state-of-the-art search systems that work intelligently over large document collections.

BUILD & TRAIN MODELS

LinkedIn



LinkedIn provides an open source library Liger-Kernel to improve performance and resource optimization when training models. The library can improve training throughput by 20% and reduce memory usage by 60% with just a single line of code for popular models like Llama, Gemma, and Qwen.

Unslloth



With unslloth AI finetune Llama 3.3, Mistral, Phi-4, Qwen 2.5, Gemma & DeepSeek-R1 2x faster with 80% less memory. Open source solution!

Arthur



Arthur AI itself is not entirely open source, their primary open-source offering is called "Arthur Bench," which is a tool designed to evaluate and compare different Large Language Models (LLMs) allowing users to freely access and contribute to its code.

Vectara



Vectara offers their popular open-source HHEM hallucination evaluation model. The HHEM model series are designed for detecting hallucinations in LLMs. They are particularly useful in the context of building retrieval-augmented-generation (RAG) applications where a set of facts is summarized by an LLM, and HHEM can be used to measure the extent to which this summary is factually consistent with the facts.

Probabl



Probabl is the official operator of the scikit-learn brand, and its main contributor. Probabl's core mission is to develop and maintain commons for data science, from scikit-learn to a complete suite of tools and solutions for machine learning and artificial intelligence.

SECURITY, PRIVACY & GOVERNANCE

Credo AI



Credo AI's open-source component is called "Credo AI Lens," which is a Python package that acts as a comprehensive framework for assessing AI systems, focusing on aspects like fairness, performance, explainability, and data profiling, allowing developers to evaluate the responsible development and deployment of AI models

Protect AI



Protect AI has released several key tools within their platform as open source, including "NB Defense", "ModelScan", and "Rebuff", which are available on GitHub and designed to detect vulnerabilities in machine learning systems; additionally, their "LLM Guard" tool is also open source and specifically focuses on securing LLMs.

Giskard



Giskard is an open-source Python library that automatically detects performance, bias & security issues in AI applications. The library covers LLM-based applications such as RAG agents, all the way to traditional ML models for tabular data.

VECTOR DATABASES

Chroma



Chroma is an AI-native open-source vector database used to simplify the development of LLM (Large Language Model) applications. It supports building these applications by making knowledge, facts, and skills easily pluggable for LLMs

Milvus



Milvus is an open-source vector database for embedding similarity search and AI applications. It aims to make unstructured data search more accessible and provides a consistent user experience across different deployment environments, including laptops, local clusters, and the cloud

Qdrant



Qdrant is a vector similarity search engine and vector database offering a production-ready service with an easy-to-use API for storing, searching, and managing vectors along with additional payload data. It provides extended filtering support, making it suitable for neural-network or semantic-based matching, faceted search, and other applications.

Weaviate



Weaviate is a cloud-native, open-source vector database that emphasizes speed and scalability. Using machine learning models, it transforms various types of data—text, images, and more—into a highly searchable vector database.

PostgreSQL



PostgreSQL is an open-source relational database that supports vector data through extensions like pgvector. This extension enables efficient similarity search on vector data, integrating with PostgreSQL's ecosystem.

Cassandra



Cassandra is a scalable NoSQL database for handling large amounts of data across many commodity servers, providing high availability with no single point of failure. With the introduction of vector search capabilities, Cassandra can efficiently manage vector data.

Cassandra



Redis is an in-memory data structure store known for its speed and flexibility. With the addition of the RedisAI module, it extends its capabilities to support vector data and AI model serving.

Supabase



Supabase is an open-source platform that helps developers build web and mobile applications. It includes a Postgres database, authentication, and real-time subscriptions.

Zilliz



Zilliz vector database management system – fully managed Milvus – supports billion-scale vector search and is trusted by over 10000 enterprise users

Why OPEN SOURCE AI FOR PRODUCTION



01 Cost Efficiency
Open-source AI eliminates or significantly reduces expenses related to licensing fees



02 Ease of Implementation
With robust community support, open-source AI tools are typically easier to integrate and deploy within existing systems.



03 Security, Risk, and Control
Using open-source AI allows organizations greater oversight and control over their systems. They can inspect, modify, and enhance the code as needed, which not only improves security but also ensures compliance with evolving regulatory requirements.



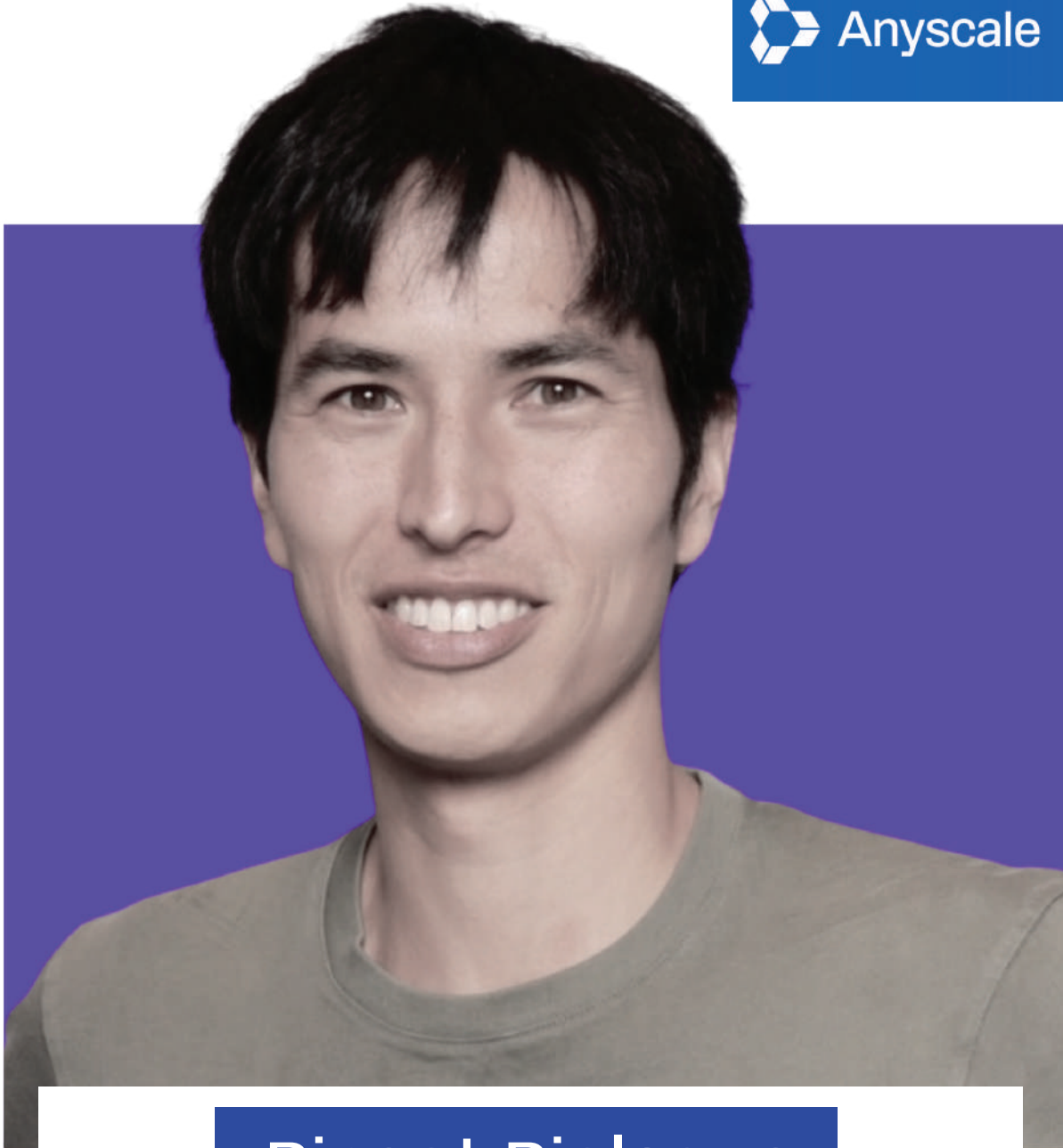
04 Resource Availability
The support ecosystem for open-source AI is extensive, encompassing community forums, developer groups, and third-party providers offering professional services. This wide range of resources aids in effective troubleshooting and continuous innovation.



05 Less Bias and More Transparency
The transparent nature of open-source software allows a diverse community of developers and users to scrutinize and refine the algorithms, greatly reducing biases and enhancing fairness in AI applications.



06 More Customizability
The inherent flexibility of open-source AI enables companies to tailor solutions to precisely fit their specific needs. Businesses can directly modify the source code to adapt the technology for unique operational workflows or to achieve enhanced performance metrics.



Direct Dialogue:

A Face-to-Face with

ROBERT NISHIHARA

CEO

Robert Nishihara is one of the creators of Ray, the world's leading open source AI compute engine. Ray is used by companies across the board from Canva to Runway to Coinbase to Pinterest to scale their machine learning training, inference, and data processing workloads. He is one of the co-founders of Anyscale, which is the company behind Ray. He did his PhD in machine learning and distributed systems in the computer science department at UC Berkeley. Before that, he majored in math at Harvard.

The Genesis of Ray: Robert, can you share the story behind Ray's inception and what inspired you to tackle the challenges of distributed computing in the AI space?

I started the PhD program at UC Berkeley in 2013 focused on machine learning algorithms. We were working on algorithms for reinforcement learning, optimization algorithms, deep learning, and so on. But as time went on, despite wanting to focus on algorithms, my collaborators and I found ourselves spending all of

our time managing clusters and building ad hoc distributed systems to scale our experiments. We realized that AI is incredibly computationally intensive, and there was an opportunity to build useful open source tools for scaling these compute intensive workloads. That led us to create Ray.

Overcoming Technical Hurdles: As Ray evolved into a robust framework, what were some of the most significant technical challenges you encountered, and how did your team overcome them?

In addition to the sheer scale of data and compute involved, a major one is that AI workloads use heterogeneous compute. We now live in a GPU-centric and accelerator-centric world, so Ray was designed with GPU compute in mind from day one. As AI compute workloads grow in complexity, AI workloads are dealing

with increasingly heterogeneous hardware. For example, many multimodal data processing pipelines mix CPU and GPU compute. CPUs for regular processing, GPUs for inference. Many previous distributed systems from the big data world like Spark, Hadoop, and Flink, were designed for CPUs. AI has different requirements.

Integration with Deep Learning Frameworks: How does Ray integrate with popular deep learning tools like TensorFlow and PyTorch, and what benefits does this bring to researchers and practitioners aiming to push AI boundaries?

Ray is complementary with deep learning frameworks like TensorFlow and PyTorch as well as with LLM inference engines like vLLM and TensorRT-LLM. While these engines focus on model optimizations and squeezing the most performance out of a GPU, Ray focuses on the challenges of distributed computing and multi-machine scaling. So in an oversimplified sense, these frameworks focus on optimizing the performance of a single model. Ray focuses on scaling that model across

a cluster, as part of a serving application, a training run, or a data pipeline.

All of this is also very complimentary with container management systems like Kubernetes. So a very common tech stack for AI compute will be something like Ray + Kubernetes + PyTorch + vLLM.

Open Source Impact: In the evolving landscape of open source AI, what role do you see Ray playing in democratizing access to scalable, production-ready machine learning solutions?

Ray is already used as the AI compute engine at the heart of many tech companies' ML platforms like Uber, Pinterest, ByteDance, Spotify, Canva, Reddit, Instacart, Coinbase, and many others.

As more and more businesses bring AI to production, our hope is to enable all of these businesses to leverage compute resources of all types to develop, scale, and deploy AI applications.

Future Roadmap: Looking ahead, can you reveal some exciting features or improvements on Ray's roadmap that you believe will further revolutionize distributed computing in AI?

One of the fastest growing workloads is multimodal data processing. More and more data processing is being done with AI. Businesses are using AI to get insights and value from their data. This is fundamentally an inference workload. What this means is that more and more data processing will become an AI workload running on GPUs. With Ray, we're investing heavily in

supporting these data intensive and accelerator intensive workloads.

Our focus with Ray is the same as ever, that is, providing a simple and performant API for scaling compute intensive workloads.



In Focus:
**YAM
MARCOVITZ**
Journey
&
Insights



Emcie
EXTREMELY COOL TECHNOLOGIES

YAM MARCOVITZ

An experienced software builder with extensive experience in mission-critical software and system architecture, Yam understands what it takes to create reliable, production-ready software. This background informs his distinctive approach to the development of predictable and aligned AI systems.

1. Can you share the initial vision behind Parlant and what specific gaps in the AI community you aimed to address with its launch?

The vision for Parlant came from a fundamental realization that “correct” AI interaction can’t be solved using a generic algorithm. It actually requires continuous integration of shifting and subjective business needs and expectations. We observed that large language models, no matter how advanced, make their own choices in conversation—unless steered and supervised by clear guidelines. Now, because some choices are inevitably arbitrary, they often fail to align with a company’s specific ethos and standards, making real-world adoption difficult, and often completely impractical. So we aimed to create a framework that allows organizations to reliably and dynamically dictate behavioral standards for their AI agents, according to their needs.

When we set out to create Parlant, many developers in the GenAI community could build a sophisticated chatbot, but there was no clear way to ensure the bot speaks and behaves in alignment with evolving business needs across the board. Parlant set out to fill that void. We wanted to support teams in delivering customer-facing agents that faithfully represented their brand and policies. To this day Parlant is unique in its approach to this challenge.

In building it, we had to focus on features that most LLM tooling overlooked: robust alignment controls and easy, real-time behavior updates. We knew from talking to people at large companies that once an AI agent was deployed, business experts inevitably wanted to tweak how it interacts with users or handles sensitive questions – tasks that might require days of fragile prompt tuning or even model retraining, depending on the approach. With Parlant, our vision was that such changes should be immediate and reliable, like updating a company policy document. We introduced the concept of “guided AI”, where small, atomically-defined rules and the context in which they apply all act as a constantly updatable playbook for the AI’s behavior.

Another gap we targeted was the lack of transparency and consistency in AI agent behavior. Parlant’s design enforces consistency by double-checking every response against the defined guidelines. We call this process supervision. This was a deliberate response to the AI community’s struggle with unpredictable LLM outputs. We wanted an AI agent that teams could trust, not just because it’s smart but also that it’s obedient to the rules you’ve set.

2. How does Parlant differentiate itself from other open-source AI projects in terms of its approach and value proposition?

Parlant actually occupies a unique niche focused on conversational AI governance. If we imagine the spectrum of AI projects, on one end we have orchestration frameworks like LangChain or multi-agent systems (CrewAI, AutoGen, etc.) that coordinate various AI tasks, and on the other end we have low-level libraries like OpenAI's function calling or guardrail systems that tackle a specific piece of the puzzle. Parlant sits somewhat in between – it's a full-fledged framework specifically made for building a single AI representative that needs strict behavior controls. When you actually focus on this so-called niche problem and try to solve it thor-

oughly, it turns out it's a huge problem, and people (especially large brands) really need it.

In a nutshell, Parlant is the framework you choose when consistency and compliance are more important than raw cleverness. For example, in enterprise AI use cases, if someone needs a GenAI chatbot that absolutely must follow company policy, Parlant stands out in a sea of more loosely guided systems. It might not have the name recognition of some larger projects yet, but technically speaking, it fills a gap. It's the specialized solution for safe, predictable AI interactions.

3. In what ways do you see Parlant as an influential project within the broader open-source AI ecosystem?

By launching Parlant as a fully open project, we wanted to rally the community around this approach, fostering shared best practices for reliable AI agents. Ultimately, the initial vision was to move the community from ad-hoc prompt engineering to a disciplined, transparent way of controlling conversational AI – treating AI behavior as a first-class, versionable artifact, much like code.

Even as a relatively new project, Parlant signals a shift in the open-source GenAI scene by pushing toward behavior-first AI development. Historically, most AI projects have emphasized model-centric improvements (bigger models, more data) or pipeline innovations. Parlant flips the script to ask: How do we operationalize AI behavior in real-world apps? By doing so in the open, we're trying to encourage the community to think beyond accuracy metrics and focus on alignment and reliability.

One way Parlant is influential is by demonstrating that strict control and flexibility can coexist in an LLM-based AI agent. We've shown it's possible to harness powerful LLMs but keep them on a short leash in terms of following business rules. This sets a bit of a blueprint for others. For example, developers who might

have used only prompt engineering are now considering declarative policy layers like Parlant's guidelines in their own projects. We've seen discussions in the community referencing Parlant's approach to dynamic guidelines as a novel solution to the "rogue AI" problem in chatbots. On Reddit, for example, we saw someone noted how Parlant provides a "unique value prop in the world of agent platforms", highlighting that virtually no other framework seems to be offering such out-of-the-box reliability controls.

Finally, the open-source AI community is increasingly fatigued from half-baked solutions, and looks for reference implementations of working design patterns. Parlant is an effective reference for safe and controlled AI deployment. Its contributions around deep input evaluation, output validation, and version-controlled AI policies could inform standards or inspire extensions in other projects. In the coming years, as concerns around AI safety and alignment grow, we anticipate that the approach we've taken with Parlant will influence industry guidelines for AI agent design. By demonstrating that alignment can be engineered systematically (and not just via training data), I think we're adding an important voice to the conversation.

4. How has the open-source community contributed to Parlant's evolution, and can you highlight some pivotal enhancements driven by community input?

The community involvement so far has been nothing short of amazing. It's done more than just adopting Parlant – they've actively helped shape critical architectural flows and components and are still doing so. One of the big advantages of developing a framework like Parlant out in the open, is that our mission of inte-

grating robustness into the flexible world of LLMs attracts heavy-hitters: senior engineers at large enterprises who've previously developed and deployed NLU-based chatbots to millions of users, accumulating extremely rare and valuable real-world experience in the process. Many of those experienced engineers are

Open Source AI

now tasked with finding a way to combine the power of LLMs with the strict alignment requirements of their companies. So we can all convene around a single table and join forces, contributing cutting-edge ideas and innovations into an open-source engine that's powerful enough to cater to such large-scale use cases. It's a clearly important and challenging task, so it attracts powerful intellects.

From day one, we knew we wanted to invite developers to join our efforts in refining the framework, and they've responded by providing both feedback and code. Community feedback influenced both external integrations as well as core engine enhancements. For instance, a popular community-driven add-on is the Parlant Q&A service, which allows a Parlant agent to answer FAQs from a managed Q&A knowledge base with a high degree of accuracy, seamlessly integrating frequently asked questions into the agent's repertoire. On the other end of the spectrum, users on Discord and

GitHub reported edge cases where their agent needed more nuanced control in certain contexts. Directly learning about the different challenges people are facing in large-scale customer-facing use cases helps us focus and leverage our team's deep R&D background to find uniquely tailored solutions.

It turns out that when you really understand a problem right at its source, along with its real-world context, it makes all the difference. This is exactly what community involvement gives you: instead of trying to solve grandiose theoretical problems in a vacuum, you can focus on immediate, practical enablement. It's like the whole world can be your design partner. It sounds simple, but I think that's what allows for real disruption: penetrating through the hype and noise and clearly seeing the challenges people are facing. In this way, a properly managed community project leads to more practical systems with broader applications.

5. Can you discuss Parlant's current position within the AI landscape, especially in relation to similar projects and solutions?

Compared to multi-agent frameworks (e.g. AutoGen, Camel), Parlant is not about having AIs talk to each other or splitting tasks among bots. Instead, it's about one agent being the best-behaved, most reliable "employee" you could deploy to interact with humans. This means that in scenarios where those frameworks might be overkill or under-controlled, Parlant is the go-to. It trades off the multi-agent breadth for depth in single-agent performance.

Relative to LangChain or Llamaindex, Parlant can actually be complementary. Those frameworks don't deeply address how the AI responds to the user. Parlant steps in at that stage: after you've retrieved knowledge or decided on an answer using (possibly) any of those frameworks, Parlant helps you ensure the answer is delivered correctly, in the right tone, at the right time, following all of your guidelines and policies.

In that regard, Parlant is more like the last-mile governance layer in a customer-facing solution. If you look at it that way, Parlant is relatively unique; few open-source projects provide this conversation policy enforcement as their primary function. Projects like Nvidia's NeMo Guardrails or Microsoft's Guidance are probably the closest analog, as they also aim to enforce rules on LLM

outputs. But, while conceptually they share similarities with Parlant, these projects are quite low-level, limited, and passive or "avoiding" in their approach. In contrast, Parlant offers a richer and more flexible behavioral framework to work around (and with) conversational edge cases using active behavioral guidance. So instead of just saying, "Here's the only input I'm willing to accept, otherwise I error out," you can say, "Here's exactly how I want to approach it when someone brings up this or that topic."

Compared to most frameworks, Parlant's is more like a specialist in an ecosystem of generalists. Instead of trying to cater to everything and everyone; it does one thing, and it does it well. My wife (a proper foodie) taught me, "If you go to a restaurant that offers both Sushi and Pizza, you're not going to get high-quality Sushi or Pizza. Decide what you want to eat, and let's go to a place that specializes in that one thing." So in the same vein, Parlant leaves data retrieval and platform integrations to others. It doesn't manage teams of agents. But it does ensure that the interaction between a particular AI agent and a human meets high standards of correctness and alignment. Projects that need that outcome need something like Parlant, highlighting its specific spot in the AI toolkit universe.

6. How do you plan to balance innovation with stability in Parlant as the project scales and more contributors get involved?

Balancing rapid innovation with stability is something we're very mindful of as Parlant's community grows. We're also trying to learn from where other projects didn't do so well. Fundamentally, we can break our

strategy down to 3 key components: (1) A strong core architecture, protected by regression tests, (2) A modular design, and (3) Delegating ownership and responsibility to experienced technical stakeholders in our community.

A strong core architecture encourages experimentation on the periphery while protecting the project's stable core. As to this core, we require that all substantial code contributions pass our regression test suite and come with their own corresponding tests. This guarantees that new features don't break existing functionality. We actually include stochastic tests with expected behavior ranges (given the inherent randomness of LLMs) to catch if a change makes the agent start deviating from guidelines unpredictably.

We also employ a modular design to contain innovation. Parlant's engine has clearly defined components that work in unison – guideline matching, tool invocation, response generation, etc. We encourage contributors to follow suit and design their improvements in a modular way. For instance, if someone has an idea for a new way to score guideline relevance using an ML model, we suggest implementing it as a module rather than over-

hauling the core logic. That way, the main stable branch of Parlant remains reliable, and new ideas can be tried in parallel. Once an experimental feature proves its worth over time, we can integrate it carefully into the core.

New contributors are encouraged to discuss changes in an issue or our Discord first. In having that dialogue, we try to ensure the architecture develops in a future-compatible way. As repo maintainers, we prioritize stability. No matter how exciting a new capability is, it won't be merged until we're confident in its impact. This might involve thorough code review and perhaps beta testing with a subset of the community for a few weeks. Only after incorporating sufficient feedback does it hit mainline. We plan to maintain this approach as the project scales.

7. Could you elaborate on the types of applications and use cases you see benefiting most from Parlant's unique approach?

Parlant's approach is most crucial where precision and compliance need to work alongside adaptability and high-coverage. One prime category is regulated or sensitive industries. For example, in financial services (like banking or insurance), there are strict regulations on what can be advised or how disclosures are given. Parlant shines here because you can encode those regulations as guidelines the agent must follow ("Never make a guarantee about investment returns" or "Always include compliance disclaimer X when initiating certain actions"). The framework then ensures the AI respects these rules, giving companies confidence that every customer interaction is compliant. Similarly, **healthcare communications** benefit from Parlant's ability to implement caution and clarity. You can enforce guidelines such as "If the patient mentions chest pain, always suggest seeking immediate medical help" or maintaining a reassuring tone for anxious users. In these domains, both guardrailing against unauthorized answers and knowing how to connect with the customer are important from a business perspective. So the ability to reliably follow expert-designed protocols is invaluable.

Compliance-focused and brand-sensitive customer service agents also greatly benefit. Think of a global brand that has a very specific tone of voice and policy on how to handle complaints. Parlant allows their AI agent to embody the unique approach they've developed in-house. You could have rules like "Always apolo-

gize first if customer is upset" or style guidelines ("Use friendly language, call the user by name, and never use slang"). Many standard chatbot platforms can't enforce tone at a granular level, but Parlant can. It helps you ensure that every support ticket response is on-brand, personalized and courteous.

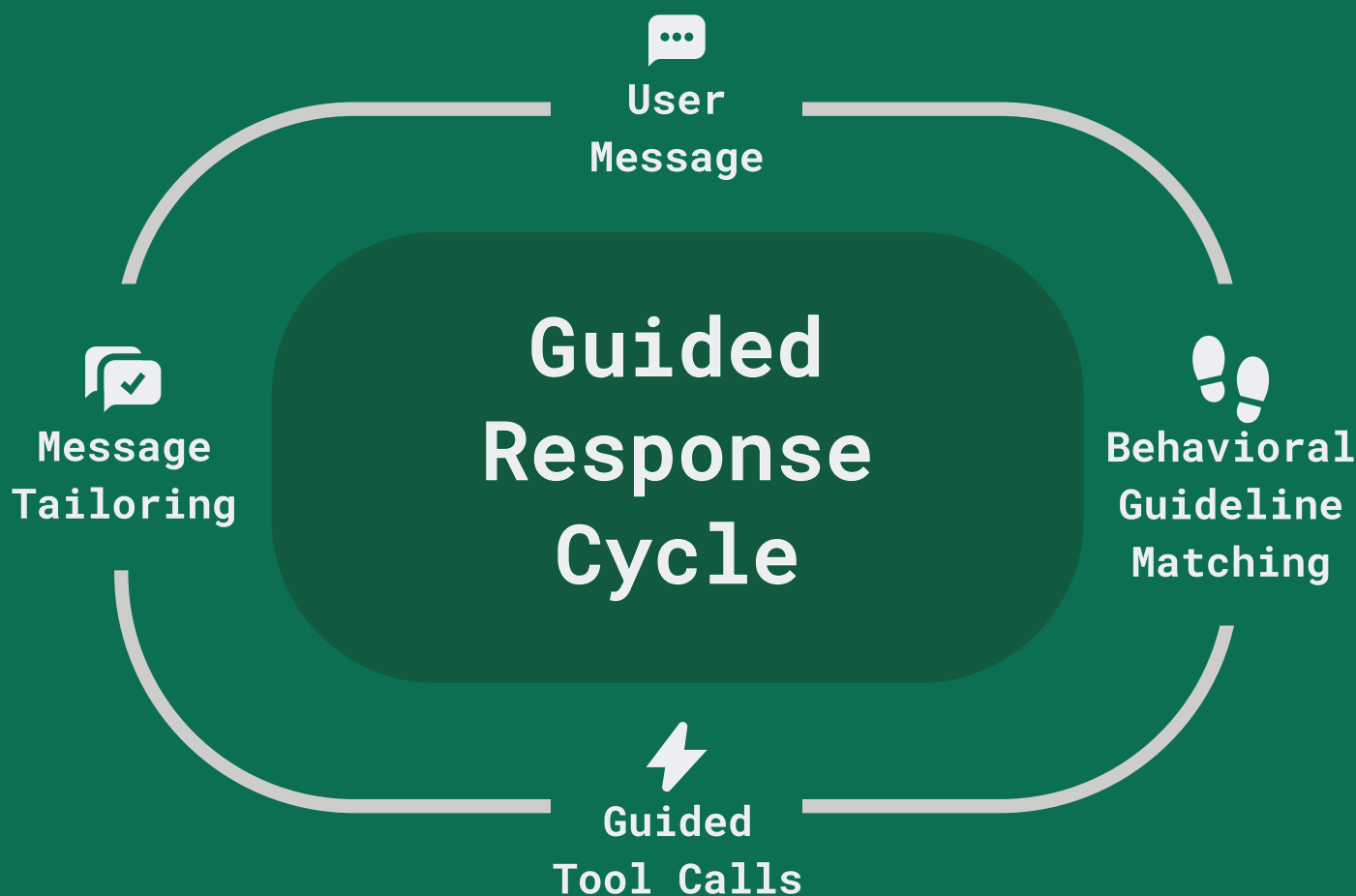
Parlant is also naturally suited for **multi-cultural optimization**. Because guidelines can include specifics about language usage based on which customer the agent is talking to, a company can communicate effectively with its customers across multiple locales just by switching guideline sets. For instance, politeness norms differ by culture, and guidelines can enforce using formal titles in one language vs. a casual tone in another, all within the same Parlant agent.

Basically, the applications that benefit most from Parlant are those where *what is said and how it's said matters as much as being correct*. Industries like finance, healthcare, legal, and high-touch customer service – those were front of mind in Parlant's design. Any use case where an AI agent must consistently represent **specific policies, ethics, or branding** is where Parlant's guided approach delivers the most value. Conversely, if you just needed a one-off Q&A bot with no special constraints, Parlant might be overkill. But sustained, policy-bound interaction – that's where it shines best.

Flow-Based Conversation Design Is Dead.

👑 Long Live GenAI Conversation Design.

Parlant is an open-source, LLM-first conversation design framework that gives engineers and designers the control they need to build hallucination-free, high-containment AI customer service for enterprise use cases.



unsloth AI

How a Two-Person Team Is Revolutionizing LLM Training with GPU Kernel Magic and Open-Source Innovation

Fine-tuning large language models (LLMs) often feels intimidating and painfully slow – complex setups, memory-intensive requirements, and frustratingly buggy implementations have kept us waiting days for meaningful results. But Unsloth AI, founded by Australian brothers Daniel and Michael Han, is revolutionizing the AI landscape, making fine-tuning not only dramatically faster and simpler but also reliable. Trusted and used by industry leaders like Google, Microsoft, Hugging Face, NVIDIA, and Meta, Unsloth also quickly identifies and fixes critical bugs in major open models (such as Google's Gemma and Microsoft's Phi-4), with contributions praised by top AI researchers like OpenAI co-founder Andrej Karpathy. With over 8 million monthly downloads and a vibrant, welcoming community of developers, Unsloth empowers everyone – from enterprise teams running multi-GPU clusters to enthusiasts fine-tuning models right from their personal laptops.

You can get started with fine-tuning locally with Unsloth by just having 3GB VRAM, or you can train for free using their Google Colab and Kaggle notebooks which provide free GPUs. Alternatively, their open-source UI allows you to fine-tune models without writing a single line of code. With Unsloth, you spend less time waiting and more time creating – because training your own LLM should be easy, reliable, and fun.

And with DeepSeek's recently released R1 reasoning model, Unsloth has already implemented Group Relative Policy Optimization (GRPO), the algorithm that was used to train R1. This allows users to effortlessly turn any open LLM like Llama or Mistral into powerful reasoning LLMs. Their optimized GRPO implementation dramatically reduces VRAM usage, by as much as 90% – letting users reproduce sophisticated reasoning capabilities even on a device with just 5GB VRAM.

What is Fine-tuning?

Before we dive into Unsloth's framework and past work, we need to understand the meaning and use cases behind fine-tuning. Fine-tuning an LLM customizes its behavior, enhances domain knowledge, and optimizes performance for specific tasks. Fine-tuning is the process of updating the actual "brains" of the language model through some process called back-propagation. By fine-tuning a pre-trained model (e.g. Llama-3.1-8B) on a specialized dataset, you can:

Update Knowledge: Introduce new domain-specific information.

Customize Behavior: Adjust the model's tone, personality, or response style.

Optimize for Tasks: Improve accuracy and relevance for specific use cases.

Example usecases:

- Train LLM to predict if a headline impacts a company positively or negatively.

- Use historical customer interactions for more accurate and custom responses.
- Fine-tune LLM on legal texts for contract analysis, case law research, and compliance.

You can think of a fine-tuned model as a specialized agent designed to do specific tasks more effectively and efficiently. Fine-tuning can replicate all of RAG's capabilities, but not vice versa.

However, fine-tuning open models like Meta's Llama or Google's Gemma can not only be extremely complicated, painfully slow and memory-intensive, but also buggy. For example, Google's open model Gemma was released to subpar results, however Unsloth managed to find and fix 8 critical bugs in the model which greatly effected the model's performance. Their fixes were so pivotal that Andrej Karpathy (*one of OpenAI's co-founder*) gave them a shoutout on Twitter (X) and their fixes were upstreamed to the official Google and Hugging Face implementations.

Open Source AI

Also, fine-tuning the popular Llama model with standard methods originally took around 85 hours on a single GPU (*Introducing Unsloth*). Large models often won't even fit into a single GPU's memory, forcing you to use multiple expensive GPUs or reduce batch sizes to

make training feasible. Unsloth makes it easier, faster, and more accurate than ever to train custom AI models – completely free, and entirely on your own local hardware.

Unsloth's Breakthrough Optimizations

Unsloth, which is fully open-source on GitHub, was born to eliminate training obstacles with a series of groundbreaking optimizations. It can speed up LLM training by up to 2x compared to all other fine-tuning implementations including those using Flash Attention 2 (FA2). (*Unsloth AI - Open Source Fine-Tuning for LLMs*). These claims were verified by Pytorch and Hugging Face themselves with the latter even conducting benchmarks showcasing Unsloth's performance compared to other implementations.

This is done without accuracy degradation but rather through low-level kernels and math algorithms. In an Alpaca example, Unsloth's enterprise version slashed the fine-tuning time from 85 hours down to just about 3 hours – a night-and-day difference in productivity (Introducing Unsloth). What's more, Unsloth drastically reduces memory usage, in some cases cutting VRAM requirements by as much as 60–90% (*depending on model*) (*Unsloth AI - Open Source Fine-Tuning for*

LLMs). This means you can fine-tune larger models or use bigger batch sizes on the same hardware (*Introducing Unsloth*). And all these speed and memory gains come with no special hardware needed – Unsloth's optimizations are pure software, so you can turn your existing GPU (*even an older or smaller one*) into a fine-tuning powerhouse. Fine-tuning that used to take days or top-tier GPUs can now be done in mere hours on a single device, without sacrificing model performance.

On multiGPUs, Unsloth is 2x faster and uses 40% less VRAM usage per GPU compared to Hugging Face + FA2. On an 8 GPU setup, you can expect your training speed to be 16x faster than before! Previously if a model like Llama 3.3 (70B) did not fit on a single GPU, you can now make it fit using 2x GPUs and training will be 4x faster!

How Unsloth Works Its Magic

What makes Unsloth so fast and efficient? Under the hood, it uses a combination of advanced techniques and engineering tricks to supercharge the training process:

- **Custom GPU Kernels:** The Unsloth team rewrote many training operations from scratch in highly optimized GPU code (using tools like OpenAI's Triton). By manually deriving gradients and hand-writing bespoke GPU kernels, Unsloth squeezes out every drop of performance from the hardware. These custom kernels bypass the usual overhead of generic frameworks, magically accelerating training without any hardware change.
- **Dynamic Quantization:** Unsloth supports dynamic 4-bit fine-tuning (QLoRA), which significantly reduces

memory usage while preserving most of its accuracy at the same time. By selectively quantizing layers, Unsloth's dynamic quantization is done in a smart way that preserves most of the model's accuracy, so you don't lose model quality while saving memory.

- **ML Algorithms:** Techniques like Unsloth's gradient checkpointing algorithm (saving intermediate states and recomputing them on the fly) and in-place operations are used to minimize memory overhead. In practice, this means far lower peak memory during training – users have reported seeing less than half the usual memory usage for the same fine-tuning task. By carefully managing allocations and computation order, Unsloth ensures your GPUs aren't sitting idle or memory-starved at any point.

Ease of Use

Unsloth is easy to adopt into your existing workflow. It's built on PyTorch and Hugging Face, which means you can use Unsloth's features inside the familiar PyTorch training loops or Hugging Face Transformers APIs with minimal changes. In fact, installing Unsloth is as simple as a 'pip install unsloth' command on Linux devices, and you're ready to go. The library also works directly on Windows and provides interfaces (like FastVisionModel

or Unsloth-optimized model classes) that mirror the usual Hugging Face model classes, so you can load models and fine-tune them just as you normally would – only much faster. There's no need to learn a new framework from scratch; Unsloth fits right in with the tools you already know and love, making acceleration virtually plug-and-play for developers.

BUG FIXES + OPEN-SOURCE CONTRIBUTIONS

Continuous Bug Fixes & Optimization

Unsloth's developers iterate quickly to squash bugs and improve performance. The team actively tracks issues

in new model releases and pushes out fixes often within days.

Trusted by Industry Leaders

Unsloth's impact has already been noticed by teams at major tech companies like Hugging Face, Microsoft, NVIDIA, and Meta, who have been adopters of Unsloth but have also collaborated with them previously. Some recent fixes include:

Google Gemma Models: Unsloth found and fixed 8 critical bugs in Google's Gemma LLMs, working with Google's team (*earning a shout-out from Andrej Karpathy*). These fixes resolved loss discrepancies and other issues that hindered Gemma's initial promise. A detailed case study will be shown below.

Microsoft's Phi-4: Unsloth identified and addressed several issues in Microsoft's Phi-4 model to enhance its performance and their fixes were up-streamed to the official Phi-4 model. These fixes collectively improved the accuracy and reliability of the Phi-4 model, as evidenced by higher scores on benchmarks like the OpenLLM Leaderboard.

Meta's Llama 3: The team resolved numerous fine-tuning issues in Meta's Llama 3, greatly improving training stability and accuracy. Many community members had found Llama 3 fine-tunes "broken," but Unsloth identi-

fied causes (e.g. *double BOS tokens, untrained special tokens*) and auto-fixes them behind the scenes. This ensures Llama 3 models fine-tune correctly with Unsloth when they otherwise would fail.

Universal Training Bugs: Unsloth even patched a universal Gradient Accumulation bug that affected everyone's LLM training (*not just Unsloth users*). The flaw caused higher losses with accumulated batches; Unsloth's fix ensures accurate loss computation during training. This improvement was up-streamed so that the entire community benefits, reflecting Unsloth's commitment to broader optimization.

Through these and many other fixes (e.g. *issues in Gemma 2, Qwen-2.5, Phi-4*), Unsloth continuously fine-tunes its engine. Each bug fix is quickly rolled into the library, boosting performance or stability. This rapid, proactive approach means Unsloth users always have an up-to-date tool that "just works," even as new models and issues emerge. The result is a fast-evolving project that prioritizes robust, correct training for everyone.

Case study: Unsloth's Gemma bug fixes

Unsloth identified and addressed several critical issues in Google's Gemma language model. Previously the Gemma model would give poor, gibberish or endless outputs however, Unsloth's bug fixes solved all these issues. Unsloth's changes were up-streamed to the official Gemma and Hugging Face implementations:

1. Inclusion of the <bos> Token:

The <bos> (Beginning of Sequence) token was missing, causing discrepancies during fine-tuning. Incorporating this token resolves accuracy issues.

2. Correction of <end_of_turn> Token Typo:

Fixed typo from <end_of_turn>model to <end_of_turn>, ensuring proper parsing of conversational data.

3. Precision in Embedding Scaling:

Addressed precision errors from casting embedding scaling to bfloat16. Precise calculation prevents stability issues.

4. Layer Normalization in float32:

Improved numerical stability by performing layer normalization in higher precision (float32) rather than lower precision formats.

5. RoPE Position Encoding Precision:

Corrected RoPE (Rotary Position Embeddings) position casting to int32, aligning with best practices and maintaining precision over long sequences.

6. RoPE Calculation Methodology:

Changed calculation method for RoPE embeddings from reciprocal multiplication ($a \cdot (1/x)$) to direct division (a/x), reducing computational errors.

7. RoPE Computation in float32:

Improved precision of RoPE computations by transi-

tioning from bfloat16 to float32, integrated in Hugging Face Transformers v4.38.2.

8. Approximate GELU Activation:

Replaced the exact GELU activation function with the computationally efficient approximate (tanh) version, enhancing performance without significant accuracy loss.

Collaborations & Integrations

Unsloth thrives on a community-driven development model, collaborating with both open-source contributors and industry leaders. The project was built to be friendly and plug-and-play with popular AI ecosystems. For example, Unsloth is fully compatible with Hugging Face's Transformers and TRL libraries, and Hugging Face even partnered with Unsloth for an official integration blog post. This deep integration means you can use Unsloth with Hugging Face trainers (SFT, DPO, PPO) out of the box, speeding up fine-tunes with no code changes. The Unsloth team (a duo of two brothers) actively engages with the open-source community on GitHub and Discord, welcoming contributions and feedback.

Crucially, Unsloth's development involves direct collaboration with model creators and other AI projects. The team works closely with researchers at companies like Google – for instance, they coordinated with Google's Gemma team to patch bugs in Gemma-2 models. They directly support llama.cpp, Ollama and vLLM (open-source inference engines) to ensure fixes for Llama and Gemma propagate across tools. The direct integrations also allow for easy exporting of fine-tuned models to llama.cpp, Open Web UI etc. These partnerships highlight Unsloth's role as a connector in the AI ecosystem – it doesn't work in isolation, but rather integrates with the best tools and helps improve them.

Unsloth's technology aligns with AI leaders' platforms as well. It offers first-class support for Meta's Llama family – from Llama 2 to the latest Llama 3 – including vision variants. When Llama 3.3 launched, Unsloth immediately enabled fine-tuning of even the 70B model on a single 41 GB GPU, and fixed bugs in Llama 3's base weights to ensure finetunes weren't "useless or poor". Similarly, Unsloth stays in sync with Microsoft's open models: the team quickly updated Unsloth to support Microsoft's Phi-3 and Phi-4 release (Llama-fying it for compatibility), and community benchmarks showed an Unsloth Dynamic Quantization of Phi-4 matching the original Microsoft version's accuracy. On the hardware side, Unsloth is optimized for NVIDIA GPUs at every level – it supports everything from a GTX 1070 to the high-end H100 tensor core GPU. By leveraging custom GPU kernels (via OpenAI's Triton) and advanced attention optimization, Unsloth squeezes maximum training efficiency out of NVIDIA hardware. This broad compatibility means whether you're using Meta's latest model, fine-tuning a Microsoft-released LLM, or running on NVIDIA GPUs in the cloud or locally, Unsloth integrates seamlessly into your workflow. It's a true community-driven project, built hand-in-hand with researchers, engineers, and enthusiasts across the AI landscape.

Usage Statistics & Real-World Impact

By the numbers, Unsloth's impact is impressive:

Speed: Fine-tuning jobs run dramatically faster. Unsloth often achieves ~2x faster training out-of-the-box, and in some benchmarks it's been up to 2.7x faster than standard Hugging Face implementations + Flash Attention (FA2).

Memory Efficiency: GPU memory usage drops 50–90% on average when using Unsloth's optimized kernels. In best-case scenarios, it has shown a 74% reduction in VRAM needed for the same task. Lower memory footprint lets users fine-tune larger models or batch sizes on the same hardware (or even use smaller GPUs than normally required).

Context Length: Unsloth pushes the limits of context windows. It enables up to 13x longer on certain ones (e.g. Llama 3.3) by combining RoPE scaling and gradient checkpointing. This unlocks training and using LLMs on extremely long documents or dialogues that vanilla setups couldn't handle.

Community Adoption: The AI community has embraced Unsloth enthusiastically. Its models on Hugging Face have been downloaded 8 million+ times per month, and the library's GitHub repository surpassed 30,000 stars. Unsloth was also selected for the GitHub Accelerator Program, underscoring its potential and community support.

These gains translate to real-world benefits for AI developers and researchers. Unsloth's optimizations make advanced LLM fine-tuning more accessible and faster for everyone. For example, tasks that once demanded an expensive high-memory GPU can now be done on a consumer-grade setup or even free cloud tiers. The Unsloth team often demonstrates this with community notebooks – e.g. using Unsloth, a 12–14 billion parameter model can be fine-tuned on a free Colab GPU (15–16 GB), something previously infeasible. One community member noted that a Phi-3 14B model “fits comfortably in a free Tesla T4 Colab with Unsloth, and you can fit 3–4× longer context lengths than [FlashAttention] 2”. In other words, Unsloth lets you do more with less – more sequence length, bigger models, or larger batches on the same hardware, which accelerates experimentation and iteration in AI workflows.

Importantly, this speed-up and efficiency come with no downsides to model performance. Unsloth was designed to preserve exact accuracy while boosting throughput. There's ample evidence that faster fine-tuning works without accuracy degradation: for instance, an Unsloth's Phi-3 model finetune achieved comparable accuracy to the original Microsoft-trained version on the HuggingFace Open LLM Leaderboard. In fact, Unsloth's version even attained a slightly lower training loss than the official release, thanks to precision improvements in its kernels. Community contributors have replicated these results, showing that Unsloth-trained models perform on par with (*or even better than*) standard-trained models, all while saving time and memory.

Conclusion

In summary, Unsloth AI has rapidly grown into a beloved tool in the LLM fine-tuning community. Its friendly, enthusiastic developers engage with users to drive constant improvements, embodying a spirit of open-source collaboration and rapid iteration. From fixing bugs that improve model reliability, to collaborating with major enterprises for seamless integration, to delivering tangible speed and efficiency boosts – Unsloth has had a tangible impact on modern AI workflows. It empowers researchers and hobbyists alike to fine-tune large models quickly, on modest hardware, and with confidence that results will be accurate. With Unsloth's continuous optimization and community-driven development ethos, the process of training

custom AI models is becoming faster, easier, and more accessible for all. The sloth may be the project's mascot, but Unsloth is helping the AI world move at a lightning pace – all while keeping a friendly, supportive vibe along the journey.

Be sure to ★Star their GitHub repo or join their Discord, Twitter (x) and Reddit communities to show your support or ask any questions!

Train your own reasoning model



unslloth



deepseek

- Train your own reasoning model with just **5GB VRAM**
- **10x longer context lengths + 90% less VRAM**
- No accuracy degradation + Open-Source

Prompt: Which is bigger? 9.11 or 9.9?

Before (Phi-4 with no reasoning)

9.11 is bigger than 9.9. When comparing decimal numbers, you look at the digits from left to right. Both numbers have the same whole number part (9), so you compare the digits in the tenths place next. In 9.11, the tenths place is 1, and in 9.9, the tenths place is 9. Since 1 is less than 9, you might initially think 9.9 is larger, but you also need to consider the hundredths place in 9.11, which is 1. When you express 9.9 as 9.90 for comparison, you see that 9.11 is greater than 9.90. Therefore, 9.11 is bigger than 9.9.

After (Phi-4 with Unslloth + GRPO)

<reasoning>

To determine which number is bigger between 9.11 and 9.9, we should compare the two numbers digit by digit from left to right.

1. First, compare the digits in the units place:
 - Both numbers have a 9 in the units place.
2. Next, compare the digits in the tenths place:
 - The number 9.11 has a 1 in the tenths place.
 - The number 9.9 has a 9 in the tenths place.

Even if we compare the hundredths place, 1 is greater than 0, but this is irrelevant since the comparison in the tenths place already determines that 9.11 is smaller than 9.9. Therefore, 9.9 is greater than 9.11. </reasoning>

<answer> 9.9 is bigger than 9.11. </answer>

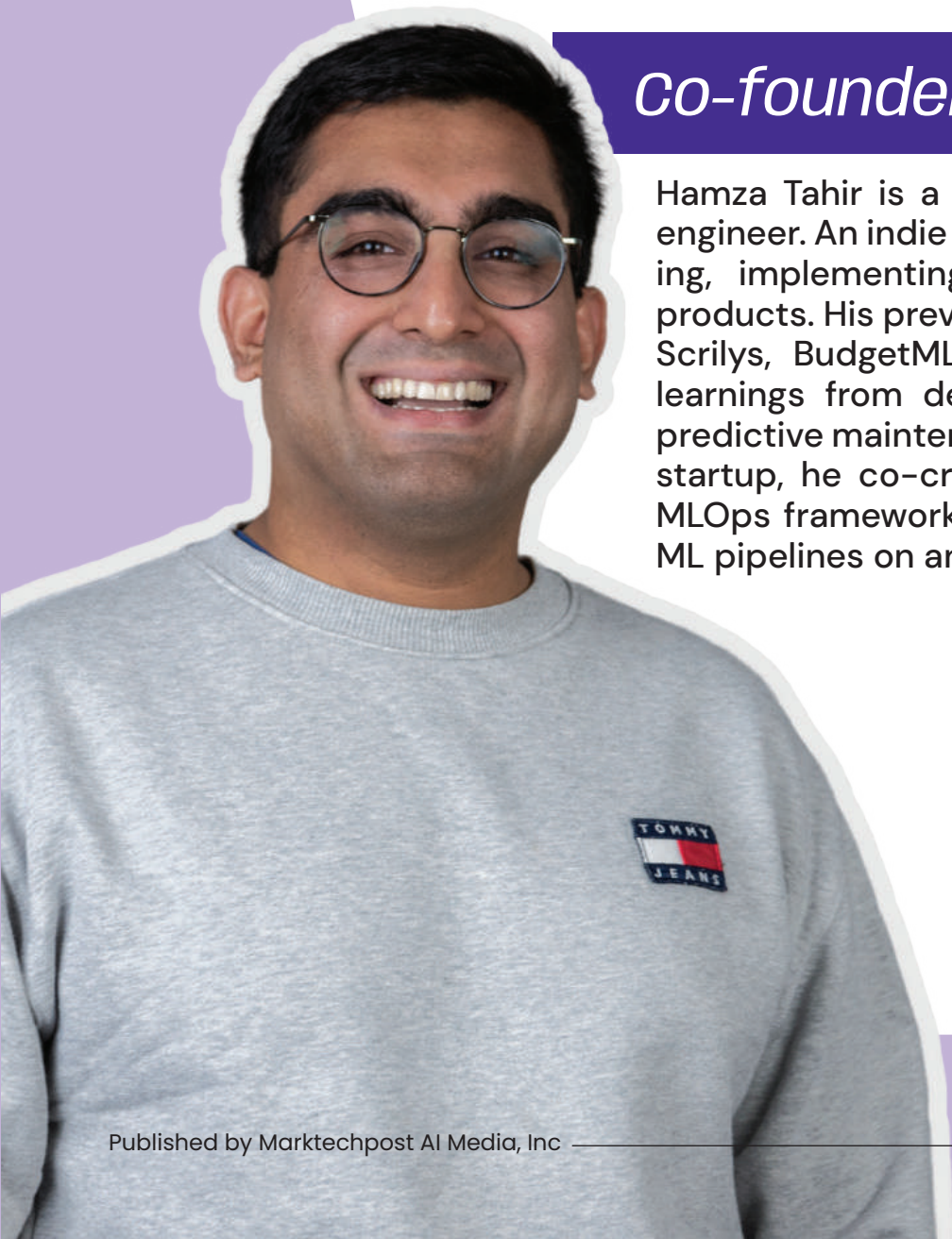


Interview with

HAMZA TAHIR

Co-founder & CTO of ZenML

Hamza Tahir is a software developer turned ML engineer. An indie hacker by heart, he loves ideating, implementing, and launching data-driven products. His previous projects include PicHance, Scrilys, BudgetML, and you-tldr. Based on his learnings from deploying ML in production for predictive maintenance use-cases in his previous startup, he co-created ZenML, an open-source MLOps framework for creating production grade ML pipelines on any infrastructure stack.



From Early Projects to ZenML: Given your rich background in software development and ML engineering—from pioneering projects like BudgetML to co-founding ZenML and building production pipelines at maiot.io—how has your personal journey influenced your approach to creating an open-source ecosystem for production-ready AI?

My journey from early software development to co-founding ZenML has deeply shaped how I approach building open-source tools for AI production. Working on BudgetML taught me that accessibility in ML infrastructure is critical – not everyone has enterprise-level resources, yet everyone deserves access to robust tooling.

At my first startup maiot.io, I witnessed firsthand how fragmented the MLOps landscape was, with teams cobbling together solutions that often broke in production. This fragmentation creates real business pain points – for example, many enterprises struggle with lengthy time-to-market cycles for their ML models due to these exact challenges.

These experiences drove me to create ZenML with a focus on being production-first, not production-even-

tual. We built an ecosystem that brings structure to the chaos of managing models, ensuring that what works in your experimental environment transitions smoothly to production. Our approach has consistently helped organizations reduce deployment times and increase efficiency in their ML workflows.

The open-source approach wasn't just a distribution strategy—it was foundational to our belief that MLOps should be democratized, allowing teams of all sizes to benefit from best practices developed across the industry. We've seen organizations of all sizes—from startups to enterprises—accelerate their ML development cycles by 50–80% by adopting these standardized, production-first practices.

From Lab to Launch: Could you share a pivotal moment or technical challenge that underscored the need for a robust MLOps framework in your transition from experimental models to production systems?

ZenML grew out of our experience working in predictive maintenance. We were essentially functioning as consultants, implementing solutions for various clients. A little over four years ago when we started, there were far fewer tools available and those that existed lacked maturity compared to today's options.

We quickly discovered that different customers had vastly different needs—some wanted AWS, others preferred GCP. While Kubeflow was emerging as a solution that operated on top of Kubernetes, it wasn't yet

the robust MLOps framework that ZenML offers now.

The pivotal challenge was finding ourselves repeatedly writing custom glue code for each client implementation. This pattern of constantly developing similar but platform-specific solutions highlighted the clear need for a more unified approach. We initially built ZenML on top of TensorFlow's TFX, but eventually removed that dependency to develop our own implementation that could better serve diverse production environments.

Open-Source vs. Closed-Source in MLOps: While open-source solutions are celebrated for innovation, how do they compare with proprietary options in production AI workflows? Can you share how community contributions have enhanced ZenML's capabilities in solving real MLOps challenges?

Proprietary MLOps solutions offer polished experiences but often lack adaptability. Their biggest drawback is the "black box" problem—when something breaks in production, teams are left waiting for vendor support. With open-source tools like ZenML, teams can inspect, debug, and extend the tooling themselves.

This transparency enables agility. Open-source frameworks incorporate innovations faster than quarterly releases from proprietary vendors. For LLMs, where best practices evolve weekly, this speed is invaluable.

The power of community-driven innovation is exemplified by one of our most transformative contributions—a developer who built the "Vertex" orchestrator integration for Google Cloud Platform. This wasn't just another integration—it represented a completely new approach to orchestrating pipelines on GCP that opened up an entirely new market for us.

Prior to this contribution, our GCP users had limited options. The community member developed a comprehensive Vertex AI integration that enabled seamless orchestration in the Google Cloud ecosystem. Today,

it's one of our most popular integrations, powering production ML systems across numerous enterprises and significantly expanding our user base.

This perfectly demonstrates the value of open-source in MLOps—community members solve their specific

challenges and contribute back, creating value for the entire ecosystem. That said, the ideal approach is often hybrid—using open-source frameworks as core infrastructure while integrating specialized proprietary tools where they add unique value.

Integrating LLMs into Production: With the surge in generative AI and large language models, what are the key obstacles you've encountered in LLMOps, and how does ZenML help mitigate these challenges?

LLMOps presents unique challenges including prompt engineering management, complex evaluation metrics, escalating costs, and pipeline complexity.

ZenML helps by providing:

- Structured pipelines for LLM workflows, tracking all components from prompts to post-processing logic
- Integration with LLM-specific evaluation frameworks

- Caching mechanisms to control costs
- Lineage tracking for debugging complex LLM chains

Our approach bridges traditional MLOps and LLMOps, allowing teams to leverage established practices while addressing LLM-specific challenges. ZenML's extensible architecture lets teams incorporate emerging LLMOps tools while maintaining reliability and governance.

Streamlining MLOps Workflows: What best practices would you recommend for teams aiming to build secure, scalable ML pipelines using open-source tools, and how does ZenML facilitate this process?

For teams building ML pipelines with open-source tools, I recommend:

- Start with reproducibility through strict versioning
- Design for observability from day one
- Embrace modularity with interchangeable components
- Automate testing for data, models, and security
- Standardize environments through containerization

ZenML facilitates these practices with a Pythonic framework that enforces reproducibility, integrates with popular MLOps tools, supports modular pipeline steps, provides testing hooks, and enables seamless containerization.

We've seen these principles transform organizations like Adeo Leroy Merlin. After implementing these best practices through ZenML, they reduced their ML development cycle by 80%, with their small team of data scientists now deploying new ML use cases from research to production in days rather than months, delivering tangible business value across multiple production models.

The key insight: MLOps isn't a product you adopt, but a practice you implement. Our framework makes following best practices the path of least resistance while maintaining flexibility.

Engineering Meets Data Science: Your career spans both software engineering and ML engineering—how has this dual expertise influenced your design of MLOps tools that cater to real-world production challenges?

My dual background has revealed a fundamental disconnect between data science and software engineering cultures. Data scientists prioritize experimentation and model performance, while software engineers focus on reliability and maintainability. This divide creates significant friction when deploying ML systems to production.

ZenML was designed specifically to bridge this gap by creating a unified framework where both disciplines can thrive. Our Python-first APIs provide the flexibility data scientists need while enforcing software engineering

best practices like version control, modularity, and reproducibility. We've embedded these principles into the framework itself, making the right way the easy way.

This approach has proven particularly valuable for LLM projects, where the technical debt accumulated during prototyping can become crippling in production. By providing a common language and workflow for both researchers and engineers, we've helped organizations reduce their time-to-production while simultaneously improving system reliability and governance.

MLOps vs. LLMOps: In your view, what distinct challenges do traditional MLOps face compared to LLMOps, and how should open-source frameworks evolve to address these differences?

Traditional MLOps focuses on feature engineering, model drift, and custom model training, while LLMOps deals with prompt engineering, context management, retrieval-augmented generation, subjective evaluation, and significantly higher inference costs.

Open-source frameworks need to evolve by providing:

- Consistent interfaces across both paradigms
- LLM-specific cost optimizations like caching and dynamic routing

- Support for both traditional and LLM-specific evaluation
- First-class prompt versioning and governance

ZenML addresses these needs by extending our pipeline framework for LLM workflows while maintaining compatibility with traditional infrastructure. The most successful teams don't see MLOps and LLMOps as separate disciplines, but as points on a spectrum, using common infrastructure for both.

Security and Compliance in Production: With data privacy and security being critical, what measures does ZenML implement to ensure that production AI models are secure, especially when dealing with dynamic, data-intensive LLM operations?

ZenML implements robust security measures at every level:

- Granular pipeline-level access controls with role-based permissions
- Comprehensive artifact provenance tracking for complete auditability
- Secure handling of API keys and credentials through encrypted storage
- Data governance integrations for validation, compliance, and PII detection

- Containerization for deployment isolation and attack surface reduction

These measures enable teams to implement security by design, not as an afterthought. Our experience shows that embedding security into the workflow from the beginning dramatically reduces vulnerabilities compared to retrofitting security later. This proactive approach is particularly crucial for LLM applications, where complex data flows and potential prompt injection attacks create unique security challenges that traditional ML systems don't face.

Future Trends in AI: What emerging trends for MLOps and LLMOps do you believe will redefine production workflows over the next few years, and how is ZenML positioning itself to lead these changes?

Agents and workflows represent a critical emerging trend in AI. Anthropic notably differentiated between these approaches in their blog about Claude agents, and ZenML is strategically focusing on workflows primarily for reliability considerations.

While we may eventually reach a point where we can trust LLMs to autonomously generate plans and iteratively work toward goals, current production systems demand the deterministic reliability that well-defined workflows provide. We envision a future where workflows remain the backbone of production AI systems, with agents serving as carefully constrained compo-

nents within a larger, more controlled process—combining the creativity of agents with the predictability of structured workflows.

The industry is witnessing unprecedented investment in LLMOps and LLM-driven projects, with organizations actively experimenting to establish best practices as models rapidly evolve. The definitive trend is the urgent need for systems that deliver both innovation and enterprise-grade reliability—precisely the intersection where ZenML is leveraging its years of battle-tested MLOps experience to create transformative solutions for our customers.

Fostering Community Engagement: Open source thrives on collaboration—what initiatives or strategies have you found most effective in engaging the community around ZenML and encouraging contributions in MLOps and LLMOps?

Open Source AI

We've implemented several high-impact community engagement initiatives that have yielded measurable results. Beyond actively soliciting and integrating open-source contributions for components and features, we hosted one of the first large-scale MLOps competitions in 2023, which attracted over 200 participants and generated dozens of innovative solutions to real-world MLOps challenges.

We've established multiple channels for technical collaboration, including an active Slack community,

regular contributor meetings, and comprehensive documentation with clear contribution guidelines. Our community members regularly discuss implementation challenges, share production-tested solutions, and contribute to expanding the ecosystem through integrations and extensions. These strategic community initiatives have been instrumental in not only growing our user base substantially but also advancing the collective knowledge around MLOps and LLMOps best practices across the industry.

Advice for Aspiring AI Engineers: Finally, what advice would you give to students and early-career professionals who are eager to dive into the world of open-source AI, MLOps and LLMOps, and what key skills should they focus on developing?

For those entering MLOps and LLMOps:

- Build complete systems, not just models—the challenges of production offer the most valuable learning
- Develop strong software engineering fundamentals
- Contribute to open-source projects to gain exposure to real-world problems
- Focus on data engineering—data quality issues cause more production failures than model problems

- Learn cloud infrastructure basics—Key skills to develop include Python proficiency, containerization, distributed systems concepts, and monitoring tools. For bridging roles, focus on communication skills and product thinking. Cultivate "systems thinking"—understanding component interactions is often more valuable than deep expertise in any single area. Remember that the field is evolving rapidly. Being adaptable and committed to continuous learning is more important than mastering any particular tool or framework.

How does ZenML's approach to workflow orchestration differ from traditional ML pipelines when handling LLMs, and what specific challenges does it solve for teams implementing RAG or agent-based systems?

At ZenML, we believe workflow orchestration must be paired with robust evaluation systems—otherwise, teams are essentially flying blind. This is especially crucial for LLM workflows, where behaviour can be much less predictable than traditional ML models.

Our approach emphasizes "eval-first development" as the cornerstone of effective LLM orchestration. This means evaluation runs as quality gates or as part of the outer development loop, incorporating user feedback and annotations to continually improve the system.

For RAG or agent-based systems specifically, this eval-first approach helps teams identify whether issues are coming from retrieval components, prompt engineering, or the foundation models themselves. ZenML's orchestration framework makes it straightforward to implement these evaluation checkpoints throughout your workflow, giving teams confidence that their systems are performing as expected before reaching production.

What patterns are you seeing emerge for successful hybrid systems that combine traditional ML models with LLMs, and how does ZenML support these architectures?

ZenML takes a deliberately unopinionated approach to architecture, allowing teams to implement patterns that work best for their specific use cases. Common hybrid patterns include RAG systems with custom-tuned embedding models and specialized language models for structured data extraction.

This hybrid approach—combining custom-trained models with foundation models—delivers superior results for domain-specific applications. ZenML sup-

ports these architectures by providing a consistent framework for orchestrating both traditional ML components and LLM components within a unified workflow.

Our platform enables teams to experiment with different hybrid architectures while maintaining governance and reproducibility across both paradigms, making the implementation and evaluation of these systems more manageable.

As organizations rush to implement LLM solutions, how does ZenML help teams maintain the right balance between experimentation speed and production governance?

ZenML handles best practices out of the box—tracking metadata, evaluations, and the code used to produce them without teams having to build this infrastructure themselves. This means governance doesn't come at the expense of experimentation speed.

As your needs grow, ZenML grows with you. You might start with local orchestration during early experimentation phases, then seamlessly transition to cloud-based orchestrators and scheduled workflows as you move toward production—all without changing your core code.

Lineage tracking is a key feature that's especially relevant given emerging regulations like the EU AI Act. ZenML captures the relationships between data, models, and outputs, creating an audit trail that satisfies governance requirements while still allowing teams to move quickly. This balance between flexibility and governance helps prevent organizations from ending up with "shadow AI" systems built outside official channels.

What are the key integration challenges enterprises face when incorporating foundation models into existing systems, and how does ZenML's workflow approach address these?

A key integration challenge for enterprises is tracking which foundation model (and which version) was used for specific evaluations or production outputs. This lineage and governance tracking is critical both for regulatory compliance and for debugging issues that arise in production.

ZenML addresses this by maintaining a clear lineage between model versions, prompts, inputs, and outputs across your entire workflow. This provides both technical and non-technical stakeholders with visibility into how foundation models are being used within enterprise systems.

Our workflow approach also helps teams manage environment consistency and version control as they move LLM applications from development to production. By containerizing workflows and tracking dependencies, ZenML reduces the "it works on my machine" problems that often plague complex integrations, ensuring that LLM applications behave consistently across environments.



Open Source & AI Hardware

EXCLUSIVE TALK

with Anita Lacea



Open Source & AI Hardware

About *Anita Lacea*

Anita has over 15 years of experience working at the intersection of hardware infrastructure and artificial intelligence. A mathematician and computer scientist by training, she currently serves as Director of Hardware Infrastructure Transformation at Microsoft, where she focuses on evolving Azure's cloud foundation to support next-generation AI workloads.

Her career to date has included leading Amazon's first non-chatbot generative AI product, and helping to establish Intel's first discrete GPU business.

Anita specializes in hardware infrastructure and enterprise-scale AI solutions. Her expertise spans deep learning systems, technology strategy, AI governance and security, and AI systems design. Throughout her career, she has consistently translated cutting-edge technology into practical business solutions, with a focus on building secure and responsible AI systems that deliver tangible results.

The following interview presents my personal professional perspective on how open source software ecosystems interact with and influence proprietary hardware development in AI. Drawing from my experience across multiple roles in the industry, I explore the technical challenges, opportunities, and innovations that emerge when open collaboration meets closed hardware design. These views are my own and do not represent the positions or policies of my current or previous employers. Our conversation examines how open source frameworks reshape proprietary silicon development, the technical misalignments between software and hardware lifecycles, and the future of specialized AI hardware in an increasingly open software landscape.

1. How did open source shape your approach on your journey in AI and hardware?

Open source has been transformative in my hardware and AI development journey, even though it has not been my primary focus. At Intel, working on the oneAPI inference kit showed me how an open standard could create a unified programming model that freed developers from proprietary ecosystems, allowing code to run efficiently across different architectures.

With the oneAPI AI reference kits, we observed measurable performance improvements and cost reductions across healthcare and manufacturing implementations that weren't achievable in closed environments. Hardware innovation progresses more effectively when

artificial barriers to collaboration are removed.

The most useful insight came from seeing how the community modified our reference implementations. Contributions to our PyTorch integration revealed applications we hadn't considered, which directly affected our hardware optimization strategies. This feedback between open source software development and hardware design continues to influence my approach to innovation — I'm constantly looking for signals from the community that indicate where computing workloads are evolving, which informs more resilient and future-proof hardware solutions.

2. How are open source AI frameworks democratizing access to advanced hardware technologies?

Open source AI frameworks have democratized access to advanced AI capabilities, creating a foundation upon which both open and proprietary solutions can thrive. Libraries like TensorFlow, PyTorch, and Keras enable individuals to develop AI applications without sophisticated hardware infrastructure upfront, while commercial platforms build specialized optimizations on these foundations. This symbiotic relationship drives innovation across the ecosystem—developers can build impressive models using consumer-grade GPUs or affordable cloud instances while still leveraging proprietary acceleration where it adds value.

The practical advantage comes from how open and proprietary solutions enhance each other through transparency and specialization. When core development happens collaboratively, both hardware and software solutions evolve more rapidly, with companies contributing optimizations that benefit their specific use cases. Nvidia's decision to open-source Run:ai,

their GPU orchestration platform, exemplifies this synergy—organizations can optimize resource allocation while commercial hardware accelerators continue to push performance boundaries. Small companies benefit from these collaborative tools to develop applications that might not appear on larger organizations' roadmaps, while proprietary solutions provide turnkey capabilities for enterprise-scale deployment.

These frameworks provide abstraction layers that hide low-level complexity, allowing users to implement models without managing tensor operations or memory allocation. This creates a practical feedback loop for hardware development—as more people use AI tools, their usage patterns inform the development of specialized hardware for specific applications. This relationship between accessible software and responsive hardware development continues to advance both fields.

3. What key challenges arise when merging open source software with specialized AI hardware, and how can they be overcome?

The primary challenge is mismatched development cycles. Hardware development has long lead times from design to production, while open source AI frameworks change much more quickly. I've seen instances where AI accelerators were designed for workloads that had evolved significantly by the time silicon was ready. This temporal disconnect creates a perpetual game of prediction—you're always targeting a moving objective that accelerates with each community contribution.

Software optimization in the open source world has become effective at delivering performance gains that sometimes rival hardware improvements. DeepSeek R1's use of 8-bit floating point (FP8) precision instead of 32-bit (FP32) reduces memory usage by 75% with min-

imal accuracy loss, enabling faster processing on standard hardware.

This creates a forecasting problem across the industry: predicting both future workloads and how software optimization might solve performance issues before new hardware is available. There's a risk of developing specialized accelerators that become less necessary due to software solutions implementing memory management or execution scheduling improvements. That being said, major technology companies are improving their forecasting strategies to minimize this risk.

Integration complexity is another issue. Hardware-

specific features often require modifications to open source software, creating friction between hardware capabilities and software frameworks. Projects using different libraries that don't align with specialized hardware create engineering challenges.

The most successful approaches I've seen embrace adaptability rather than perfect prediction. Hardware

that provides flexible acceleration primitives rather than overly specialized functions tends to remain relevant longer as open source workloads continue their rapid evolution. The key is developing hardware that accommodates the experimental nature of open source AI while still delivering meaningful performance improvements for established patterns.

4. How do open source tools help identify and resolve performance bottlenecks in AI hardware systems?

Open source tools provide practical methods for identifying and resolving performance bottlenecks in AI hardware by offering customizable platforms that allow developers to analyze hardware performance and optimize workloads.

Take Triton and Azure MAIA 100 for example. Developed by OpenAI, Triton is an open-source programming language and compiler designed to simplify GPU optimization for neural network operations, providing an accessible alternative to CUDA for high-performance GPU kernel development in AI workloads.

The integration of Triton with Microsoft's Azure MAIA 100 AI chip, via the MAIA SDK, demonstrates how open-source optimization tools can complement proprietary hardware. This combination offers developers a flexible and efficient environment for AI model

development while leveraging the performance benefits of specialized accelerators.

The open source ecosystem includes various performance analysis tools. PyTorch Profiler, TensorBoard, and NVIDIA Nsight provide detailed information about hardware utilization, memory issues, and computational efficiency.

In my experience working with enterprise AI deployments, these tools democratize hardware optimization. Teams without massive resources can now identify exactly where their AI systems struggle, whether it's memory bandwidth constraints, underutilized compute resources, or inefficient data movement patterns.

5. Can you share an instance where community contributions significantly accelerated an AI hardware breakthrough?

While working on Intel's oneAPI and OpenVINO, I observed how community contributions expanded hardware compatibility. OpenVINO is an open source software suite that optimizes deep learning models across various hardware platforms.

Developers created new device plugins that extended OpenVINO to work with additional hardware, including ARM architectures and specialized accelerators. During Google Summer of Code projects, participants specifically focused on improving Generative AI performance on ARM devices—addressing latency and memory issues while expanding the hardware ecosystem supported by OpenVINO.

In another instance, Codeplay released an open-source layer enabling Intel's oneAPI and SYCL/DPC++ to operate on NVIDIA GPUs via CUDA.

This pattern of community-driven development becomes more important as AI hardware diversifies. Significant advances increasingly come from developers working across organizational boundaries, sharing optimizations and finding solutions to hardware constraints.

6. In what ways does open source simplify the design and development process for next-generation AI accelerators?

Open source reduces uncertainty in hardware design. Standardized libraries, frameworks, and established usage patterns allow more accurate predictions about workload characteristics. This shared foundation is

valuable when developing specialized hardware that needs to work effectively across different AI applications.

Open benchmarks and reference implementations allow early testing against actual workloads. Teams can identify performance issues and optimization opportunities earlier in development—before finalizing silicon designs, when changes are less costly.

Analysis of open source usage reveals which operations are most common in popular frameworks. A study of GitHub repositories indicated TensorFlow is used primarily for image classification (32%), NLP (28%), and recommender systems (18%). While these specific percentages lack peer-reviewed validation, the directional insights are a data point for aligning accelerator

designs with prevalent usage patterns. For production systems, this data can directly inform hardware architecture—leading to designs with 4–8x wider vector units optimized for these workloads' matrix operations. These insights drive decisions from memory hierarchy design to the balance between specialized and general-purpose processing elements.

This results in a more efficient development process that produces hardware better aligned with actual requirements. Teams can build on established technical foundations rather than designing in isolation.

7. Which emerging open source trends do you believe will most transform AI hardware innovation in the near future?

Several technical trends in open source are changing AI hardware development. There's a notable shift toward optimizing inference rather than training as the primary focus for hardware development. As models become more accessible through open source, competitive advantage comes from efficient deployment rather than proprietary training capabilities. Organizations find that optimizing production performance often provides more practical value than incremental training improvements.

Software techniques in the open source ecosystem are changing hardware requirements. Quantization, pruning, and model distillation improve performance without requiring proportional improvements in hardware specifications. Open compiler stacks like Triton and OpenX-LA map AI workloads to different hardware targets, increasing flexibility and vendor choice. The field is moving toward compiler autotuning services that find

optimal configurations across hardware types, and repositories of optimized models for different hardware.

The community's exploration of sparse computation and attention mechanisms will influence the next generation of hardware specialization. As these approaches become more common in open source models, we'll likely see accelerators designed specifically for operations like sparse matrix multiplication and key-value lookups, moving beyond the dense matrix operations that characterized previous hardware.

These developments point toward a future where hardware and software co-evolve rapidly, with open source bridging research breakthroughs and production-ready solutions. The most successful innovations will emerge from teams that deeply understand both domains.

8. How do you navigate security and intellectual property challenges within the open source AI ecosystem?

This requires constant balancing of competing priorities. On the IP front, I've observed the need for more nuanced licensing models that protect innovation while enabling broader adoption. The data licensing challenges we're seeing with models trained on scraped content highlight how our intellectual property frameworks haven't kept pace with technological capabilities. I favor content attribution practices, but I realize they are hard to implement.

Security concerns are equally complex. While not specific to open source, what worries me most is the potential for compounding errors when models are chained together in workflows, especially as we sprint towards deploying agentic AI. Small inaccuracies can cascade through systems in ways that become increasingly difficult to diagnose or correct, in turn

breaking trust. My approach focuses on transparency in model development and comprehensive testing frameworks that can identify potential vulnerabilities before deployment. One of the biggest lessons from my generative AI product launch at Amazon was the importance of red teaming and ensuring a fully secure and safe system before opening the flood gates in production. You have to understand how what you are building works, down to its core, to understand the potential attack vectors.

The current paradigm is unsustainable—we need new approaches that recognize the unique characteristics of AI systems and support innovation while providing appropriate safeguards throughout the process, from data collection to user interaction.

9. How do you balance the benefits of proprietary solutions with the innovation driven by open source frameworks?

I view open source and proprietary approaches as complementary forces rather than competitors. Each serves a distinct purpose in advancing AI technology. Open source frameworks accelerate development through collective effort. Collaboration among thousands of developers produces algorithms and libraries more rapidly than any single organization could achieve. This prevents duplicated work and creates common foundations that benefit the entire field.

Proprietary hardware delivers the specialized integration needed for demanding workloads. When we design custom silicon, we gain precise control over perfor-

mance, efficiency, and security—unlocking capabilities that justify the substantial investment costs.

The key to success is maintaining clear boundaries. Core computational primitives and codecs benefit from open standards, while proprietary solutions should focus on areas of genuine differentiation. I've found this balanced approach accelerates testing cycles while enabling both ecosystem growth and competitive advantage.

10. What advice would you offer to startups and developers eager to leverage open source AI in hardware applications?

Focus on understanding actual workloads throughout hardware development. The common error I see is developing hardware based on theoretical expectations rather than observed behavior patterns or real technical problems. Startups should maintain direct connections with users and collect continuous feedback during development. In this rapidly changing field, it's difficult to predict which workloads will be relevant when hardware is ready for production, but under-

standing trends is essential to prevent your product being rendered obsolete at launch.

Incorporate insights from the open source community into hardware development. This requires more than monitoring code repositories—it means understanding the technical constraints driving community development decisions. These indicators often signal future trends that should inform architectural choices.

11. What excites you most about the future of open source AI, and how do you see it evolving in the hardware domain?

The practical decentralization of AI is most interesting—shifting from central services to capable local models without proprietary requirements. This resembles the transition from mainframe computing to personal computers in the 1980s.

We're witnessing an unprecedented acceleration in open source model capabilities. Features that were exclusive to frontier AI labs just months ago are now accessible to individual developers. This broader access is driving development of specialized hardware focused on efficient inference rather than large-scale training infrastructure.

The technical relationship forming between hardware and software is particularly noteworthy. As models become more accessible, we're seeing diversification in implementation scenarios—from edge devices to spe-

cialized accelerators—each optimized for specific functions. This creates market opportunities for specialized processing hardware that wouldn't have existed in a centralized model.

This hardware evolution driven by broader access will change the industry structure in the coming 18–24 months. Organizations that understand and adapt to this shift will define the next phase of AI infrastructure.

ACHIEVING CRITICAL RELIABILITY IN INSTRUCTION- FOLLOWING WITH LLMS

How to Achieve AI Customer Service
That's 100% Reliable (Parlant)

Written By :
Yam Marcovitz

Introduction

Ensuring reliable instruction-following in LLMs remains a critical challenge. This is particularly important in customer-facing applications, where mistakes can be costly. Traditional prompt engineering techniques fail to deliver consistent results. A more structured and managed approach is necessary to improve adherence to business rules while maintaining flexibility.

This article explores key innovations, including granular atomic guidelines, dynamic evaluation and filtering of instructions, and Attentive Reasoning Queries (ARQs), while acknowledging implementation limitations and trade-offs.

The Challenge: Inconsistent AI Performance in Customer Service

LLMs are already providing tangible business value when used as assistants to human representatives in customer service scenarios. However, their reliability as autonomous customer-facing agents remains a challenge.

Traditional approaches to developing conversational

LLM applications often fail in real-world use cases. The two most common approaches are:

1. Iterative prompt engineering, which leads to inconsistent, unpredictable behavior.

2. Flowchart-based processing, which sacrifices the real magic of LLM-powered interactions: dynamic, free-flowing, human-like interactions.

In high-stakes customer-facing applications, such as banking, even minor errors can have serious consequences. For instance, an incorrectly executed API call (like transferring money) can lead to lawsuits and reputational damage. Conversely, mechanical interactions that lack naturalness and rapport hurt customer trust and engagement, limiting containment rates (cases resolved without human intervention).

For LLMs to reach their full potential as dynamic, autonomous agents in real-world cases, we must make them follow business-specific instructions consistently and at scale, while maintaining the flexibility of natural, free-flowing interactions.

How to Create a Reliable, Autonomous Customer Service Agent with LLMs

To address these gaps in LLMs and current approaches, and achieve a level of reliability and control that works well in real-world cases, we must question the approaches that failed.

One of the first questions I had when I started working on Parlant (an open-source framework for customer-facing AI agents) was, "If an AI agent is found to mishandle a particular customer scenario, what would be the optimal process for fixing it?" Adding additional demands to an already-lengthy prompt, like "Here's how you should approach scenario X..." would quickly become complicated to manage, and the results weren't consistent anyhow. Besides that, adding those instructions unconditionally posed an alignment risk since LLMs are inherently biased by their input. It was therefore important that instructions for scenario X did not leak into other scenarios which potentially required a different approach.

We thus realized that instructions needed to apply only in their intended context. This made sense because, in real-life, when we catch unsatisfactory behavior in real-time in a customer-service interaction, we usually know how to correct it: We're able to specify both what needs to improve as well as the context in which our feedback should apply. For example, "Be concise and to the point when discussing premium-plan benefits," but "Be willing to explain our offering at length when comparing it to other solutions."

In addition to this contextualization of instructions, in training a highly capable agent that can handle many use cases, we'd clearly need to tweak many instructions over time as we shaped our agent's behavior to business needs and preferences. We needed a systematic approach.

Stepping back and rethinking, from first principles, our ideal expectations from modern AI-based interactions and how to develop them, this is what we understood about how such interactions should feel to customers:

1. Empathetic and coherent: Customers should feel in good hands when using AI.

2. Fluid, like Instant Messaging (IM):

Allowing customers to switch topics back and forth, express themselves using multiple messages, and ask about multiple topics at a time.

3. Personalized: You should feel that the AI agent knows it's speaking to you and understands your context.

From a developer perspective, we also realized that:

1. Crafting the right conversational UX is an evolutionary process. We should be able to confidently modify agent behavior in different contexts, quickly and easily, without worrying about breaking existing behavior.

2. Instructions should be respected consistently. This is hard to do with LLMs, which are inherently unpredictable creatures. An innovative solution was required.

3. Agent decisions should be transparent. The spectrum of possible issues related to natural language

and behavior is too wide. Resolving issues in instruction-following without clear indications of how an agent interpreted our instructions in a given scenario would be highly impractical in production environments with deadlines.

Implementing Parlant's Design Goals

Our main challenge was how to control and adjust an AI agent's behavior while ensuring that instructions are not spoken in vain—that the AI agent implements them accurately and consistently. This led to a strategic design decision: granular, atomic guidelines.

1. Granular Atomic Guidelines

Complex prompts often overwhelm LLMs, leading to incomplete or inconsistent outputs with respect to the instructions they specify. We solved this in Parlant by dropping broad prompts for self-contained, atomic guidelines. Each guideline consists of:

Condition: A natural-language query that determines when the instruction should apply (e.g., *"The customer inquires about a refund..."*)

Action: The specific instruction the LLM should follow (e.g., *"Confirm order details and offer an overview of the refund process."*)

By segmenting instructions into manageable units and systematically focusing their attention on each one at a time, we could get the LLM to evaluate and enforce them with higher accuracy.

2. Filtering and Supervision Mechanism

LLMs are highly influenced by the content of their prompts, even if parts of the prompt are not directly relevant to the conversation at hand.

Instead of presenting all guidelines at once, we made Parlant dynamically match and apply only the relevant set of instructions at each step of the conversation. This real-time matching can then be leveraged for:

Reduced cognitive overload for the LLM: We'd avoid prompt leaks and increase the model's focus on the right instructions, leading to higher consistency.

Supervision: We added a mechanism to highlight each guideline's impact and enforce its application, increasing conformance across the board.

Explainability: Every evaluation and decision generated by the system includes a rationale detailing how guidelines were interpreted and the reasoning behind skipping or activating them at each point in the conversation.

Continuous improvement: By monitoring guideline effectiveness and agent interpretation, developers could easily refine their AI's behavior over time. Because guidelines are atomic and supervised, you could easily make structured changes without breaking fragile prompts.

3. Attentive Reasoning Queries (ARQs)

While "Chain of Thought" (CoT) prompting improves reasoning, it remains limited in its ability to maintain consistent, context-sensitive responses over time. Parlant introduces Attentive Reasoning Queries (ARQs)—a technique we've devised to ensure that multi-step reasoning stays effective, accurate, and predictable, even across thousands of runs. You can find our research paper on ARQs vs. CoT on parlant.io and arxiv.org.

ARQs work by directing the LLM's attention back to high-priority instructions at key points in the response generation process, getting the LLM to attend to those instructions and reason about them right before it needs to apply them. We found that "localizing" the reasoning around the part of the response where a specific instruction needs to be applied provided significantly greater accuracy and consistency than a preliminary, nonspecific reasoning process like CoT.

Acknowledging Limitations

While these innovations improve instruction-following, there are challenges to consider:

Computational overhead: Implementing filtering and reasoning mechanisms increases processing time. However, with hardware and LLMs improving by the day, we saw this as a possibly controversial, yet strategic

design choice.

Alternative approaches: In some low-risk applications, such as assistive AI co-pilots, simpler methods like prompt-tuning or workflow-based approaches often suffice.

Why Consistency Is Crucial for Enterprise-Grade Conversational AI

In regulated industries like finance, healthcare, and legal services, even 99% accuracy poses significant risk. A bank handling millions of monthly conversations cannot afford thousands of potentially critical errors. Beyond accuracy, AI systems must be constrained such that errors, even when they occur, remain within strict, acceptable bounds.

In response to the demand for greater accuracy in such applications, AI solution vendors often argue that humans also make mistakes. While this is true, the difference is that, with human employees, correcting them is usually straightforward. You can ask them why

they handled a situation the way they did. You can provide direct feedback and monitor their results. But relying on “best-effort” prompt-engineering, while being blind to why an AI agent even made some decision in the first place, is an approach that simply doesn’t scale beyond basic demos.

This is why a structured feedback mechanism is so important. It allows you to pinpoint what changes need to be made, and how to make them while keeping existing functionality intact. It’s this realization that put us on the right track with Parlant early on.

Handling Millions of Customer Interactions with Autonomous AI Agents

For enterprises to deploy AI at scale, consistency and transparency are non-negotiable. A financial chatbot providing unauthorized advice, a healthcare assistant misguiding patients, or an e-commerce agent misrepresenting products can all have severe consequences. Parlant redefines AI alignment by enabling:

Enhanced operational efficiency: Reducing human intervention while ensuring high-quality AI interactions.

Consistent brand alignment: Maintaining coherence with business values.

Regulatory compliance: Adhering to industry standards and legal requirements.

This methodology represents a shift in how AI alignment is approached in the first place. Using modular guidelines with intelligent filtering instead of long, complex prompts; adding explicit supervision and validation mechanisms to ensure things go as planned—these innovations mark a new standard for achieving reliability with LLMs. As AI-driven automation continues to expand in adoption, ensuring consistent instruction-following will become an accepted necessity, not an innovative luxury.

If your company is looking to deploy robust AI-powered customer service or any other customer-facing application, you should look into Parlant, an agent framework for controlled, explainable, and enterprise-ready AI interactions.

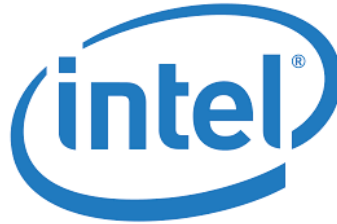
Yam Marcovitz

Author

An experienced software builder with extensive experience in mission-critical software and system architecture, Yam understands what it takes to create reliable, production-ready software. This background informs his distinctive approach to the development of predictable and aligned AI systems



FROM VISION TO GENERATIVE AI:



An Inside Look at Intel OpenVINO with Ravi Panchumarthy
and Helena Kloosterman (*AI Frameworks Engineers at Intel*)

Bios

Ravi Panchumarthy

Ravi is an AI Frameworks Engineer at Intel, where he collaborates closely with Intel's customers and partners, providing technical guidance and support to build, optimize, and deploy AI solutions on Intel architectures. He specializes in integrating tools like OpenVINO to accelerate AI inference, delivering significant performance and efficiency gains across diverse sectors. He holds a PhD in computer science and engineering from the University of South Florida with a dissertation focused on developing novel non-boolean computing techniques for computer vision applications using nanomagnetic field-based computing. He holds two patents and several peer-reviewed publications in journals and conferences. In his free time, he enjoys traveling and hiking.

Helena Kloosterman

As an AI Frameworks Engineer at Intel, Helena enables organizations to unlock the potential of AI by leveraging OpenVINO, Intel's AI inference runtime. She focuses on making AI accessible by enhancing developer experiences and integrating OpenVINO with popular ecosystems like Hugging Face. With a background in mathematics, Helena is passionate about empowering developers to easily deploy high-performance AI models on Intel hardware (CPUs, GPUs and NPU). Based in the Netherlands, she works to bridge the gap between complex AI technologies and practical applications.

1. Evolution and Vision:

How has OpenVINO evolved since its inception, and what core changes have driven its transformation in the open source AI landscape?

The Intel Distribution of OpenVINO toolkit started as a toolkit focused on optimizing computer vision models for Intel CPUs, but it has since evolved into a comprehensive AI inference framework supporting a wide array of deep learning tasks, including generative AI and LLMs. The core transformation started when this project was made open-source in 2019, opening community collaboration and significantly expanding our user base. Later, supporting a wide range of hardware (GPUs, NPUs, FPGAs), integrating frameworks like PyTorch, TensorFlow, ONNX, etc and the most recent evolution

has been our investment in LLM and generative AI support. Through key projects like NNCF (Neural Network Compression Framework), OpenVINO Model Server, along with strategic integrations like Hugging Face Optimum Intel, torch.compile, LangChain, Ultralytics, Keras 3, vLLM, etc., OpenVINO has continually expanded its capabilities and reach. Throughout this evolution, our vision has remained consistent - to democratize AI deployment by providing optimized inference capabilities that work seamlessly across hardware platforms.

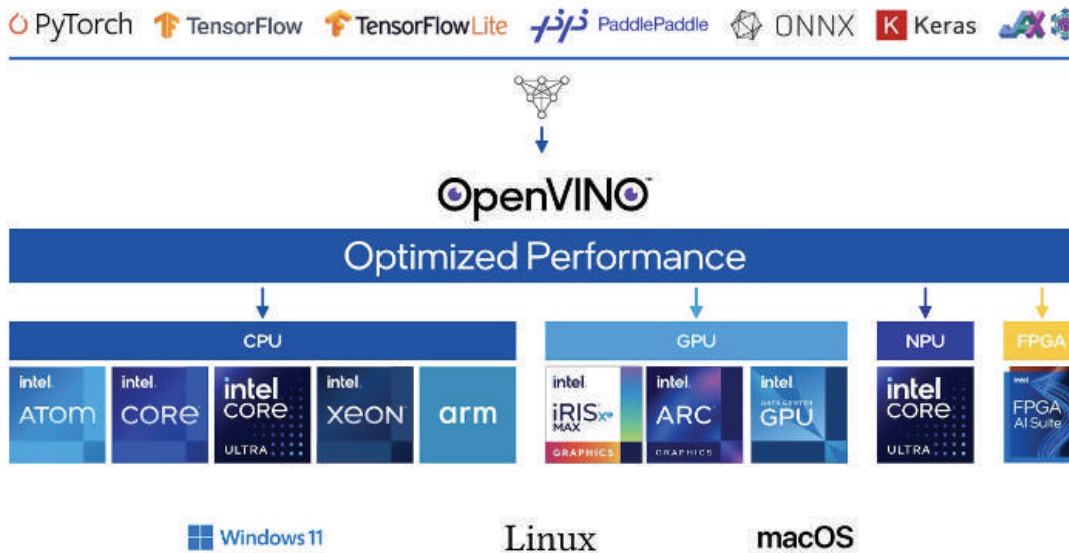


Image Source: <https://docs.openvino.ai/2025/index.html>

2. Heterogeneous Deployment:

OpenVINO supports inference across CPUs, GPUs, NPUs, and even FPGAs. Can you share insights on the design philosophy that enables such seamless deployment on heterogeneous hardware?

OpenVINO's design philosophy for heterogeneous deployment are based on abstraction and modularity. By decoupling model representation from hardware specifics via plugins (e.g., GPU plugin), OpenVINO provides a unified API allowing developers to write inference code once and deploy it across different hardware platforms without rewriting it for each platform. OpenVINO's hardware-agnostic intermediate representation (IR) serves as a common language that all supported hardware can understand. Underneath, the hardware-specific plugins translate the IR into optimized code for each target device, whether CPU, GPU, NPU, or FPGA. This philosophy ensures seamless scaling from legacy systems to cutting-edge accelerators without code changes.

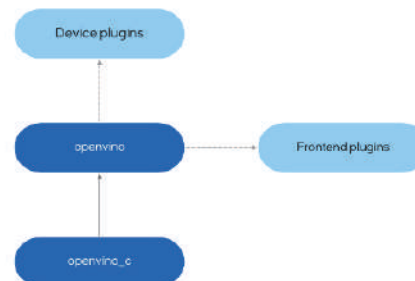


Image Source: <https://docs.openvino.ai/2025/openvino-workflow/deployment-locally.html>

3. Performance Optimization:

What are some of the key optimization techniques in OpenVINO that help reduce latency and model footprint while maintaining accuracy?

OpenVINO uses several sophisticated optimization techniques to enhance inference performance. Graph Optimization, Quantization, Weight Compression, and Hardware-Specific Optimizations work together to minimize latency and memory footprint, ensuring that even complex models run smoothly on resource-constrained hardware. The most basic optimizations include fusing of operations and dropping of nodes that are not needed for inference. A further optimization is quantization. For LLMs we use weight compression,

where only weights get quantizations, but activations remain in their original precision. For most models we see no accuracy impact when reducing precision from FP32 to INT8, and even INT4 works pretty well. For other models we support full quantization through NNCF. There are also optimizations on a hardware level that are leveraged by OpenVINO such as AVX512, AMX. OpenVINO automatically enables 16 bit floating point precision if the hardware supports it, for example Intel® Xeon® Processors and Intel® Arc™ Graphics

4. Generative AI and LLMs:

Recent updates have added support for large language models and generative AI. How do these enhancements extend the toolkit's capabilities and impact real-world applications?

The largest enhancement is in developer experience. In traditional AI models, inference involves providing a model input and receiving output in return. However, with generative AI, the process is more complex. The output is generated in multiple steps, and there are many settings to optimize these steps. For example, you can set the temperature to influence the kind of

output the model generates. Writing code to do this is not straightforward, and OpenVINO GenAI hides this complexity from the developer, so they can focus on developing their applications, without having to know how to implement advanced generative AI inferencing strategies themselves, not only for Python, but also for C++ applications.

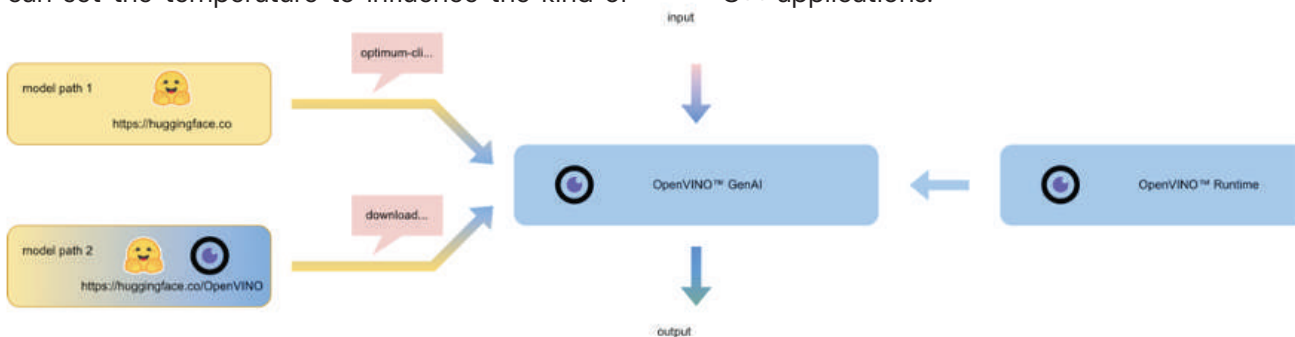


Image Source: <https://docs.openvino.ai/2025/openvino-workflow-generative/inference-with-genai.html>

5. Open Source Commitment:

Balancing cutting-edge performance with open source accessibility is no small feat. How does Intel ensure that OpenVINO remains both highly performant and community friendly?

Yes, indeed it is a challenge, but at Intel®, a community-first mindset paired with technical rigor makes it a possibility. We actively listen to developers through forums, GitHub discussions, and work closely with partners and customers, incorporating their feedback into our roadmap. This ensures OpenVINO addresses real user needs. Our detailed documentation and contribution guidelines along with mentorship programs like

Google Summer of Code (GSoC) and Good First Issues in GitHub encourage developers to contribute and improve the toolkit. OpenVINO invested in a robust continuous integration system that tests performance across diverse hardware configurations, ensuring optimizations through contributions deliver expected performance while maintaining the openness.

6. Edge and Cloud Synergy:

As AI moves increasingly to the edge, how does OpenVINO support deployment on resource-constrained devices, and what challenges do you face when optimizing for both edge and cloud environments?

OpenVINO is lightweight and flexible. You can for example compile a custom version of OpenVINO that only contains support for a particular model and the device that you are deploying on. Model Quantization and Compression will further minimize memory and power usage, critical for edge devices. OpenVINO heterogeneous hardware support maximizes performance on Intel edge hardware by leveraging integrated GPUs and NPUs. The challenge with optimizing between edge and cloud is often more at an application level. On the client, most applications are single user; in the cloud, a multi-user server application is often expected. Edge devices have limited memory, power, and compute,

requiring smaller models and inference runtime, but in contrast, cloud setups prioritize scalability and throughput, demanding different strategies. To achieve consistent optimized performance across diverse platforms – from resource-constrained edge chips to powerful cloud servers, OpenVINO provides a unified API and runtime that abstracts the underlying hardware differences, optimizing for each platform's strengths. This allows developers to deploy applications seamlessly from edge to cloud without extensive code modifications.

7. Prototyping to Production:

OpenVINO is praised for its ability to accelerate development cycles. What strategies or tools does the toolkit offer to help developers quickly prototype and then seamlessly transition their applications to production?

- Write once build anywhere – no need to rewrite your code when you switch to different devices. Build on your laptop, and the same application works seamlessly in the cloud.
- An API for asynchronous inference for optimizing throughput. Async is often tricky to implement, and we abstract away the trickiness
- OpenVINO GenAI makes it very easy to develop prototype generative AI applications with only a few lines of code, in either C++ or Python.
- A big challenge for production level AI applications is the ecosystem. Python deployment can be challenging,

especially when you want to support a wide variety of systems. One challenge is that many Python applications have a lot of dependencies on other libraries; and this is a fast moving field where things often break. OpenVINO addresses this in two ways:

- *C++ and Python APIs from OpenVINO are similar, making it easy to convert a Python prototype into a C++ application.*
- *For Python, OpenVINO has very few dependencies. Numpy is the only required external dependency. This reduces the risk of deployment issues and security vulnerabilities during the application's lifecycle.*

8. Latest Release Highlights:

With the 2025.0 release, what new features or enhancements are you most excited about, and how do they further advance the performance or usability of OpenVINO?

- OpenVINO GenAI and LLM support was much improved. More features for text generation, faster audio models (Whisper), and this release added support for several popular visual language models, image generation models, multimodal, and reasoning models.
- NPU improvements are amazing. Inference speed is now faster, models load faster, and more features are supported on NPU (for example sample search for LLMs, GenAI API support of LLMs on NPUs). These improvements are thanks to improvements on OpenVINO level and NPU driver level.

ANIMESH SINGH

*Director of AI & ML
Platforms at*

LinkedIn

Animesh Singh is the Director of AI and ML Platforms at LinkedIn, where he is spearheading the development of the next-generation AI platform. This platform is designed to create foundational AI models that cater to the needs of over 1 billion members. Animesh is focused on building and scaling various AI technologies, including distributed training, machine learning pipelines, generative AI, LLMs, GNNs, and incremental and online learning

Before joining LinkedIn, Animesh was a distinguished engineer, CTO, and senior director at IBM Watson AI and Data Open Tech. He also served as the LFAI Trusted AI NA Chair and co-founded Kubeflow.

In Their Own Words

Animesh Singh's Journey

1. Career & Motivation: Could you share a bit about your journey into AI and open source, and what initially inspired you to lead initiatives like Liger-Kernel?

In 2011, I was at IBM when their Watson DeepQA computer made history by defeating two of Jeopardy's all-time champions. That was the beginning of my interest in AI.

After leading several initiatives around training and inferencing in IBM's Watson group, I joined LinkedIn where AI-powered initiatives are driving a significant amount of the daily engagements on our professional network. I have been leading LinkedIn's massive AI Infrastructure expansion, which has included increasing our GPU footprint by more than 100X in FLOPS and training models that are 180X bigger than ones we've managed just a few years ago.

Regarding OpenSource, I have been a firm believer since the early days of my career that the power of the OpenSource community is immense, and it's hard for proprietary software to have the same quality and functionality maintained over an extended period of

time. Additionally, the diverse set of people coming together, debating, and building software in open source creates something magical. We have seen this play out with Linux, Kubernetes, PyTorch, and now with AI models like Llama and others.

The next frontier of innovation is the AI infrastructure and accelerator space. A few years ago, I cofounded a project in OpenSource called KServe for serverless model inference, which included joining hands with companies like Bloomberg, Google, RedHat, and Nvidia. It's one of the leading platforms in OSS for Model Inferencing on Kubernetes.

Liger was the next natural journey in that space – going onto the GPU layer itself and optimizing models directly from inside GPU Kernels.

2. Project Genesis: What was the driving vision behind the creation of Liger-Kernel, and how do you see it fitting into the broader ecosystem of AI platforms and infrastructure?

Liger Kernels, based on Triton, originated from LinkedIn's internal needs when we were training some heavy foundation models and teams were competing for GPU horsepower. We were also up against some restrictive time-bound goals to complete the training for different projects. When considering how to address these needs, we decided our primary goal was to improve performance – achieving massive throughput gains, while using lesser memory – for both training and infer-

encing, which brought us to our Liger project. Once launched, we got a great reception for Liger in the open source community, catching the attention of industry veterans like Andrej Karpathy and Jeremy Howard.

Internally, Liger Kernel improved our 70B model training speed by 3X. We've also added optimized Post-Training kernels that deliver up to 80% memory savings for alignment and distillation tasks.

3. Community Collaboration: Open source thrives on community involvement. Could you discuss how community contributions have shaped the evolution of Liger-Kernel and what role collaboration plays in your strategic vision?

We partnered with the OSS community and have integrated with mainstream training frameworks, including Axolotl, LLaMa-Factory, SFTTrainer, Hugging Face Trainer, SWIFT. We've also supported distributed training frameworks such as PyTorch FSDP and Microsoft DeepSpeed.

We've also seen dynamic adoption from the industry itself, with many companies integrating Liger Kernels organically, like Anyscale and Glows.ai. Within the first few months, we have seen 1M+ downloads of these Kernels, with many contributors and pull requests coming from the industry.

Expanding on this, with our open source strategy we aim to:

- Create a leading open-source Kernels library ecosystem for machine learning
- Leverage the power of open source to scale the kernels to handle a permutation of X Kernels, Y Models, with M frameworks, and H/W accelerators
- Extend to non LLM use cases, by collaboration with OSS. Additionally invest in CUDA, Cutlass, Triton, PTL based Kernels modifications

Our goal is to make Liger Kernel the most efficient ML Kernels at runtime for both training and post-training use cases.

4. Balancing Priorities: How do you navigate the balance between maintaining robust, enterprise-grade infrastructure and staying true to the open source ethos?

This balance is something we care about a lot. Our primary job is to deliver the best AI infrastructure and platform for LinkedIn, and as far as possible build on the latest and greatest open source technologies. Picking up projects depends on different dimensions. Sometimes it's performance at scale which dominates the decision making, and on other dimensions it's the AI developer's intuitiveness.

The culture in the team is set to be a net contributor, not just a consumer of OpenSource. That's how you build better products, and that's how OpenSource gets better. We are always finding opportunities to contribute back any internal extensions to OpenSource we consume (Horovod, Tensorflow, PyTorch, Flyte, etc.), while ensuring we are remaining true to our primary job of creating the best AI Infrastructure for LinkedIn.

5. Competitive Landscape: In a landscape crowded with proprietary AI solutions, what sets LinkedIn's approach to AI infrastructure apart from other tech giants?

We have ambitious goals both on our AI modeling strategy and AI infrastructure buildout, but balancing it with pragmatic spending on AI infrastructure keeps the culture of innovation active. Scarcity is the mother of invention, so ensuring we have enough spend to contribute towards business goals, but having a tighter control around spending ensures we do get to innovate and create projects like Liger and others.

Additionally, LinkedIn is very vigilant about duplication. We don't have multiple pockets of AI infrastructure within LinkedIn, but instead we have centralized on a

company wide GPU fleet, a central training infrastructure, and consolidating our AI serving in a central place as well. This sets a strong standardization across the company, and makes it easier to monitor and control capacity decisions.

And definitely building on standardized open source software which is proven in industry at scale, helps us hire the right talent, which is coming with pre existing skills in the stack.

6. Future of AI Infrastructure: As the demands on AI platforms evolve, where do you see the greatest opportunities—and challenges—in infrastructure innovation over the next 5 to 10 years?

With the pace at which AI is growing, it is hard to make very long-term projections, but let's take a 5 year lens. There are a few things I can see:

Purpose-fit AI accelerators will become a norm: the fungibility argument (same GPUs across different workloads like training and inferencing), and different model types will play out in the next couple of years. Beyond that, the cost of fungibility will become too high for most companies, and purpose-fit AI accelerators for training and inferencing, use cases like heavy encoder/-decoder instances, and model architectures (e.g. Transformer/LLMs), will be seriously invested in because of the ballooning cost.

Power demands will drive innovation: Even if companies have deep coffers to buy the most expensive AI accelerators, surging power demands, and the pace at which new AI datacenters will need to be built will spur innovation in the space to look beyond the current power hungry AI workloads, which cannot be sustained at this pace.

Post training and inferencing on GPUs across Data-centers: The inference landscape across the industry is dominated by CPUs. Current model architectures,

and upcoming ones cannot be served through them. Additionally, post training techniques (e.g. distillation, fine tuning, alignment, pruning) have created another class of workloads which are GPU hungry, so there will be new spend on AI infrastructure in these areas.

Agentic architectures mean more reasoning capabilities pushed down in the models to pick the right tools and take the right actions: So inference-time reasoning cost will balloon, and a huge class of Inference workloads will need to relax latency requirements, to get more deterministic, well-reasoned outputs.

Accelerator resident storage for AI workloads is a huge opportunity: More and more workloads are demanding storage as close as possible to the GPU compute to ensure we are not paying the network and latency cost of bringing data to GPUs. And the capacity needed for GPU resident storage (network/disk) is increasing. Both in AI accelerator design, and at the software layer, this would have been solved comprehensively.

FROM GENAI DEMOS TO PRODUCTION:

Why Structured Workflows are Essential

BY ALEX STRICK VAN LINSCHOTEN

At technology conferences worldwide and on social media, generative AI applications demonstrate impressive capabilities: composing marketing emails, creating data visualizations, or writing functioning code. Yet behind these polished demonstrations lies a stark reality. What works in controlled environments often fails when confronted with the demands of production systems.

Industry surveys reveal the scale of this challenge: *68% of organizations have moved 30% or fewer of their generative AI experiments into production*, while only 53% of AI projects overall progress from prototype to production – with a mere 10% achieving measurable ROI (*Walloo*). Why does this gap persist? The controlled environment of a demonstration bears little resemblance to the unpredictable demands of real-world deployment.

Most current GenAI applications rely on what some have called 'vibes-based' assessments rather than rigorous validation. A developer reviews the output, determines it looks reasonable, and the system advances to the next stage of development. While this approach might sometimes identify obvious flaws, it

Published by Marktechpost AI Media, Inc

fails to detect subtle inconsistencies that emerge only at scale or with edge-case inputs.

These reliability concerns become critical when AI systems influence business decisions with tangible consequences. *70% of organizations estimate needing at least 12 months to resolve challenges in achieving expected ROI from GenAI, highlighting the high stakes of production failures*. Each misstep carries measurable costs: an incorrect product recommendation affects not just immediate sales but customer retention; an inaccurate financial summary might lead to misallocation of resources; a flawed legal interpretation could create significant liability exposure.

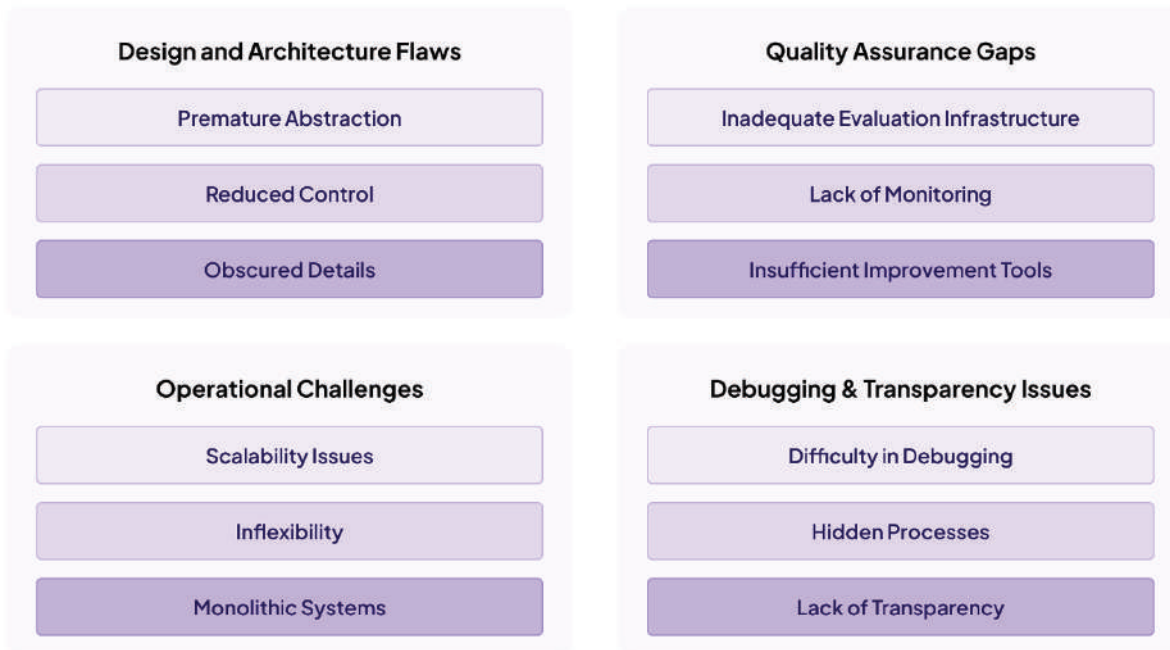
The transition from promising demonstrations to dependable production systems requires more than incremental improvements. It demands a fundamental shift in how we architect and evaluate GenAI applications. Structured workflows and systematic evaluation offer a methodical path forward—one that transforms unpredictable prototypes into systems worthy of trust with consequential decisions.

The Limitations Of Monolithic GenAI Applications

Most first-generation GenAI applications employ a deceptively simple architecture: user input enters the system, a language model processes it with some contextual information, and the system produces a

response. This end-to-end approach, while straightforward to implement, introduces significant limitations when deployed beyond controlled environments.

The Limitations of Monolithic GenAI Applications



zenml.io

The most pressing challenge involves identifying the source of errors. When a monolithic system produces incorrect, biased, or nonsensical output, determining the cause becomes an exercise in speculation. Did the retrieval mechanism provide irrelevant context? Was the prompt construction flawed? Does the base model lack necessary capabilities? Without visibility into these components, improvement efforts resemble guesswork rather than engineering. [Choco](#), a food distribution platform, discovered this when their single "catch-all" prompt worked in a hackathon but proved "not scalable or maintainable" in production.

Language models introduce another complication through their probabilistic nature. Even with identical inputs, these models may generate different outputs across successive executions. This variability creates a fundamental tension: creative applications benefit from diverse outputs, but business processes require consistency. The legal field saw [an infamous example](#) when an attorney unknowingly submitted hallucinated court cases from ChatGPT, leading to sanctions. The lack of internal measurement points further hampers improvement efforts. Without defined evaluation boundaries, teams struggle to isolate performance issues or quantify improvements.

Many current frameworks exacerbate these problems through premature abstraction. They encapsulate functionality behind interfaces that obscure necessary details, creating convenience at the expense of visibility and control. [A team at Prosus](#) found that off-the-shelf agent frameworks were fine for prototyping but too inflexible for production at scale.

These limitations become most apparent as organizations scale from prototype to production. Approaches that function adequately in limited tests falter when confronted with the volume, variety, and velocity of real-world data. Production deployment requires architectures that support not just initial development but ongoing operation, monitoring, and improvement—needs that monolithic systems struggle to satisfy. Successful teams have responded by breaking monolithic designs into modular pipelines, taming randomness with deterministic components, building comprehensive evaluation infrastructure, and favoring transparent architectures over premature abstractions.

Component-Driven GenAI: Breaking Down the Black Box

The transition to component-driven architecture represents more than a technical preference—it applies fundamental software engineering principles to generative AI development. By decomposing monolithic systems into discrete functional units, this approach transforms opaque black boxes into transparent, manageable workflows.

Component-based architecture divides complex systems into units with specific responsibilities, connected through well-defined interfaces. In GenAI applications, these components might include:

Data Retrieval Component: A vector database with embedding search that finds relevant documents or knowledge snippets based on user queries (e.g., Pinecone or Weaviate storing product information).

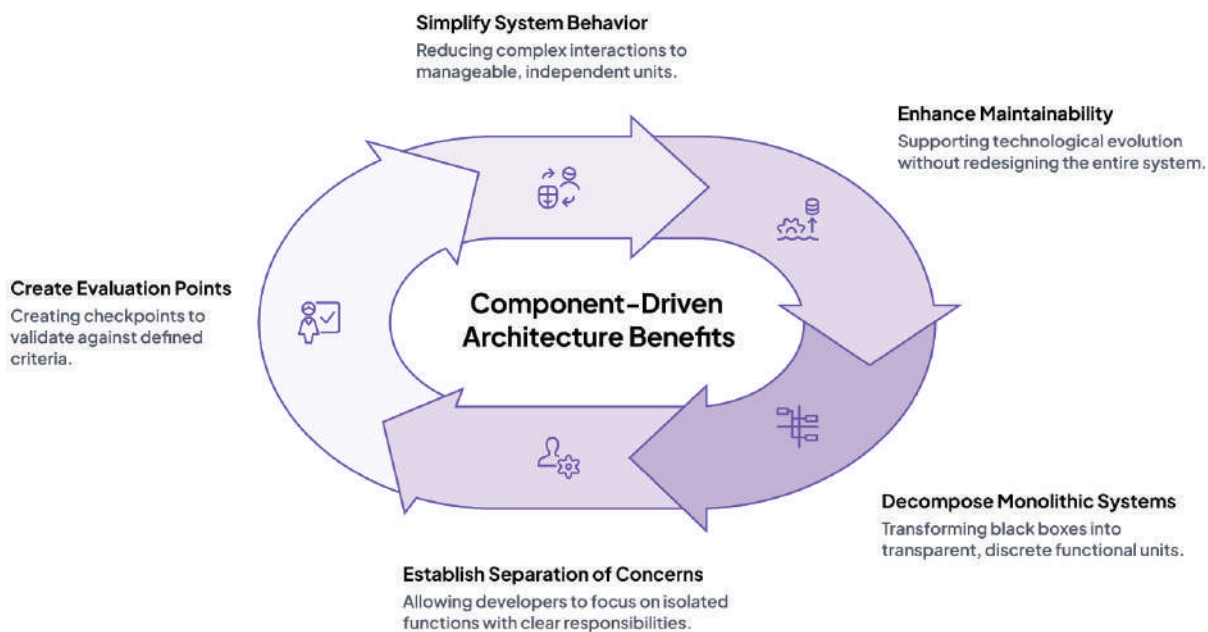
Prompt Construction Component: A template engine that formats retrieved information and user input into optimized prompts (e.g., a system that assembles query context).

Model Interaction Component: An API wrapper that handles communication with language models, manages retries, and standardizes input/output formats (e.g., a service that routes requests to Azure OpenAI or local Ollama endpoints).

Output Validation Component: A rule-based or LLM-based validator that checks outputs for accuracy, harmful content, or hallucinations (e.g., a fact-checking module that compares generated statements with retrieved knowledge).

Response Processing Component: A formatter that restructures raw model output into application-appropriate formats (e.g., a JSON parser that extracts structured data from text responses).

Each component addresses a specific function, creating natural boundaries for both execution and evaluation.



zenml.io

This decomposition yields several practical advantages that directly address the limitations of monolithic approaches. First, it establishes separation of concerns, allowing developers to focus on specific functionality without addressing the entire system simultaneously. Second, it creates discrete evaluation points where inputs and outputs can be validated against defined criteria. Third, it simplifies reasoning about system behavior by reducing complex interactions to manageable units that can be understood and modified independently.

Leading organizations have demonstrated these benefits in production. Uber's DragonCrawl, a system for automated mobile app testing, uses LLMs to execute tests with human-like intuition. While not explicitly described as component-driven in Uber's blog, its architecture effectively separates concerns into functional areas working together:

- A representation component that converts app UI screens into text for the model to process

Open Source AI

- A decision-making component using a fine-tuned MPNet model (110M parameters) that determines what actions to take based on context and goals
- An execution component that implements these decisions as interactions with the app

This structured approach achieved "99%+ stability" in November–December 2023 and successfully executed end-to-end trips in 85 out of 89 top cities without any city-specific tweaks. Most importantly, the system required no maintenance—when app changes occurred, DragonCrawl figured out how to navigate new flows on its own, unlike traditional tests that required hundreds of maintenance hours in 2023. The deliberate model selection process (evaluating multiple options against precision metrics) further demonstrates how systematic evaluation leads to reliable production systems.

The Component-Evaluation Pair: A Fundamental Pattern

At the heart of reliable GenAI systems lies a simple but powerful pattern: each component should have a corresponding evaluation mechanism that verifies its behavior. This component-evaluation pair creates a foundation for both initial validation and ongoing quality assurance.

This approach parallels unit testing in software engineering but extends beyond simple pass/fail validation. Component evaluations should verify basic functionality, identify performance boundaries, detect drift from expected behavior, and provide diagnostic information when issues arise. These evaluations serve as both quality gates during development and monitoring tools during operation.

Real-world implementations demonstrate this pattern's effectiveness. Aimpoint Digital built a travel itinerary generator with separate evaluations for its retrieval component (measuring relevance of fetched results) and generation component (using an LLM-as-judge to grade output quality). This allowed them to quickly identify whether issues stemmed from poor information retrieval or flawed generation.

Payment processing company Stripe implemented a component-evaluation pair for their customer support AI by tracking "match rate" – how often the LLM's suggested responses aligned with human agent final answers. This simple metric served as both quality gate and production monitor for their generation component.

Well-designed interfaces between components further enhance system maintainability. By establishing explicit contracts for data exchange, these interfaces create natural boundaries for testing and make components interchangeable. For example, a data retrieval component might specify that it accepts natural language queries and returns relevant document chunks with source metadata and relevance scores. This clear contract allows teams to swap between different retrieval implementations (keyword-based, embedding-based, or hybrid) without changing downstream components as long as the interface remains consistent.

The one-to-one relationship between components and evaluations enables targeted improvement when issues emerge. Rather than making broad changes to address vague performance concerns, teams can identify specific components that require attention. This precision reduces both development effort and the risk of unintended consequences from system-wide modifications.

The metrics from component evaluations form a comprehensive dashboard of system health. Engineers can monitor these indicators to identify performance degradation before it affects end users—a significant advantage over systems where problems become apparent only after they impact customers. This proactive approach supports maintenance activities and helps prevent production incidents.

When implemented systematically, component evaluations build confidence in system composition. If each component demonstrates acceptable performance against defined metrics, engineers can combine them with greater assurance that the resulting system will behave as expected. This compositional reliability becomes particularly important as systems grow in complexity.

Eval-First Development: Starting With Measurement

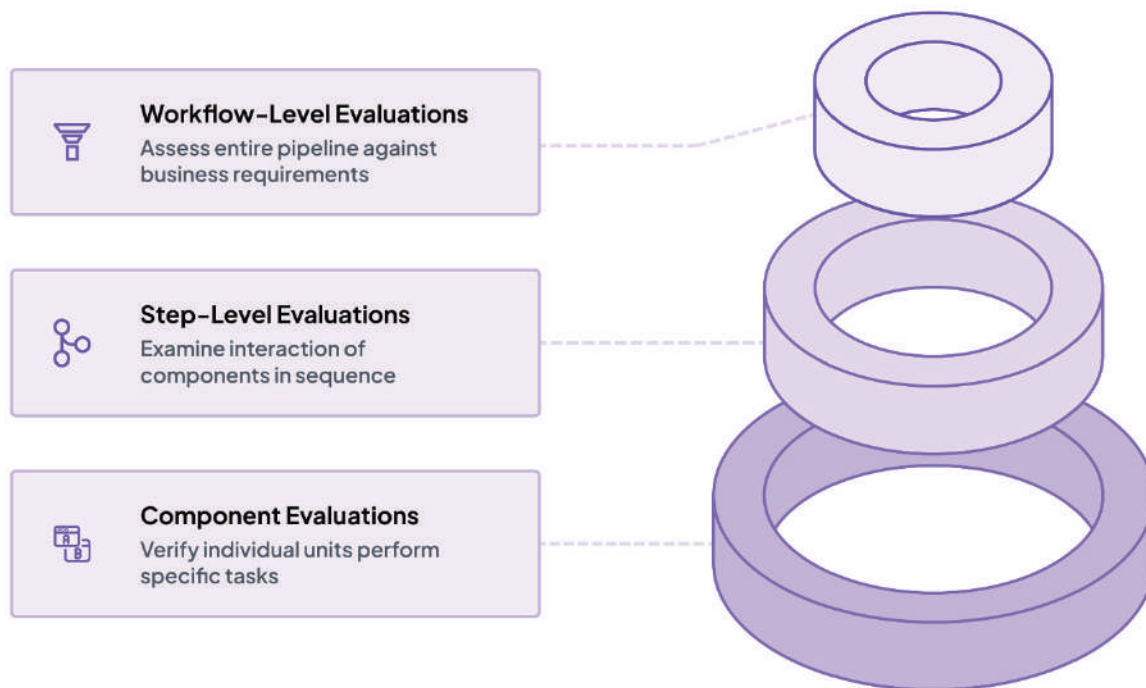
Conventional development processes often treat evaluation as an afterthought—something to be addressed after implementation is complete. Eval-first development inverts this sequence, establishing evaluation criteria before building components. This approach ensures that success metrics guide development from the outset rather than being retrofitted to match existing behavior.

The eval-first methodology creates a multi-tiered framework that operates at increasing levels of abstraction:

At the **component level**, evaluations function like unit tests in software development. These assessments verify that individual functional units perform their specific tasks correctly under various conditions. A retrieval component might be evaluated on the relevance of returned information across different query types, while a summarization component could be assessed on factual consistency between source text and generated summaries. These targeted evaluations provide immediate feedback during development and ongoing

Step-level evaluations examine how components interact in sequence, similar to integration testing in software development. These assessments verify that outputs from one component serve as appropriate inputs for subsequent components and that the combined functionality meets intermediate requirements. For example, step-level evaluation might confirm that a classification component correctly routes queries to appropriate retrieval components, which then provide relevant context to a generation component.

Workflow-level evaluations assess whether the entire pipeline satisfies business requirements. These system-level tests validate end-to-end performance against defined success criteria. For a customer support system, workflow evaluation might measure resolution rate, customer satisfaction, escalation frequency, and handling time. These metrics connect technical implementation to business outcomes, providing a framework for prioritizing improvements.



zenml.io

This layered approach offers significant advantages over end-to-end evaluation alone. First, it provides a comprehensive view of system performance, identifying issues at multiple levels of granularity. Second, it establishes traceability between business metrics and component behavior, connecting technical performance to business outcomes. Third, it supports incremental improvement by highlighting specific areas that require attention.

Organizations that implement eval-first development often discover requirements and constraints earlier in the development process. By defining how components will be evaluated before implementation begins, teams identify potential issues when they're least expensive to address. This proactive approach reduces both development costs and time-to-market for reliable systems.

Implementing Component-Based GenAI Workflows

Practical implementation of component-based GenAI workflows requires methodical decomposition of applications into steps that can be evaluated. This process begins with identifying core functions, then establishing clear responsibilities and interfaces for each component.

Effective breakdown balances granularity with practicality. Each component should have a single responsibility without creating excessive interaction overhead. Uber's GenAI Gateway demonstrates this through a unified service layer handling 60+ LLM use cases. By mirroring OpenAI's API interface, they created standardized endpoints that separate integration logic from application business logic.

Well-designed interfaces specify both data formats and semantic requirements. Microsoft's Azure Copilot uses RESTful APIs between components like its Knowledge Service (document chunking) and LLM processors. This enables independent development while ensuring components exchange properly structured, semantically valid data.

Components and evaluations should be versioned together for traceable evolution. Uber's approach allows centralized model upgrades – adding GPT-4V required only gateway adjustments rather than client changes. This containment of version impacts prevents system-wide disruptions.

Agentic components require constrained decision boundaries. Microsoft implements extensible plugins where each Azure service team builds domain-specific

"chat handlers." These predefined operations maintain control while enabling specialized functionality. Sophisticated fallback mechanisms become possible with component isolation. Uber's gateway implements automated model fallbacks, switching to internal models when external providers fail. This graceful degradation maintains service continuity without compromising entire workflows.

Microsoft's golden dataset approach provides versioned benchmarking against 500+ validated question/answer pairs. Component updates are tested against this dataset before deployment, creating a closed feedback loop between evaluation and improvement.

Key challenges persist:

- 1. Initial Investment** – Designing interfaces and evaluation frameworks requires upfront resources
- 2. Skill Gaps** – Teams need both software engineering and AI expertise
- 3. Coordination Overhead** – Inter-component communication adds complexity

Organizations must balance these against the benefits of maintainability and incremental improvement. As demonstrated by Uber's gateway – now handling authentication, PII redaction, and monitoring across all LLM interactions – centralized components with clear contracts enable scalability while maintaining governance.

Practical Considerations

Implementing component-based GenAI workflows involves several practical considerations that influence their effectiveness in production environments.

Parcha discovered users preferred reliable "agent-on-rails" designs over fully autonomous systems after their initial agent approach proved too unpredictable. RealChar implemented a deterministic event-driven pipeline for AI phone calls, achieving low latency through fixed processing cycles rather than free-form agent architectures.

The organizational implications of component-based architecture extend beyond technical considerations. PagerDuty formed a centralized LLM service team that enabled four new AI features in two months by standardizing infrastructure across product teams. This mirrors how companies established dedicated data platform teams during earlier tech waves.

Organizations with established machine learning infra-

structure have a significant advantage when implementing component-based GenAI systems. Many foundational MLOps capabilities transfer directly to LLMOps with minimal adaptation. For example, existing model registry systems can be extended to track LLM versions and their performance metrics. Data pipeline orchestrators that manage traditional ML workflows can be repurposed to coordinate GenAI component execution. Monitoring systems already watching for ML model drift can be adapted to detect LLM performance degradation.

Leading organizations have found that reusing these battle-tested MLOps components accelerates GenAI adoption while maintaining consistent governance and operational standards. Rather than building parallel infrastructure, enterprise companies have extended their ML platforms to accommodate the unique needs of LLMs, preserving the investment in tooling while adapting to new requirements.

Open Source AI

Resource allocation represents another practical consideration. Component-based architectures require investment in infrastructure for component orchestration, interface management, and comprehensive evaluation. These investments compete with feature development and other organizational priorities. Suc-

Building for the Future

Component-based, evaluated workflows provide a foundation for sustainable GenAI development that extends beyond current capabilities. This approach positions organizations to incorporate emerging technologies without wholesale system replacement.

The field of generative AI continues to evolve rapidly, with new model architectures, specialized models, and improved techniques emerging regularly. Component-based systems can integrate these advances incrementally, replacing individual components as better alternatives become available. This adaptability provides significant advantage in a rapidly evolving field, allowing organizations to benefit from technological progress without disruptive rebuilding.

The reliability advantage of evaluated components becomes increasingly important as GenAI applications address critical business functions. Organizations that implement systematic evaluation establish quantitative evidence of system performance, supporting both internal confidence and external trust. This evidence-based approach helps organizations navigate regulatory requirements, customer expectations, and internal governance. As regulatory scrutiny of AI systems increases, the ability to demonstrate systematic evaluation and quality assurance will become a competitive differentiator.

Component evaluation enables continuous, data-driven improvement by providing detailed performance

successful implementation requires executive support based on understanding the long-term benefits of maintainable, evaluable systems over short-term feature delivery.

insights. Rather than relying on broad assessments or anecdotal feedback, teams can analyze component-level metrics to identify specific improvement opportunities. This targeted approach supports efficient resource allocation, directing effort toward areas with measurable impact.

Organizations should assess their current GenAI implementations through the lens of componentization and systematic evaluation. This assessment might examine several questions: Are system responsibilities clearly divided into evaluable components? Do explicit interfaces exist between these components? Are evaluation metrics defined at component, step, and workflow levels? Does the architecture support incremental improvement?

The transition from impressive demonstrations to reliable production systems ultimately requires both technical architecture and organizational commitment. Component-based workflows with systematic evaluation provide the technical foundation, while organizational priorities determine whether this foundation supports sustainable development or merely adds complexity. Organizations that commit to this approach—investing in component design, interface definition, and comprehensive evaluation—position themselves to deliver not just impressive demonstrations but dependable systems worthy of trust with consequential decisions.

Alex Strick van Linschoten

Alex is a Machine Learning Engineer at ZenML. He previously worked as a historian and researcher but now focuses on LLM and GenAI initiatives for ZenML's MLOps and LLMOps platform. Alex holds a PhD in War Studies from King's College London and a BA in Arabic and Persian from SOAS. He's based in Delft (Netherlands).





ZenML

zenml.io

Because Production Shouldn't Be an Afterthought

Build LLM Workflows

Reduced complexity, increased transparency

Eval-First Development

bridge the 90% prototype-to-production gap

Full observability

with component-level evaluation metrics

Book your demo now!



Deep Dive: Exploring **BYRON HSU'S**

Member of Technical Staff at xAI



I am currently a member of the technical staff at xAI, working on ML systems. I am the creator of Liger Kernel and a committer to SGLang and Flyte. I enjoy exploring the intersection of machine learning and systems. "These are my own opinions and not the position of xAI."

1. Foundational Influences & Strategic

Open source AI has undergone a remarkable evolution—from the advent of transformer architectures to the emergence of distributed training paradigms. Could you reflect on the groundbreaking research and pivotal milestones that first captivated you, and share how these influences have shaped your strategic vision for fostering community-driven innovation and transparency?

The pivotal milestone papers for me to realize the importance of LLM systems are ZeRO and vLLM. ZeRO proposed an innovative algorithm to shard weights on

the fly while maintaining low communication cost in order to train big models. vLLM proposed paged attention to efficiently manage KV cache with optimal memory utilization in order to serve big models.

LLMs are powerful yet need lots of work to build strong LLM system to run efficiently, both for training and inference. ZeRO and vLLM are the best proofs of the win of open source. There are countless follow-up works of them and they enable numerous developers or researchers to innovate models more efficiently.

2. Liger-Kernel Genesis

What specific challenges in large-scale LLM training motivated the creation of Liger-Kernel, and how did you conceptualize its core architecture?

Liger-Kernel originated from the need at LinkedIn to train LLMs efficiently. We suffered from critical performance bottlenecks that hindered large-scale training on large datasets. We started by optimizing the distributed training level, adopting tools like DeepSpeed, FSDP, and Megatron. However, we realized the gains were insufficient, so we decided to go one level lower to the

kernel level. The goal of Liger Kernel is to provide a performant yet easy-to-use kernel library tailored for LLM training. We have done much of the heavy lifting in performance optimization underneath, while keeping the interface friendly and integrating nicely with popular frameworks like Hugging Face

3. Liger-Kernel Innovations

Liger-Kernel reportedly boosts throughput by 20% while reducing memory usage by 60%—could you elaborate on the key innovations, such as kernel fusion and chunking techniques, that make these improvements possible?

We mainly utilize kernel fusion, in-place operations, and chunking techniques to optimize performance. Kernel fusion minimizes data transfer time between GPU memory, which is effective for bandwidth-bound kernels like RMSNorm or RoPE. We use chunking to avoid large logits materialization for large-vocabulary models. Instead of generating output logits all at once, we

generate them chunk by chunk. This can effectively reduce memory usage by approximately 60% end-to-end. We also employ in-place operations to reuse memory sitting idle, which saves tensor creation time and reduces memory consumption.

4. Engineering Trade-offs

How do you navigate the balance between computational efficiency and maintaining precision in AI model training, especially within multi-GPU environments?

Balancing efficiency, precision, and model quality requires careful model and system co-design. There are mainly two approaches for trading precision with efficiency: kernel-level and modeling-level. At the kernel level, we can use INT8/FP8/FP4 quantization to achieve more TFLOPs because low-precision tensor

cores are faster. However, we need careful upcasting and downcasting to preserve model quality. At the model level, we can perform compression and pruning. Both are lossy operations and require careful execution by running numerous experiments.

5. Integration & Ecosystem

What critical considerations come into play when integrating cutting-edge open source AI solutions into established enterprise infrastructures?

Flexibility, scalability, and performance are three key factors I would consider for adopting open-source AI into an enterprise. First of all, every enterprise has its own specific requirements. For example, in LLM inference, some enterprises prioritize throughput, while others focus on latency. The project must be flexible enough to meet these different needs. Secondly, the solutions have to be scalable. Running on thousands of GPUs is a very different challenge from running on a single GPU. It is extremely difficult to make a project scalable if it is not designed to be so from the start. A

good indicator is whether existing adopters are operating at a scale similar to your enterprise. Thirdly, people should run careful benchmarks to measure performance in advance. Benchmarks reported in papers are not always accurate due to variables like hardware and hyperparameters. Enterprises should conduct benchmarks using their own workloads and hardware to measure actual performance before making a decision.

6. Community Collaboration

How has community collaboration influenced your projects, and can you share an instance where external contributions led to a pivotal breakthrough?

Community is a big driver for open-source projects, including Liger Kernel. One of our essential features, Flex Chunk Loss—the first optimized open-source post-training losses—was inspired by the open-source community. The PyTorch team provided a prototype of implementing chunking with torch compile. We gener-

alized the solution and applied it to more post-training losses like DPO, ORPO, and GRPO, achieving approximately 80% memory reduction. It turned out to be a huge success and helped many post-training researchers.

7. Future Horizons

Looking ahead, which emerging trends in hardware acceleration and open source AI most excite you, and how do you envision they will reshape the industry?

I am excited by the growing trend of open-source GPU kernels. They have been notorious for their difficulty to develop and debug for the general public because they require a deep understanding of GPU hardware and considerable insider knowledge. It is awesome to see

companies like DeepSeek release multiple GPU kernels during their recent open-source week. I believe this can democratize GPU efficiency and accelerate AI progress.

"These are my own opinions and not the position of xAI."

INSIDE VECTARA'S AI JOURNEY:

In a candid conversation with Marktechpost, Amr Awadallah—CEO of Vectara and veteran of Yahoo and Cloudera—shares his vision on blending open source with proprietary innovation, ensuring the reliability of AI outputs, differentiating Vectara's platform, and offering advice for entrepreneurs and developers working in production-level open source AI.

Dr. Awadallah, with your extensive background from Yahoo, Cloudera, and now leading Vectara, how do you approach integrating open source and proprietary methods when developing AI solutions?

The integration of open source and proprietary elements in our AI solutions isn't a one-size-fits-all answer—it's about aligning the approach with your business model. If your goal is to build a project purely for community success, a fully open source approach with permissive licenses like Apache or MIT might be the right path. However, the challenge with a fully open model is that while it can create a phenomenal product (just think of the Apache web server), monetization becomes nearly impossible.

At Cloudera, we learned early on that building a business requires more than offering support on top of open source. That model—charging for support after giving away the code—has a natural ceiling. Customers might only pay enough to cover the salary of one engineer until they decide to build the solution internally or simply leverage a hyperscaler's resources.

So many companies have evolved towards an "open core" model: the core functionality is open source, but

enhanced features or performance improvements are available only to paying customers. For instance, while Apache Spark is completely open source, providers like Databricks have layered proprietary functionalities on top when delivering fully managed cloud services.

At Vectara, we're taking a slightly different approach called "open perimeter." Our core remains proprietary—guarding our critical competitive advantages—while we open source components that sit on the platform's edge. For example, we've open sourced our hallucination evaluation model. This not only generates community interest but also serves as a gateway to attract developers and users. Our internal version is even more refined—about 20% more accurate and supporting additional languages—ensuring that our paying customers receive enhanced performance. In essence, while the open source elements build engagement, the proprietary core enables us to extract value and provide a competitive edge.

Vectara's 'trusted GenAI' framework is generating buzz for tackling issues like hallucinations and bias. How does your platform address these challenges while still leveraging open source technology?

At its heart, our platform is not just about stopping hallucinations—it's about building an AI assistant platform that is robust, secure, and scalable across business applications. The framework we employ has four

pillars:

1. Platform Approach:

We deliver a comprehensive end-to-end platform that

Open Source AI

allows businesses to build AI agents and assistants without reinventing the wheel each time. Think of it as how you use a database—there's no need to cobble together multiple disparate open source components to achieve functionality. Instead, you have one well-integrated platform.

2. Accuracy and Reliability:

Accuracy, for us, involves two main aspects. First, we tackle hallucinations—ensuring that the generated outputs are reliable. Second, we focus on the “R” in retrieval augmented generation (RAG). While the generation of text has become somewhat commoditized, retrieving the right information from a vast corpus is the real challenge. Our secret sauce is our highly optimized retrieval process that finds the most relevant “needles” in the haystack of company data. We also incorporate a hybrid system where the open source hallucination detection is complemented by our enhanced internal model.

3. Security:

Large language models are vulnerable to prompt

attacks—much like social engineering with humans. Imagine someone tricking the AI into revealing sensitive data. To mitigate such risks, we enforce rigorous role-based and entity-based access controls. Our platform ensures that sensitive data remains compartmentalized and only accessible to those with the proper clearance.

4. Explainability and Observability:

In regulated industries like finance or healthcare, merely getting an answer isn't enough—you need to know the source of that answer. Our system is built with built-in explainability. It tracks the provenance of every piece of data used in the answer, allowing businesses to verify and audit the reasoning process. This transparency is crucial not only for compliance but also for continuous improvement.

In summary, our trusted GenAI framework builds a production-ready system where each AI-generated answer is fact-checked through rigorous retrieval and detection processes, while the open source elements accelerate our development pace.

What innovations set Vectara's platform apart from traditional enterprise search and the emerging wave of generative AI solutions?

The fundamental innovation lies in delivering an end-to-end platform that functions like a database—removing the need to assemble a myriad of open source tools into a makeshift solution. Traditional enterprise search systems often require you to integrate components like document parsers, indexing services, and various AI models manually. With Vectara, you interact with a single, cohesive API.

This unified approach allows for dynamic adjustments. For instance, our system can automatically select the appropriate embedding or ranking model based on the language, context, or even the type of document. Whether it's a PDF, a PowerPoint, or an image of a diagram, our platform evaluates and selects the best strategies to retrieve and process that information. Another key differentiator is our dynamic ranking mechanism. Every document in our system can be enriched with metadata—like author details, time-

stamps, security levels, and even user feedback such as upvotes and downvotes. This metadata isn't just stored; it actively influences our scoring functions so that, for example, the most recent version of an HR policy is ranked above older versions. Additionally, real-time updates are a built-in feature. If a document is found to be outdated or inaccurate (say, an incorrect answer about pizza cheese), it can be corrected immediately, ensuring that the system learns and evolves continuously.

Finally, our hallucination detection stands out. We have engineered a classifier model with around 600 million parameters that can detect factual inconsistencies in milliseconds. This speed and accuracy are crucial when integrating multiple data sources in real time—allowing our clients to confidently rely on the system even in high-stakes environments.

Looking ahead, what breakthrough trends in open source AI do you anticipate will redefine production environments over the next five years?

There's a wave of innovation on the horizon that's reshaping how we think about and implement AI solutions. One significant trend is the move towards multi-modal AI. Today, most systems are optimized primarily for text, but tomorrow they'll need to process images, diagrams, video, and even voice concurrently. Imagine an AI that can interpret a Kubernetes cluster diagram without the text, or extract meaningful insights from a video by understanding both visual cues and audio patterns. We're already working on solutions that bridge

these modalities—leveraging open source tools where they excel and integrating our own enhancements where needed.

Another trend is the evolution of AI agents. We're transitioning from basic retrieval augmented generation (RAG) to more advanced systems that decompose complex queries into multiple sub-queries. These agents can coordinate across various data sources—from SQL databases to microservices—and

Open Source AI

seamlessly integrate the results. While many of these solutions are still in their infancy, I see a future where AI agents manage transactional processes—like filing a vacation request or ordering supplies on your behalf. However, given the current hallucination rates, truly transactional AI won't be mainstream until we achieve near-zero error rates in factual accuracy.

Beyond multimodality and intelligent agents, the next

Finally, what key advice would you offer aspiring entrepreneurs and developers who are eager to harness open source AI in production environments?

There are two major pieces of advice I'd offer, and they target two different audiences: the developers creating open source projects and the entrepreneurs building businesses.

For developers, focus on building something that genuinely works and brings value to the community. Your primary goal should be functionality and adoption—let the community help you refine and improve your project. Open source thrives on collaboration; if your project is solid and addresses real pain points, you'll naturally attract contributors and users.

For entrepreneurs, the message is different. Do not start with the technology for its own sake. Instead, begin by identifying a clear customer problem. Ask yourself: What pain point am I solving? Why will my solution command a premium? Too often, I see pitches that focus solely on the underlying technology, neglecting to

wave will likely focus on ensuring that AI systems are not just powerful but also safe. As these systems take on more critical tasks, the need for robust safety nets, real-time monitoring, and explainability will only intensify. We may see widespread adoption of what I call "authentic transactions" where AI systems not only retrieve and generate data but also perform safe, verifiable transactions in high-stakes environments.

articulate the market need or the specific use case that justifies a business model. Remember, technology is merely a means to an end—it's the value you create for your customers that counts.

Another crucial piece of advice for early-stage startups is to embrace agility—not just in your product roadmap but also in your team dynamics. Every team member's performance is critical. Be prepared to make tough decisions early on. Your time and the speed at which you iterate are your most valuable assets in a competitive market.

Ultimately, the key is balance: leverage open source to accelerate development, but always build a compelling business case around solving a real-world problem. The market will reward those who can combine technical excellence with a deep understanding of customer needs.

Dr. Amr Awadallah



Dr. Amr Awadallah is the CEO and Co-founder of Vectara, a company that is enabling organizations and enterprises to leverage GenAI in business applications by reducing risks from hallucinations, bias, copyright infringement, and model weight pollution.

Amr previously served as Vice President of Developer Relations for Google Cloud. Prior to joining Google in November 2019, Amr co-founded Cloudera in 2008, and as Global Chief Technology Officer, he spent 11 years working closely with enterprises around the world on how to ingest and extract value from big data (he famously coined the concept of "schema-on-read vs schema-on-write"). Cloudera went public in 2017 on the New York Stock Exchange, and was acquired by KKR + CD&R in 2021 for \$5.30 billion.

Earlier in his career, Amr served as Vice President of Product Intelligence Engineering at Yahoo! from 2000–2008. He joined Yahoo after they acquired his first startup, Aptivia, in mid-2000, which was a search engine for online product information. Amr received his PhD in Electrical Engineering from Stanford University, and his Bachelor's and Master's Degrees from Cairo University, Egypt.

Amr is one of 10 luminary scientists and technologists that were granted honorary citizenship to Saudi Arabia in 2021.



AI PCs – A BIG MOMENT FOR OPEN SOURCE AI

by
DARREN OBERST,
CTO, LLMWare.ai

At LLMWare.ai, we have been focused on local deployment of small language models for enterprise RAG and Agent automation since the very first day of launching in open source in October 2023. Our mission has remained true to today and we believe even more strongly that bringing AI capabilities to your data, whether data centers or directly to laptops such as AI PCs (laptops or PCs with integrated GPU or NPU acceleration for AI inferencing), is the future.

It might be difficult at first to appreciate the nexus between the AI PC and the open source AI community, but we will make the case that it may be the single most important new technology for open source AI enterprise adoption in 2025 – and offers a transformational opportunity for deploying open source AI models at scale across the enterprise.

AI PC meets the Small Model moment

The AI PC is an initiative started by Microsoft in early 2024 as encouragement to chip manufacturers and OEMs to expand the neural processing capabilities in Windows laptops, desktops and workstations, including GPU, NPU and CPU. There are targets set in “trillions of operations per second” (TOPS) in 2024, with the first scale enterprise roll-outs of these AI PCs happening in 2025. An increasing number of standard-issue corporate laptops from Dell, HP and Lenovo will have 40+ TOPs available in GPUs and NPUs, from chip manufacturers such as Intel, AMD and Qualcomm. It is expected that more than 150 million AI PCs will be sold in 2025, as enterprises refresh their laptops and that 60% of laptops and PCs sold will be AI-capable by the end of 2027.

It is likely that 2025 marks the beginning of a virtuous cycle over the next few years, in which OEMs and chip-makers compete to increase TOPs generation, software developers increasingly build out the low-level enabling capabilities, application developers write features that leverage these TOPs capabilities, and enterprises adopt at scale as standard operating procedure. Today, on AI PCs, models up to 32 billion parameters (with quantization/optimization) can be run in day-to-day work, all at zero incremental token costs and with zero risk of data exposure – with up to 14B parameters running at great inferencing speed. It is reasonable to believe that what runs pretty fast in 2025 will run very fast in 2026–2027 – and that model infer-

encing performance will be an increasingly important benchmark on the roadmaps for all of the major edge chip and device makers for years to come.

This increased computing capability coincides in a perfect storm with the evolution of open source small language models. The open source foundation models released in the second half of 2024 and beginning of 2025 in the size range of 3B–14B are truly stunning in their improvements from previous vintages and, in our estimation, can likely meet 90%+ of all enterprise RAG and Agent tasks today. The latest releases from Qwen 2/2.5 series, Llama 3.2+ series, along with Mistral, Yi, DeepSeek and Phi-4 from Microsoft, all represent substantial improvements in smaller models from 1–14B parameters over the previous generations. The AI PC becomes the perfect deployment platform for the widespread roll-out of specialized lightweight applications and workflows that leverage these open source small language models. In addition, high quality multi-

modal vision-to-text models (Qwen-VL, Llama, Phi-Vision) can run on the edge, and will only get faster in time. It is reasonable to believe that with increased model improvements over the next 12–18 months, virtually all language-based use cases, will be achievable by models in this size range, and deployable on an AI PC. In 2025, many Windows enterprise users will experience local edge inferencing with small open source models for the first time—and on an AI PC. It will be eye-opening to most users the extent that high-quality models can be run relatively quickly and easily, locally on a laptop.

As more enterprises have the capability at scale to run high-quality models up to 14B parameters on their local equipment, privately, securely and at \$0 incremental inference costs, how does this change the paradigm of how AI gets rolled-out?

Open source is perfect for the next 100 enterprise AI projects – but how to deploy?

After the Big Flagship AI Chatbot project, there is no standard “second” AI project in the enterprise. Rather, there are 100 next projects, typically among different departments and teams, sometimes in chat, and often in some workflow automation or offline or batch processing, in which there is a high intersection of corporate documents and sensitive knowledge-based processes. Usually, this is where cost, complexity, and deep integration into the fabric of the enterprise is needed – and this presents the perfect entry points for small open source models. The question remains, however, how to set up in a cost effective and practical way?

Traditional AI deployments can be a challenge for these use cases. Once you get past the high-volume, high-concurrency, continuous batching chat use case with relatively massive compute scale, there are these 100 projects with “moderate volume, moderate concurrency” requirements – and for anyone who has tried to set up these types of cost-efficient, but high-quality inference servers, there is the fundamental challenge of balancing GPU utilization with managing memory saturation – at reasonable cost. There will be spikes and valleys in users, and potentially many hours with negligible (if any) inferencing traffic. However, to handle usage spikes, enterprises are still required to provision a larger infrastructure, and it is really easy to saturate GPU memory with even a few concurrent users (in a relatively modest GPU setup). Utilization and cost can be a major challenge in these “moderate volume” use cases – combined with all of the usual issues such as

BCP, redundancy, test and dev environments, security, authentication, token tracking for charge-backs, need for 24x7 up time, user expectations on inferencing performance, need to monitor and manage a complex environment, change management, etc.

On the other hand, if you think of a different topology, the problem looks very different. A thought experiment for anyone new to AI PCs: What if you keep the documents on a private cloud or data center server with retrieval mechanisms, and then distribute the model inferencing to the edge? For many of these “moderate” use cases (e.g., 1–14B parameter models accessed by a group of tens to low hundreds of people), there is a strong case to be made that distributing the models for local inferencing results in radically lower costs. In this scenario, rather than pushing data “up” to a public cloud API, push it “down” to a secure corporate laptop, and run the inferencing at zero incremental cost.

While this may sound very retro or radical, depending on your perspective, there is a good analogy for this topology. This is the way that most of us work with databases today in the enterprise. A very small number of people directly touch the database, but they create extracts, reports and pivot tables, and send to business users who then drop into Excel, and run the analysis and do the work directly on their laptop CPU. In this common scenario, the data is “pushed down” by a filtering and retrieval mechanism, and then the real analysis and interaction is all done locally.

Turning point – inferencing as the workload that drives computing

For fifty years, since the old mainframe roamed the earth and dominated computing, the humble database read and write has been at the core of the computing paradigm and arguably has been driving compute topologies, e.g., a database is a shared resource that needs to be accessed by multiple clients (put it on a server that can support concurrency and multiple client connections of a large shared resource).

Arguably over the last twenty years, the web transaction, usually backed up a database read and write, has supplemented the database as the transaction du jour that drives computing topologies, resulting in increasing thinner clients as more of the processing is picked up by the web server, which also must be designed to handle concurrency of users and connections, often times as a “front end” or middle-end server connecting to a back-end database.

Both the database and web transaction pulled workload and processing from the endpoint client – as most of the good stuff was sitting on the shared server resource.

The AI PC represents a turning point in which model inferencing becomes the marginal requirement that drives the next wave of computing – and unlike the shared resource nature of database and web transactions, model inferencing is much more like a singleton compute-driven process that can be easily distributed. The PC revolution was captured best by Bill Gates’s radical vision of a PC on every desk, at a time when most thought that “real computing” operated on a mainframe (e.g., the “frontier models” of their day), with many questions about use cases and viability of the PC as hobbyist electronic toys. How ironic (but also quite fitting) would it be if Microsoft’s push around the AI PC becomes a tipping point from the current era of Cloud

Computing “centralization” back towards more decentralization and localized deployment?

Will we have inferencing on every desktop? And if so, what does this mean for the ongoing cloudification of everything—does that trend reverse or rebalance? Furthermore, what does this mean for the competitive landscape between the cloud providers and edge device OEMs and chipmakers—does this mark a turning point in the relationships and competitive dynamics between the cloud providers and the device OEMs and chip makers—is everyone essentially fighting for the same incremental compute inference unit?

There has been a lot of ink spilled over the last two years about all of the ways that generative AI could be disruptive to the computing landscape, and require new paradigms for how to manage and deploy IT and address concerns about data privacy, safety and cost. What if the solution for enterprise deployment of generative AI is hiding in plain sight and unlocked by the AI PC—don’t reinvent your enterprise for AI, rather bring AI directly into the (sometimes forgotten and ignored) computing laptop backbone of the company and leverage a lot of existing infrastructure?

What if model inferencing over time looks more the spreadsheet, rather than the data warehouse?
 What if model inferencing has zero incremental cost?
 What if data is “pushed down” to the user device for inferencing, rather than data moving “upwards” to a centralized cloud service for processing?

What if the frontier models—and all of their complexity and cost—start to look like the mainframes of the AI world—useful for a relatively small number of tasks, while most of the world opens up their laptops and runs their generative AI workloads locally?

Conclusion: Open Source and the AI PC – A Turning Point for Enterprise AI

The AI PC represents a transformative moment for open source AI in the enterprise. By bringing powerful, smaller open source models directly to local devices, enterprises can achieve a radical shift in how AI is deployed—securely, privately, and at zero incremental inference costs. This new model challenges the cloud-centric paradigm, offering a practical and scalable path forward for the “next 100” enterprise AI projects that often demand moderate volume, moderate concurrency, and deep integration into business

processes. As AI PCs become more prevalent, the opportunity for open source AI to thrive at the edge could lead to a significant rebalancing of the computing landscape—one where inferencing at the desktop becomes as commonplace as running a spreadsheet, and where the future of AI deployment is as much about decentralization as it is about raw processing power.

GOODBYE OPENAI:

The Future is Open-sourced

Author-
Devvret Rishi (CEO and Cofounder)

 **Predibase**

Devvret is the CEO and Co-founder of Predibase. Prior he was an ML product leader at Google working across products like Firebase, Google Research and the Google Assistant as well as Vertex AI. While there, Dev was also the first product lead for Kaggle – a data science and machine learning community with over 8 million users worldwide. Dev's academic background is in computer science and statistics, and he holds a masters in computer science from Harvard University focused on ML.

Let's face it: OpenAI has been a game-changer making it easy to rapidly build prototypes with large language models (LLMs). In just a few days, an engineer can develop a powerful demo for a new GenAI feature using OpenAI's APIs. But when it's time to move these LLMs into full-scale production, engineering teams are increasingly saying "no thanks" to commercial solutions.

Why? Because open-source alternatives—like DeepSeek R-1, Llama 3.3, and Qwen 2.5 —now deliver perfor-

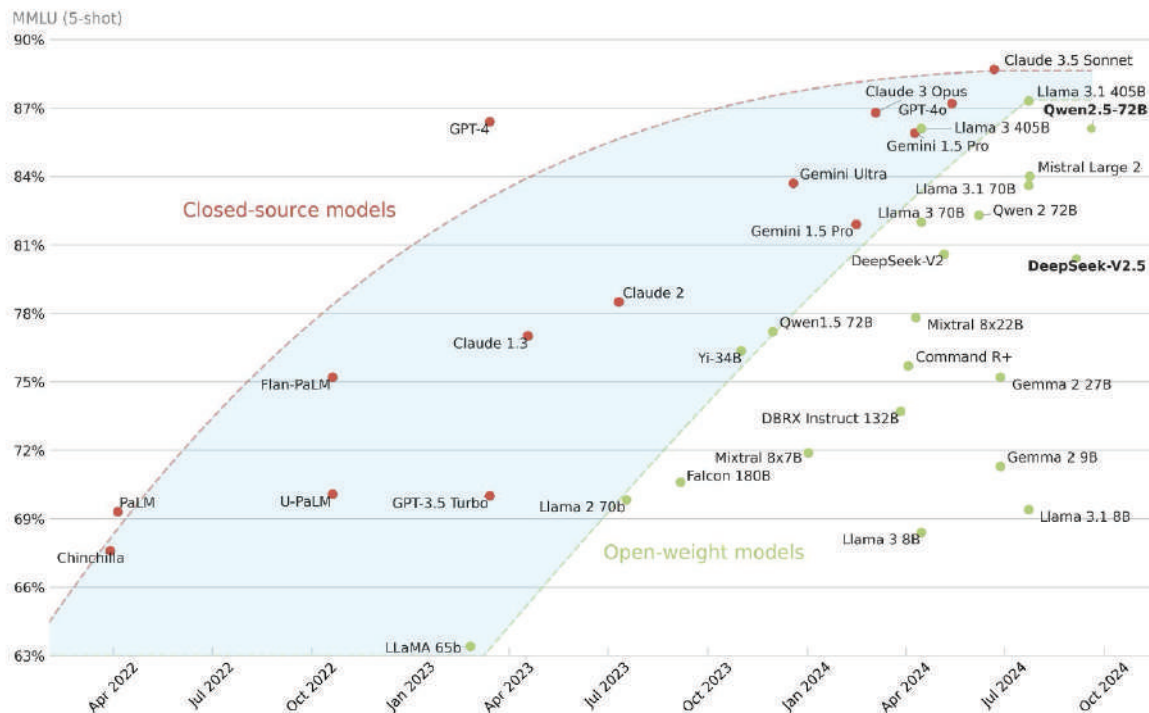
mance on par with (and sometimes better than) their proprietary counterparts, all while consuming fewer resources and safeguarding valuable IP. No more vendor lock-in, no more sharing sensitive data; just cost-effective and efficient AI. The real question isn't whether to ditch commercial providers but how soon you can make the jump to open-source. Let's explore the motivation for open-source models and the path to production in more detail.

Open-source LLMs are Closing the Gap with Commercial Models

Closed-source vs. open-weight models

@maximelabonne

OpenAI's new o1 models are not represented because not directly comparable with the results.



Source: Maxime Labonne's model comparison

<https://x.com/maximelabonne/status/1838170077021053004>

Safeguarding IP and Avoiding Vendor Lock-In

If you're building mission-critical AI applications, then relying on locked-down LLMs from a commercial vendor can become more of a liability than an advantage. These closed-source models are a black box, restricting access to the model source code and prohibiting you from exporting model artifacts, thus making it impossible to move on. If you're planning or actively using commercial models, then you need to first consider these key pain points:

Data Control: Handing your sensitive data to a 3rd party model provider is a huge risk. Keeping everything in-house means you stay in the driver's seat, especially for industries handling regulated or sensitive PII.

Intellectual Property: AI is becoming the heart of countless products, and you don't want to be stuck layering a thin UI on top of someone else's locked-down API. Owning your model is the key to true innovation (and peace of mind) and only way to avoid lock-in.

Production SLAs: Commercial APIs can introduce unpredictable latency and reliability issues into the mix. When you own the entire stack, you also own performance guarantees—no more crossing your fingers and hoping a third-party service stays up.

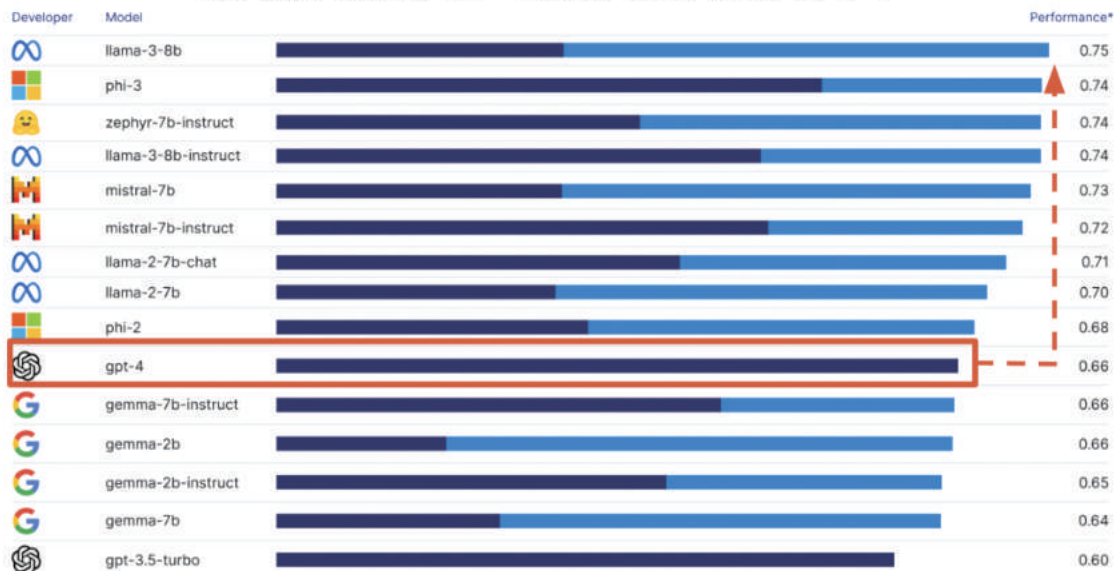
The Rise of Specialized Models: Smaller, Faster, Cheaper

Open-source models once seemed less capable, but developments in fine-tuning have changed the narrative. Fine-tuned open-source models now regularly outperform large commercial models while remaining significantly smaller in size. To illustrate this point, we published the Fine-tuning Model Leaderboard (<https://pbase.ai/leaderboard>), a series of benchmarks that compares the performance of fine-tuned open-source LLMs against OpenAI models across 30 distinct tasks—85% of the fine-tuned models outperform GPT-4.

Fine-tuning isn't just about performance and shaping the model to your use case—it's a gateway to faster inference speed and reduced costs. Take this real-world example: Checkr, the leader in automated background checks, fine-tuned Llama-3-8B and saw over a 10-point bump in accuracy, a 30x speedup in inference, and a 5x cost reduction compared to GPT-4 (detailed story: <https://pbase.ai/checkr>). That's the kind of ROI that open-source, tailor-made AI can deliver.

Fine-tuning Leaderboard

We fine-tuned 20+ SLMs: 85% beat GPT-4



<https://predibase.com/fine-tuning-index>

*Performance averaged over 31 tasks

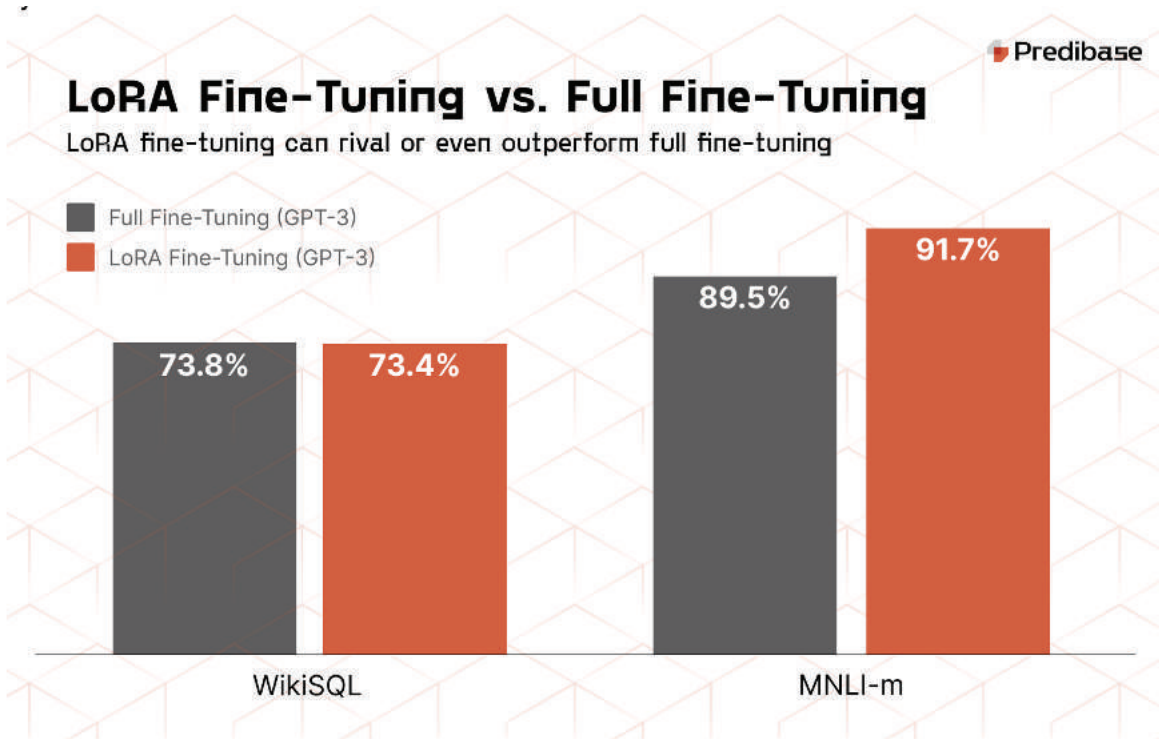
Source: [Predibase's](https://predibase.com) Fine-Tuning Leaderboard demonstrates how open-source models, when properly fine-tuned, outperform GPT-4.

The Path to Efficient Fine-tuning with LoRA

Fine-tuning an existing open-source model is generally a more efficient path to customization than training a new model from scratch. When you start fresh, you need enormous amounts of data, computing power, and time—all of which can quickly become prohibitive. In contrast, fine-tuning builds on the existing strengths of a pre-trained model, letting you adapt it to your specific tasks or domains without investing in a full, resource-intensive training pipeline.

Within fine-tuning, LoRA-based fine-tuning has emerged as a game-changer and the industry standard.

By adjusting fewer than 1% of a model's parameters, LoRA delivers performance comparable to a fully custom-trained model—but at a fraction of the cost, time, and compute. The traditional approach of full fine-tuning (fine-tuning all of the model's parameters) ties up GPUs for days or weeks, whereas LoRA speeds things up by freezing most parameters and focusing on lightweight adapters. The end result? You get nearly the same top-notch performance for specialized domains without sinking a fortune or enduring lengthy training cycles.



The Future of Fine-tuning with Reinforcement Learning

For years, Reinforcement Learning (RL) and fine-tuning belonged to completely different worlds. On one hand, fine-tuning is all about supervised learning: you train a model on labeled data, and it learns to predict the correct outputs for given inputs. On the other hand, RL focuses on decision-making in dynamic environments: an agent interacts with its surroundings, gets feedback (rewards), and adapts based on trial and error. Enter Reinforcement Fine-Tuning (RFT), a new twist on applying RL principles to fine-tuning tasks.

We won't explore all of the intricacies of RFT, but the opportunity for model customization is important to understand. One of the historic blockers that teams faced when fine-tuning models was securing the required

labeled data for training. With RFT-based techniques, you can now adapt models for your use case without needing massive amounts of data. There are three scenarios where RFT shines:

You can't provide labeled data, but you can verify the output: If you don't have explicit ground truth data but can still check for correctness (like transpiling source code), RFT handles the use case beautifully.

You have labeled data—but barely any: If you're working with under 100 labeled examples, RFT outperforms standard fine-tuning approaches that often choke on limited datasets.

You see a big boost from chain-of-thought reasoning: When your model benefits significantly from step-by-step logic at inference time (e.g. complex math problems or game strategy), RFT is built to capitalize on that advantage.

RFT makes it even easier to customize open-source LLMs and train your own reasoning models akin to DeepSeek-R1 for a broad range of powerful use cases. You can learn more about that in this blog post: <https://pbase.ai/RFT-vs-SFT>

Charting a Course to Open-Source Production

While open-source LLMs present a compelling path forward, the process of getting them to a robust production state has its own hurdles:

Fine-Tuning Complexity: Not all teams have the machine learning and MLOps expertise to fine-tune effectively.

Infrastructure Management: Deploying LLMs requires scalable and reliable infrastructure to handle real-world demands.

Quality Assurance & Ethics: AI systems need guardrails to ensure they're safe, secure, and aligned with organizational values.

That's where specialized platforms like Predibase come into play. By offering managed, developer-friendly solutions with easy-to-use fine-tuning capabilities, and production-ready serving infrastructure, you can unlock the true potential of open-source LLMs without sacrificing control or performance.

Looking Ahead: Custom AI as a Competitive Edge

We're witnessing a fundamental shift: the move from behemoth commercial models to fine-tuned, open-source solutions that deliver both efficiency and IP ownership. As organizations look to differentiate themselves through custom AI, specialized models will become the standard—providing superior performance, lower costs, and the assurance of data privacy.

For developers and AI teams, the message is clear:

Stay Nimble: Evaluate open-source solutions to avoid technical and contractual lock-in.

Own Your IP: Safeguard your data and models by taking control of your AI pipeline.

Leverage Fine-Tuning: Customize smaller models that can outperform larger commercial options.

Scale Responsibly: Invest in platforms or tools that simplify infrastructure and ensure best practices in security, privacy, and ethics.

This new era of AI is all about ownership, specialization, and efficiency. Open-source LLMs are poised to lead the charge, ushering in a future where developers can innovate faster, cut costs, and truly take control of their AI destiny.



 **Predibase**

Devvret Rishi
CEO and Cofounder

VERDICT: A LIBRARY FOR SCALING JUDGE-TIME COMPUTE

By: Nimit Kalra & Leonard Tang

Introduction

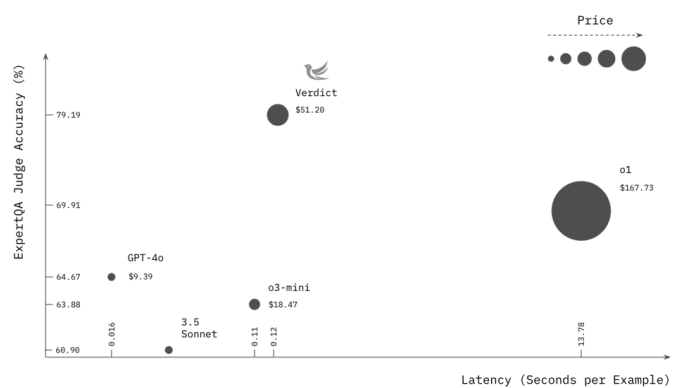
Despite tremendous progress in AI through scaled train-time and inference-time compute, AI remains unreliable in open-ended, non-verifiable domains. The key limitation is not generation—it is evaluation. Specifically, it is evaluation that is aligned with human taste, that is capable of generalizing to in-the-wild examples, and that is operationalizable at scale. Therefore, the next big leap for AI comes from gains in judging AI outputs. The next frontier is judge-time scaling. In service of this future, we developed Verdict, a fully open source library for scaling judge-time compute.

A Verdict judge is not an ordinary LLM-as-a-judge. A Verdict judge leverages greater inference-time compute for accurate, reliable, and calibrated evaluation. A Verdict judge draws from patterns in scalable oversight, HCI, reward modeling, and more. A Verdict judge is applicable across arbitrary domains and tasks. Of chief significance, a Verdict judge simply works very well.

Verdict judges amplify AI development across a wide range of use cases, including at least the following:

- Scaled evaluation of AI systems
- Runtime guardrails and oversight
- Verification to aid inference-time compute
- Rewards during reinforcement learning
- Synthetic data filtering

In this article, we break down why we created the Verdict library, what Verdict is, and finally how we can leverage Verdict for powerful downstream applications.



Verdict judges are able to achieve SOTA on the evaluation tasks of hallucination detection, content moderation, and fact-checking. Verdict judges even beat out reasoning models like o1 and o3-mini at a fraction of their latency and cost, as shown below!

Motivation: Towards Better LLM Judges

Automated correctness checks using LLMs, a.k.a. LLM-as-a-judge, is a widely adopted practice for both developers and researchers building LLM-powered applications. However, LLM judges are painfully unreliable. Today's LLM judges struggle with inconsistent output formats, mode collapse, miscalibrated confidence, superficial biases towards answer positioning, data frequency, model family, length, style, tone, safety, and numerous other failure modes. This makes the problem of evaluation twice the trouble, as both the evaluator and evaluatee may be unreliable.

One promising solution is to scale up judge-time compute — the number of inference tokens used for judging. We do this in a very particular way: by composing judge architectural primitives grounded in the scalable

oversight, automated evaluation, and generative reward modeling research.

This is the foundation of Verdict, our open source library for building compound systems to scale up judge-time compute. Verdict provides both the primitives (Units) and execution framework for building such systems. Instead of a single LLM call to produce a judge result, Verdict Judges combine multiple units of reasoning, verification, debate, and aggregation into a single judge system. This judge-time scaling allows Verdict systems to produce impressive results on evaluation tasks, oftentimes beating out even frontier reasoning models like o1 and o3-mini at a fraction of the cost and latency.

Verdict Mechanics

At the heart of Verdict are

1. Primitives for judging (Units), and
2. Methods for linking, orchestrating, and executing systems of Units.

Basic Anatomy of a Unit

A Unit is the fundamental building block of a judge system. A unit is composed of the following:

Prompt: The instructions for how a Unit should function.

Model: The model that the Unit calls to generate responses.

Scale: The domain that generated values must be restricted to. For example, the 1–5 Likert Scale, or the Yes/No Scale, or the Safe/Unsafe Scale, or any other ordinal or categorical Scale.

Input Schema: The values and types the Unit can accept, either from a previous Unit or from raw user-provided input.

Response Schema: The values and types the Unit's model can generate.

Output Schema: The values that are ultimately generated by the Unit. This usually involves either postprocessing values produced by the Response Schema or updating a cumulative state from prior Units. These values get passed to the subsequent Unit.

The structure of a Unit solves two major pain points of standard LLM judges:

1. The output structure of the judge is predictable even for small language models, governed in particular by the Scale, Response Schema, and Output Schema.

2. Format and function are specified separately. The Prompt implements the evaluation logic— the function—of the Unit, while the Scale and Schemas manage the format of the Unit.

In the context of a Verdict system, Units that are chained together are also automatically typechecked. The Output Schema of a Unit must match the Input Schema of its subsequent Unit. This allows for information to flow through a compound judge system in a stable and predictable fashion.

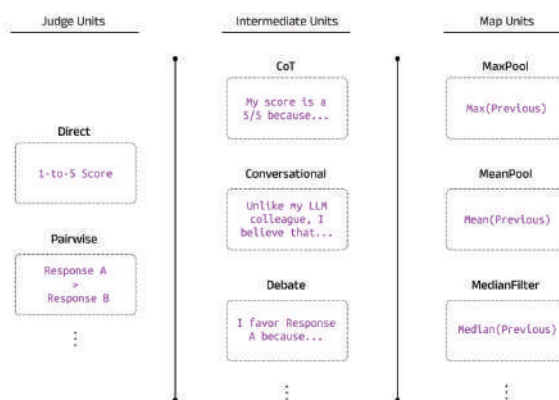


Figure 1: A few common Units in VERDICT, segregated by functionality. JudgeUnits produce a judge result (e.g. a float score or selected category); IntermediateUnits produce reasoning as part of a broader VERDICT system; MapUnits aggregate intermediate results into a final verdict.

Stitching Units Together

Units on their own only re-implement existing LLM-judge methods, albeit in a more reliable and consistent fashion. Stitching them together, however, unlocks the full power of Verdict and increased inference-time compute. Combining Units with the right architecture priors can yield impressive results across a variety of evaluation, judging, and reward modeling tasks.

Verdict draws inspiration from deep learning libraries like PyTorch vis-à-vis managing Unit groups and interactions. The most fundamental organizational principle is that of a Layer, which is a list of Units. This is

analogous to how neural network layers are a list of neurons.

A standard Layer propagates information through a judge system in a feedforward fashion. By default, subsequent Layer's Units receive the output of previous Layer's Units in a one-to-one fashion, and Units within a Layer are fully independent of one another. However, it is possible to customize Unit behavior both within a Layer (*using keyword inner=*) and between current and subsequent Layers (*using keyword outer=*). Below are common instances of how Units can be stitched together from Layer to Layer.

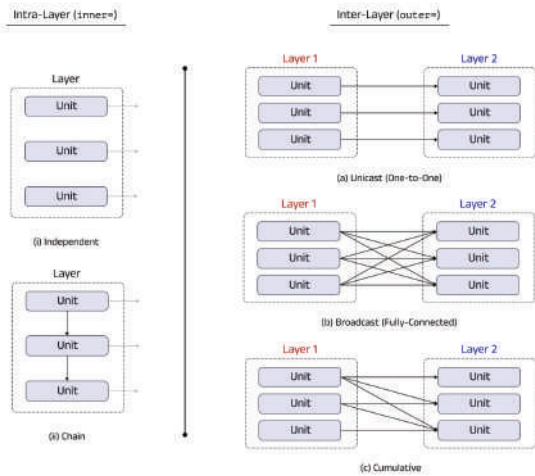


Figure 2: A few common VERDICT patterns for stitching Units together within and between Layers.

Verdict enables us to create complex judge protocols in just a few lines. For example, the following 20-line-or-so architecture enables us to reach SOTA on the ExpertQA evaluation task:

```
self.hierarchical_protocol = (
    Block()
    >> CategoricalJudgeUnit(
        name="judge",
        categories=self.output_categories,
        explanation=self.judge_explanation,
    )
    .prompt(self.judge_prompt)
    .via(
        policy_or_name=self.model,
        retries=self.retries,
        temperature=self.judge_temperature,
        **self.kwargs,
    )
    >> CategoricalJudgeUnit(name="verify", categories=self.output_categories)
    .prompt(self.verify_prompt)
    .via(
        policy_or_name=self.model,
        retries=self.retries,
        temperature=self.verify_temperature,
        **self.kwargs,
    )
)

self.pipeline = (
    Pipeline()
    >> Layer([self.hierarchical_protocol], repeat=self.repeat)
    >> MaxPoolUnit()
)
```

Verdict for Evaluators, Guardrails, Verifiers, and Reward Models

Verdict judges can be used anywhere to replace human feedback and verification. Naturally, they apply to at least the following scenarios:

1. Automated Evaluation of AI Applications. Verdict judges enable tailored and automated evaluation of AI applications.
2. Run-Time Guardrails. Verdict judges are guardrails that sit on top of AI applications running in production.
3. Test-Time Compute Scaling. Verdict judges are verifiers that help rank, prune, and select candidates during test-time compute scaling.
4. Reward Modeling & Reinforcement Learning. Verdict judges provide signal in reinforcement learning — particularly in settings where rewards are not verifiable. Verdict is well-suited for general-purpose verification,

given that it is:

1. More general than fine-tuned reward models. VERDICT judges readily apply across different tasks and domains, as seen by our experiments on safety moderation, checking for factual and logical correctness, and hallucination detection.
2. More stable and reliable compared to simple LLM judges. VERDICT judges beat out all simple LLM judges (and fine-tuned evaluators), barring the o1 models on JudgeBench, on the three tasks presented here.
3. Capable of generating soft rewards, unlike verifiers in formal settings like mathematics or programming.
4. Relatively low-latency and cost-efficient, which is necessary for methods leveraging heavy inference-time compute.

Empowering Researchers

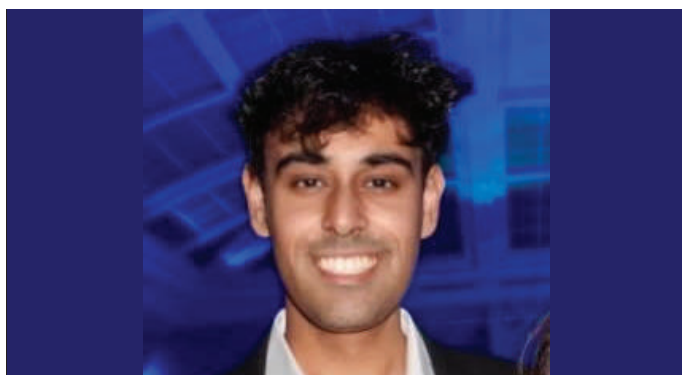
To summarize, we introduce Verdict: a modular, expressive, and flexible approach to automated evaluation of LLM outputs. By enabling the composition of diverse reasoning units—such as verification, debate, and aggregation—Verdict enhances the robustness, interpretability, and accuracy of LLM judges.

Verdict judges achieve SOTA or near-SOTA across a wide range of challenging evaluation tasks, including safety moderation, factual verification, and hallucination detection. Notably, Verdict-based judges—without any bespoke customization—surpass both 1) models that are finetuned specifically for each evaluation task, as well as 2) orders-of-magnitude larger judge models. This highlights Verdict’s potential as an efficient and scalable alternative for AI evaluation.

Beyond immediate performance gains, Verdict serves as a unified framework for building more transparent and adaptable evaluation pipelines. By offering a flexible architecture for reasoning and decision-making, we hope Verdict will enable researchers and practitioners to develop more reliable, interpretable, and scalable AI evaluation systems, advancing automated evaluation, oversight, and AI alignment in the broader machine learning community.

We are excited to see what evaluators, verifiers, reward models, and guardrails the broader ML community chooses to build with Verdict: <https://github.com/haizelabs/verdict>.

Bio



Nimit Kalra

Nimit Kalra is Member of Technical Staff on the research team at Haize Labs, where he is focused on aligning and robustifying real-world AI systems. Nimit’s research develops methodologies across adversarial robustness, uncertainty calibration, and automated evaluation, working toward building reliable, interpretable AI systems. Previously, he spent time at Citadel and in computer vision/robotics research, with an emphasis on domain adaptation. He is based in New York City and enjoys hiking, karaoke, and road trips



Leonard Tang

Leonard Tang is the Co-Founder and CEO of Haize Labs, where he works on solving the ultimate extant problem of AI: ensuring its robustness, quality, and alignment for any application. Prior to this, his research covered adversarial robustness, mathematical reasoning pitfalls, computational neuroscience, interpretability, and language models. Leonard dropped out of, before starting, Stanford PhD in computer science to pursue Haize Labs. In the very limited time he spends outside the company, he enjoys playing the guitar.



BORIS

YANGEL

**Head of AI R&D at
NEBIUS**

Could you please tell us a bit about your career and how you end up working for Nebius?

My background is in machine learning and engineering, and I've been working at the intersection of these fields for more than 15 years. My career began in search and ranking at Yandex, where I worked on improving image search quality and later led the image search relevance team. After that I worked at Machine Learning Group at Microsoft Research Cambridge, where I focused on automated Bayesian inference and Infer.NET inference framework. I then came back to Yandex, where I led the ML team responsible for launching its voice assistant, Alice. It is now a flagship product of the company, with more than 60M MAU. I've then transitioned to Yandex

Self-Driving Group, where I worked on neural world models and planning for self-driving vehicles and food delivery robots for several years. I then transitioned to deep learning for drug discovery at Charm Therapeutics before joining Nebius AI. At Nebius, I lead the AI R&D team, where we focus on exploring ways to improve large-scale AI training infrastructure, fine-tuning methods, and agentic systems. What drew me to Nebius was the opportunity to apply my expertise in ML and engineering at large scale, working with state-of-the-art computing infrastructure and systems.

As the Head of AI R&D at Nebius AI, can you explain the role of your team within the company and why AI research is essential to Nebius's mission?

Our AI R&D team serves two main purposes. First, we conduct research that aligns with our core business—providing computing infrastructure and resources to AI developers. This helps us discover innovations that could evolve into new products while also improv-

ing our understanding of customer needs. Second, we act as large-scale internal users of our products. By running demanding AI workloads ourselves, we ensure our infrastructure is robust, scalable, and ready to meet the needs of our customers.

Could you describe the infrastructure and tools your team uses and how they support your open-source initiatives?

We operate large GPU clusters with NVIDIA H100 and H200 GPUs connected by InfiniBand networks. This setup handles compute-intensive tasks like training complex models and running large-scale simulations. We use JAX for its performance and flexibility, although we've had to develop some tools ourselves where the ecosystem was lacking.

One example is our long-sequence attention library, which enables efficient processing of lengthy sequences across multiple GPUs or nodes. We open-sourced it to help address this gap in the ecosystem and to support others working with long-context models.

Another major challenge for the community is the lack of high-quality datasets—both problem datasets and agent trajectories—needed to train and evaluate AI agents. Collecting these datasets is expensive and requires scalable infrastructure to validate problems, set up environments, and ensure data quality. Our infrastructure allows us to handle these challenges efficiently, which is why we've released datasets to support the development of AI agents.

This is just the beginning. We plan to release more tools, datasets, and other artifacts in the future to help the AI community move faster and build better systems.

Your team has been investigating agentic systems, particularly for software engineering. Could you share more about this focus and its significance?

Our interest in AI agents comes down to three key points:

1. Understanding Customer Needs:

As developers increasingly use our infrastructure to build and deploy AI agents, we need to understand their challenges and requirements. This helps us improve our offerings.

2. Preparing for the Future:

We anticipate that AI agents will eventually buy and manage computing resources on their own. By developing these systems in-house, we're ensuring our infrastructure is ready for that shift.

3. Making AI More Accessible:

We're interested in whether building powerful agents will remain limited to a few leading labs or whether it's possible using open models. If the latter is true, we aim to support and accelerate that process.

Software engineering is a great testbed for AI agents because their results can be verified. Success in this

area can significantly accelerate software development, including the development of AI itself. It's a domain where you can clearly measure progress, which makes it easier to evaluate different techniques and approaches.

We started by exploring guided search techniques, which help agents systematically explore different solutions and learn which approaches work best. This method has allowed us to both improve agents built on open-weight large language models and boost the performance of more powerful models like GPT-4o. But guided search is just a starting point. Our long-term goal is to use reinforcement learning to create agents that continuously improve by learning from their mistakes. This involves creating feedback loops where agents analyze their past performance, refine their decision-making, and apply those lessons to new tasks. One way to build this loop is by repeatedly distilling systems that use guided search back into large language models, making them more efficient and capable over time.

Platforms for OPEN SOURCE AI

Model Cloud Hosting



Model Depot



Model Finetuning



Prompt Engineering



Observability



RAG Tools



MLOps



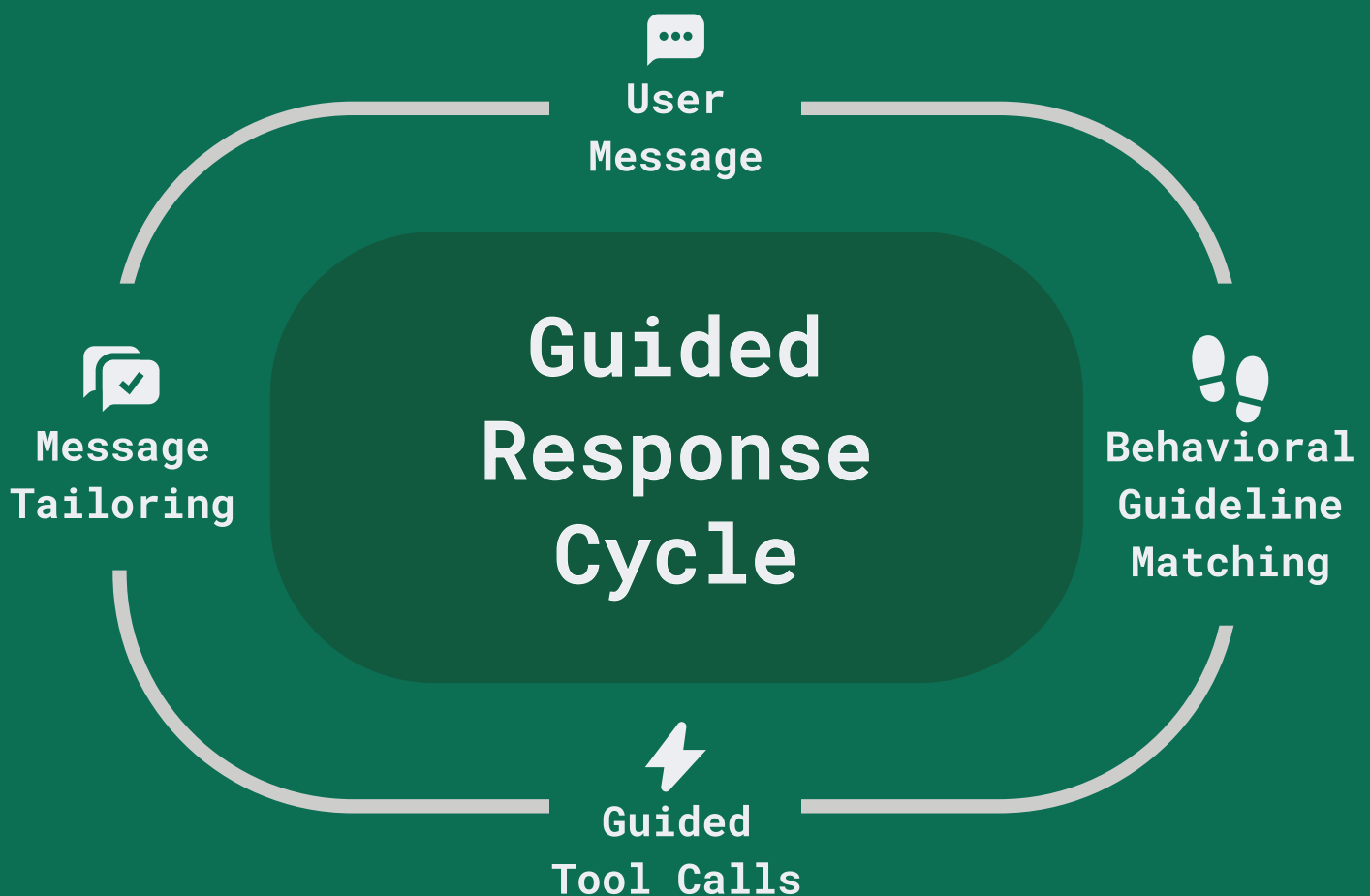
Vector Database



Flow-Based Conversation Design Is Dead.

👑 Long Live GenAI Conversation Design.

Parlant is an open-source, LLM-first conversation design framework that gives engineers and designers the control they need to build hallucination-free, high-containment AI customer service for enterprise use cases.



OPEN-SOURCE AI

MARKTECHPOST

📍 300 Spectrum Center Dr Ste 400, Irvine, CA

✉️ asif@marktechpost.com

🌐 www.marktechpost.com



Scan here to Partner

Designed By: Hema Dhawan

