The Goldilocks Approach to LLMs: Balancing Accuracy, Latency, and Cost for Optimal Performance [S62163]

Janaki Vamaraju, Deep Learning Architect and Scientist, NVIDIA
Elena Agostini, Senior Software Engineer, NVIDIA
Nik Spirin, Director of Generative AI and LLMOps, NVIDIA

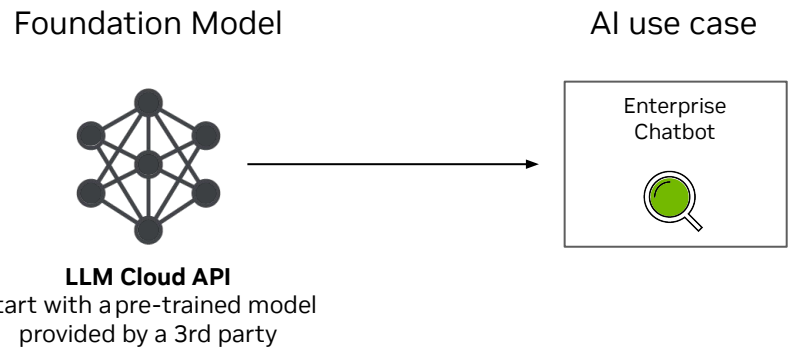
Business KPIs for an LLM Application

Balancing accuracy, latency, and cost for optimal performance

Accuracy	Helpfulness, harmlessness, honesty across a wide range of scenarios (e.g. language, task, domain)
Latency	Time-to-first token (TTFT), inter-token latency (ITL), end-to-end latency (E2E)
Cost	Output tokens per second (OTPS), total tokens per second (TTPS), requests per second (RPS)

Use an LLM Cloud API for an MVP

Building an MVP is easy and doesn't require any specialized skills



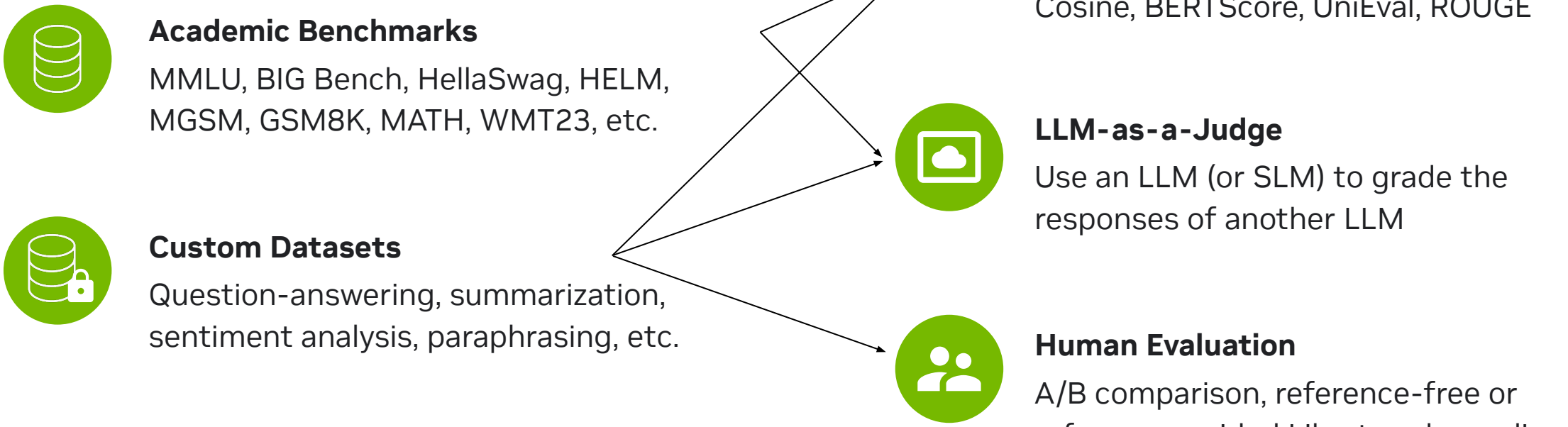
LLM Evaluation Scenarios

Formalize your success criteria early

Question \ Scenario	Easy	Hard
Is the answer (target variable) well-defined?	Yes	No
What dataset do we use for evaluation?	Proxy	Real
Does the evaluation dataset contain labeled ground truth data?	Yes	No
Who does the evaluation of LLM results?	Humans	Automated

LLM Evaluation Methodology

Combining automated benchmarking, human evaluation, and LLM-as-a-judge



Academic Benchmarks

MMLU, BIG Bench, HellaSwag, HELM, MGSM, GSM8K, MATH, WMT23, etc.

Custom Datasets

Question-answering, summarization, sentiment analysis, paraphrasing, etc.

Automated Evaluation

Exact match, Precision, Recall, F1, Cosine, BERTScore, UniEval, ROUGE

LLM-as-a-Judge

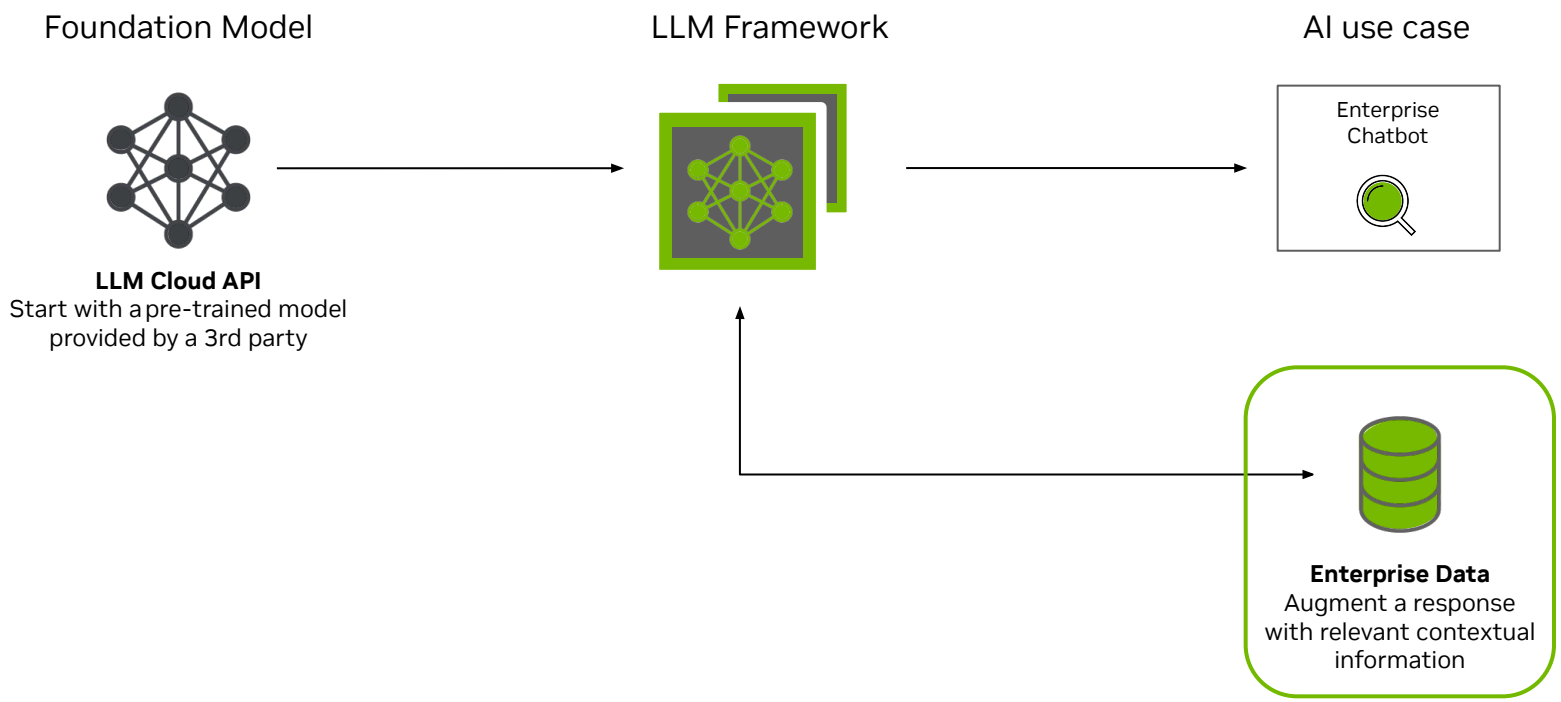
Use an LLM (or SLM) to grade the responses of another LLM

Human Evaluation

A/B comparison, reference-free or reference-guided Likert scale grading

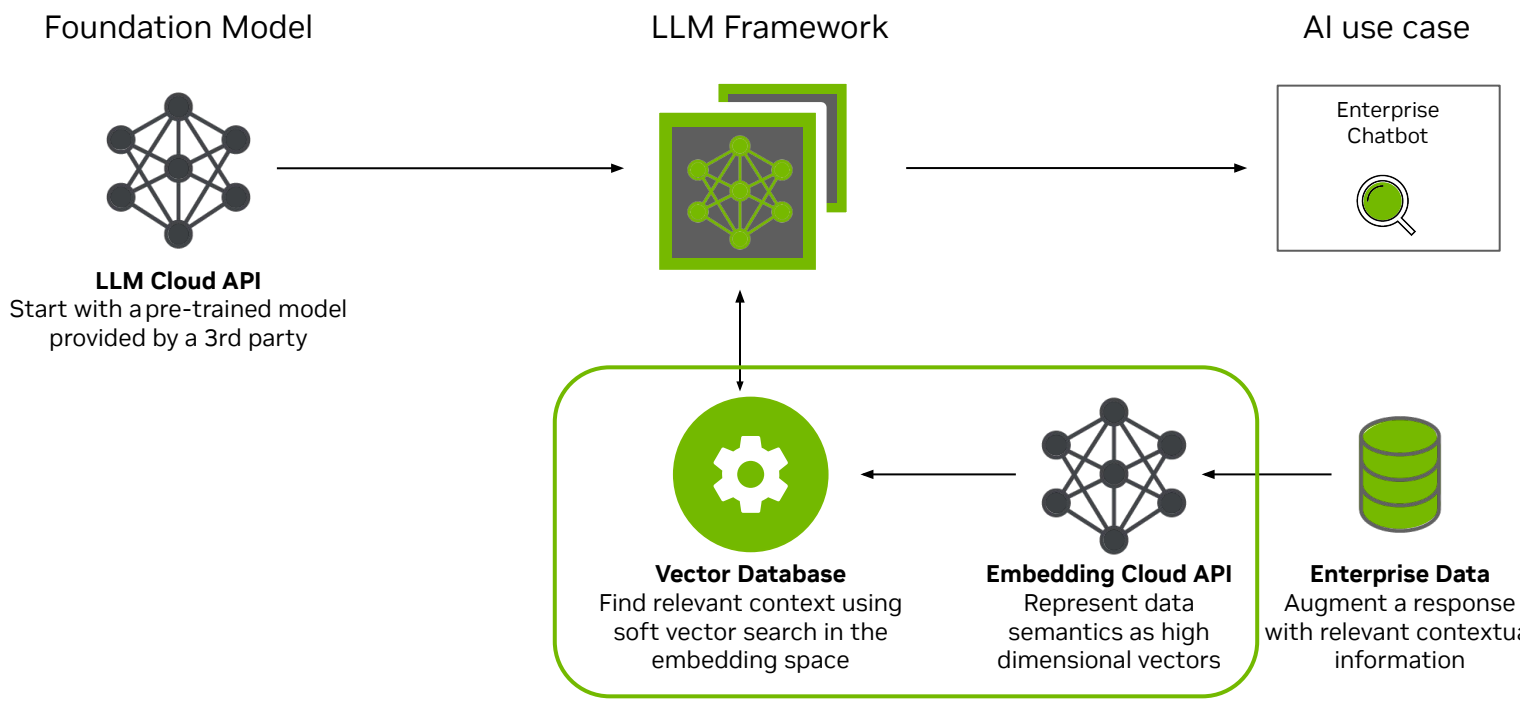
Use Retrieval-Augmented Generation (RAG)

Provide context at a query time to minimize hallucinations and keep LLM answers fresh



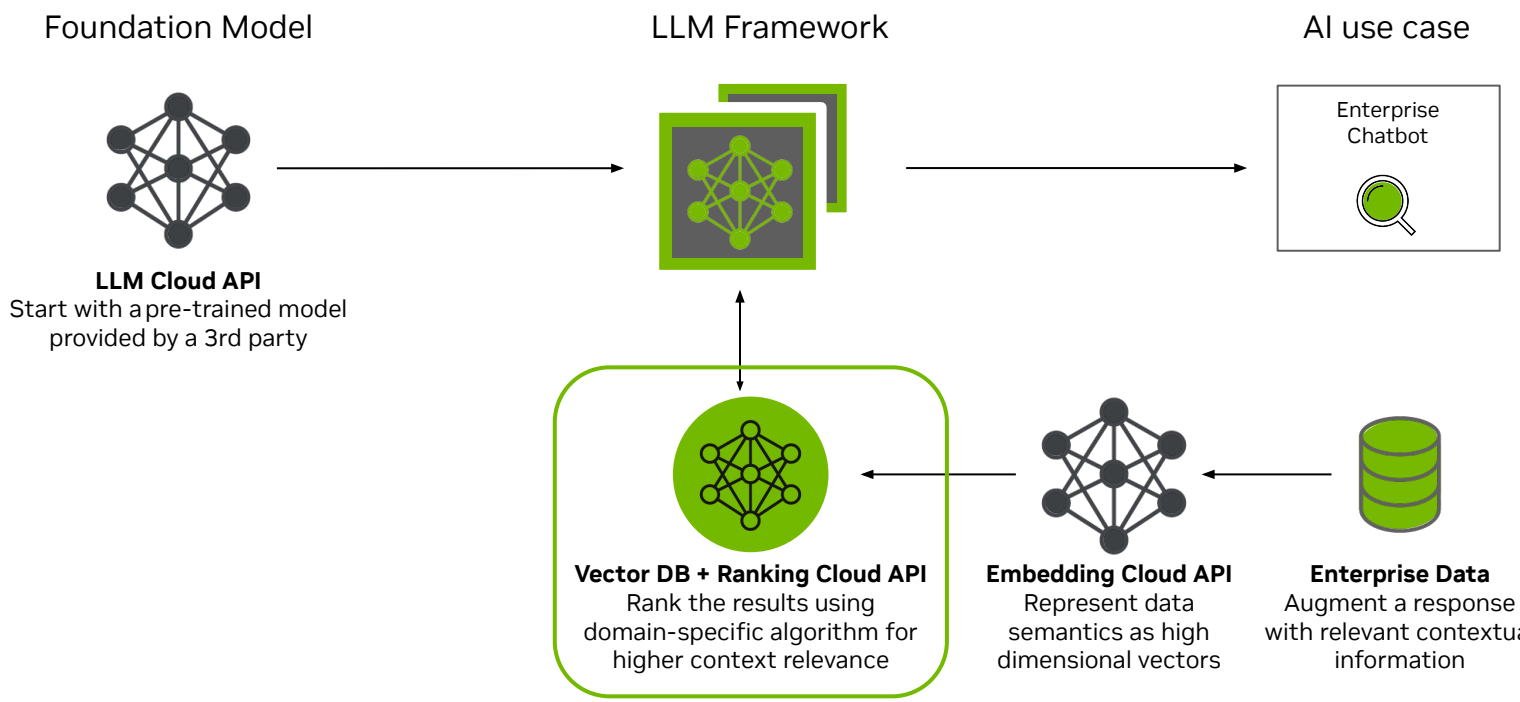
Use Retrieval-Augmented Generation (RAG)

Represent data as embeddings to support “soft” vector similarity search



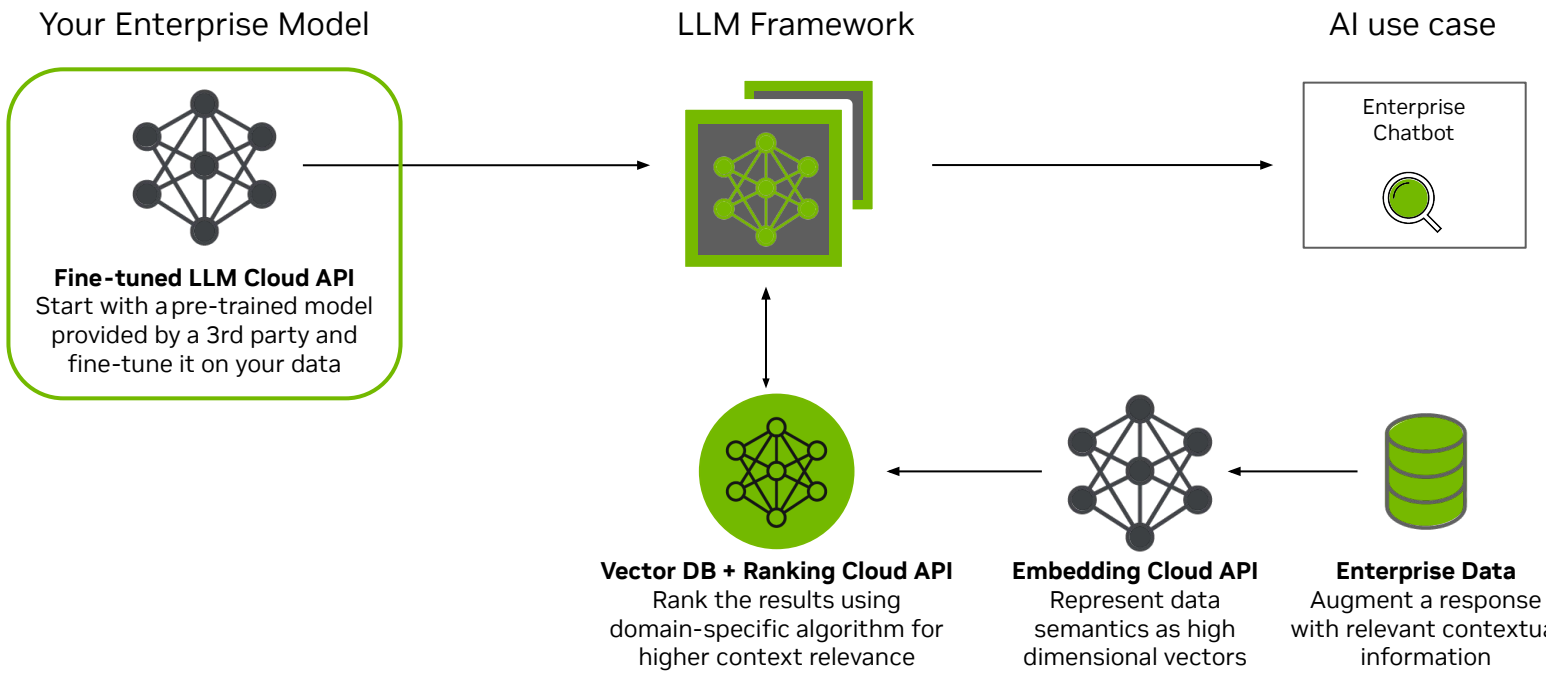
Use Retrieval-Augmented Generation (RAG)

Increase context relevance using domain-specific (re)ranking algorithm



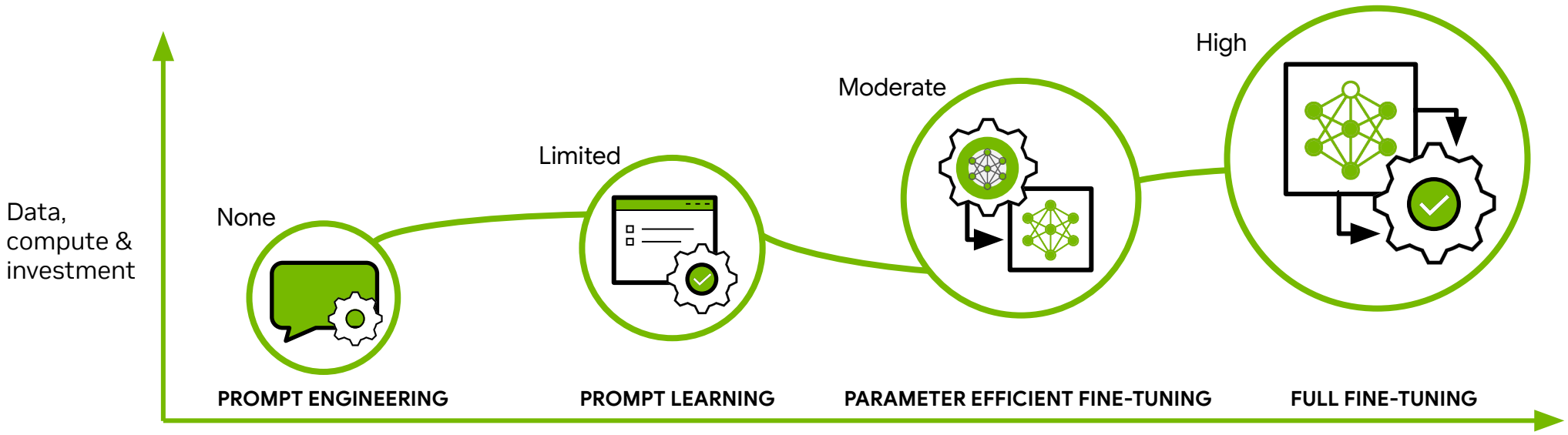
Fine-tune Your Model to Understand Domain Semantics

Increase LLM accuracy by customizing for your enterprise use case



LLM Customization Techniques for Each Scenario

Balancing accuracy, latency, and cost for optimal performance

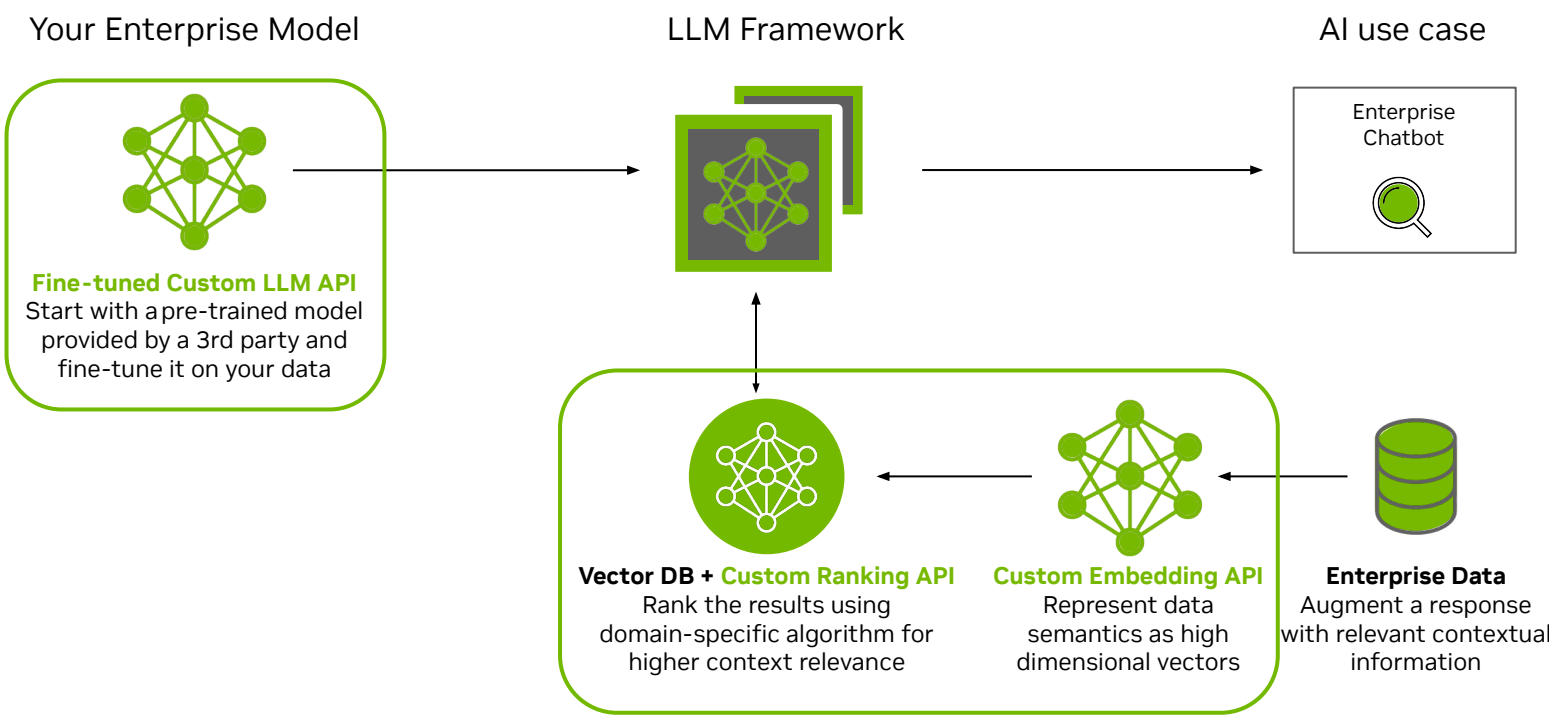


	PROMPT ENGINEERING	PROMPT LEARNING	PARAMETER EFFICIENT FINE-TUNING	FULL FINE-TUNING
Techniques	Few-shot learning, CoT reasoning, system prompting	Prompt tuning P-tuning	Adapters, LoRA IA3	SFT / SteerLM RLHF
Benefits	Good results leveraging pre-trained LLMs Lowest investment Least expertise	Better results leveraging pre-trained LLMs Lower investment Will not forget old skills	Best results leveraging pre-trained LLMs Will not forget old skills Add incremental knowledge	Best results leveraging pre-trained LLMs Change all model parameters Add domain-specific skills
Challenges	Cannot add as many skills or domain specific data to a pre-trained LLM	Less comprehensive ability to change all model parameters	Medium investment Takes longer to train More expertise needed	May forget old skills Large investment Most expertise needed

Adopt Open Source Models to Gain Flexibility and Control

Open source models (LLM, embedding, ranking) help protect enterprise data and IP

Falcon 40B
Gemma 2B
Gemma 7B
Llama-2 7B
Llama-2 13B
Llama-2 70B
Code Llama 34B
Mistral 7B
Mixtral 8x7B
Nemotron 8B
Nemotron 43B
GPT3 175B
MPT 30B



Foundation Model Provider's Compliance with the EU AI Act

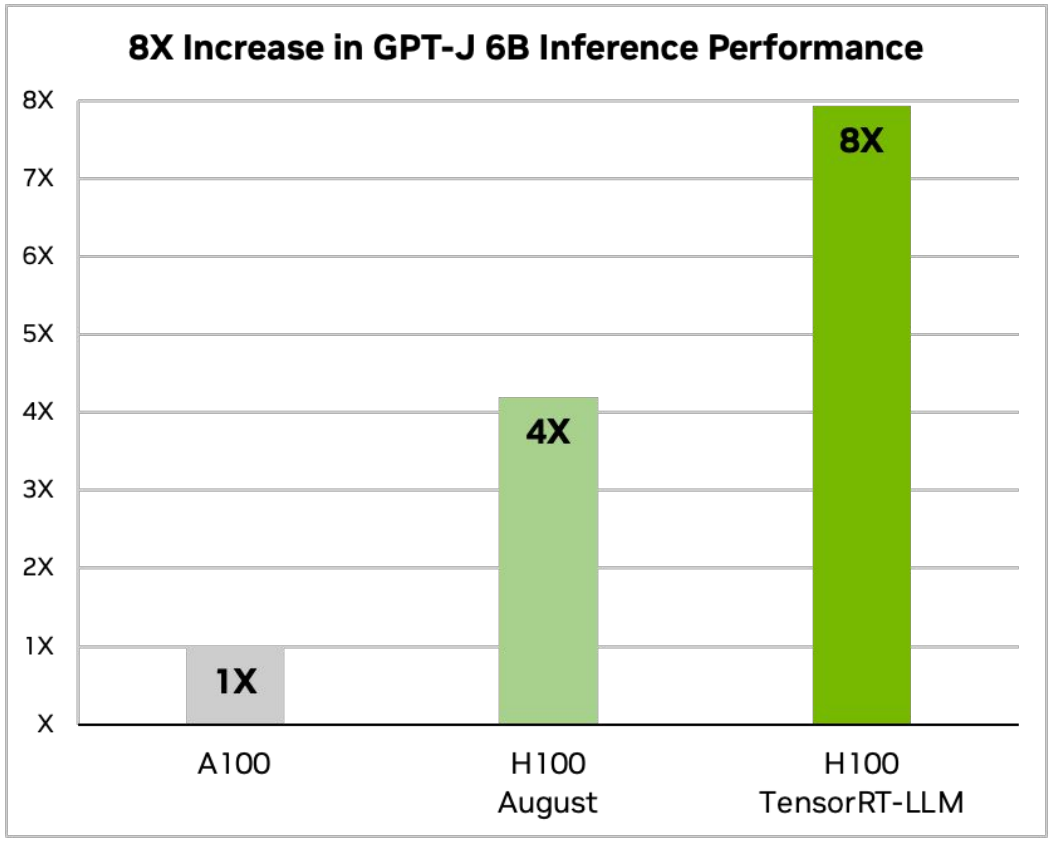
Assessment of providers across data, compute, model, and deployment scenarios (Stanford CRFM and HAI, June 2023)

Draft AI Act Requirements	GPT-4	Cohere Command	Stable Diffusion v2	Claude 1	PaLM 2	BLOOM	LLaMA	Jurassic-2	Luminous	GPT-NeoX	Totals
Data sources	● ○ ○ ○	● ● ● ○	● ● ● ●	○ ○ ○ ○	● ● ○ ○	● ● ● ●	● ● ● ●	○ ○ ○ ○	○ ○ ○ ○	● ● ● ●	22
Data governance	● ● ○ ○	● ● ● ○	● ● ○ ○	○ ○ ○ ○	● ● ● ○	● ● ● ●	● ● ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ● ● ○	19
Copyrighted data	○ ○ ○ ○	○ ○ ○ ○	○ ○ ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ● ● ○	○ ○ ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ● ● ●	7
Compute	○ ○ ○ ○	○ ○ ○ ○	● ● ● ●	○ ○ ○ ○	○ ○ ○ ○	● ● ● ●	● ● ● ●	○ ○ ○ ○	● ○ ○ ○	● ● ● ●	17
Energy	○ ○ ○ ○	● ○ ○ ○	● ● ● ○	○ ○ ○ ○	○ ○ ○ ○	● ● ● ●	● ● ● ●	○ ○ ○ ○	○ ○ ○ ○	● ● ● ●	16
Capabilities & limitations	● ● ● ●	● ● ● ○	● ● ● ●	● ○ ○ ○	● ● ● ●	● ● ● ○	● ● ○ ○	● ● ○ ○	● ○ ○ ○	● ● ● ○	27
Risks & mitigations	● ● ● ○	● ● ○ ○	● ○ ○ ○	● ○ ○ ○	● ● ● ○	● ● ○ ○	● ○ ○ ○	● ● ○ ○	○ ○ ○ ○	● ○ ○ ○	16
Evaluations	● ● ● ●	● ● ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ● ○ ○	● ● ● ○	● ● ○ ○	○ ○ ○ ○	● ○ ○ ○	● ○ ○ ○	15
Testing	● ● ● ○	● ● ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ● ○ ○	● ● ○ ○	○ ○ ○ ○	● ○ ○ ○	○ ○ ○ ○	○ ○ ○ ○	10
Machine-generated content	● ● ● ○	● ● ● ○	○ ○ ○ ○	● ● ● ○	● ● ● ○	● ● ● ○	○ ○ ○ ○	● ● ● ○	● ○ ○ ○	● ● ● ○	21
Member states	● ● ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ● ○ ○	● ● ● ●	○ ○ ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ○ ○ ○	● ● ○ ○	9
Downstream documentation	● ● ● ○	● ● ● ●	● ● ● ●	○ ○ ○ ○	● ● ● ●	● ● ● ●	● ● ○ ○	○ ○ ○ ○	○ ○ ○ ○	● ● ● ○	24
Totals	25 / 48	23 / 48	22 / 48	7 / 48	27 / 48	36 / 48	21 / 48	8 / 48	5 / 48	29 / 48	

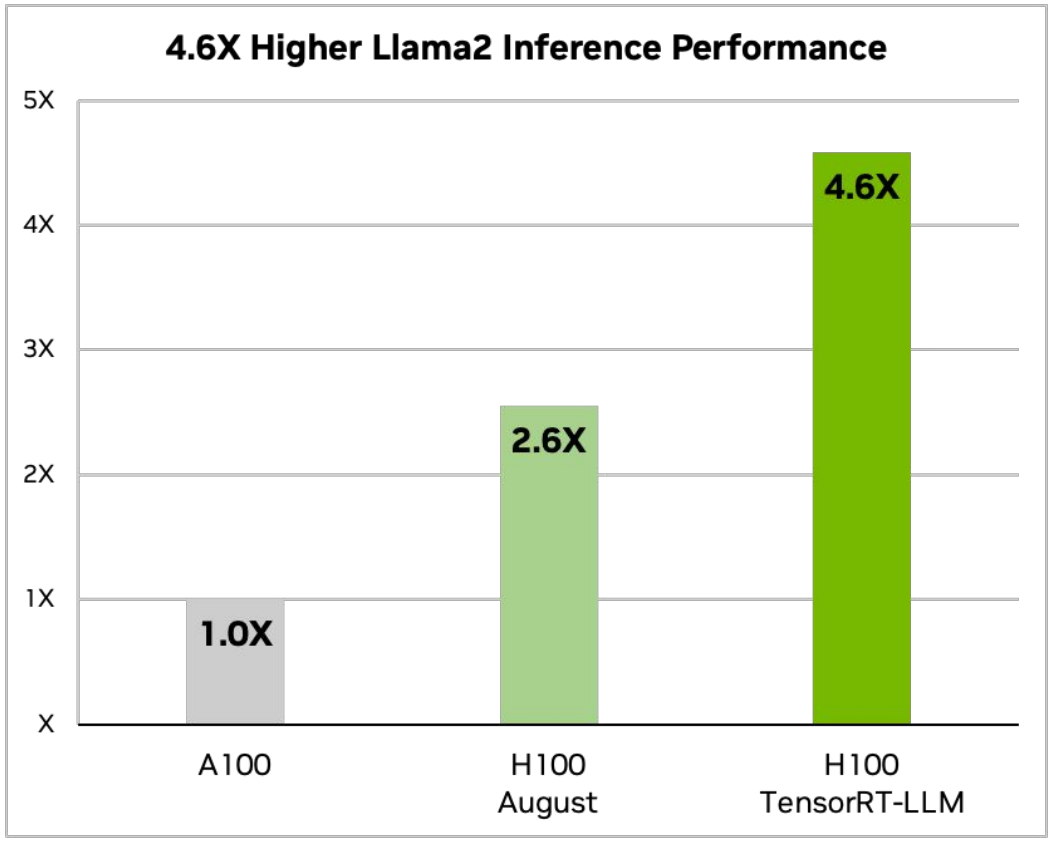
Source: Do Foundation Model Providers Comply with the Draft EU AI Act? (link)

HW & SW Co-optimization Yields the Best Results

Latency, TCO, and energy efficiency improvements with H100 and TensorRT-LLM



GPT-J-6B A100 compared to H100 with and w/o TensorRT-LLM

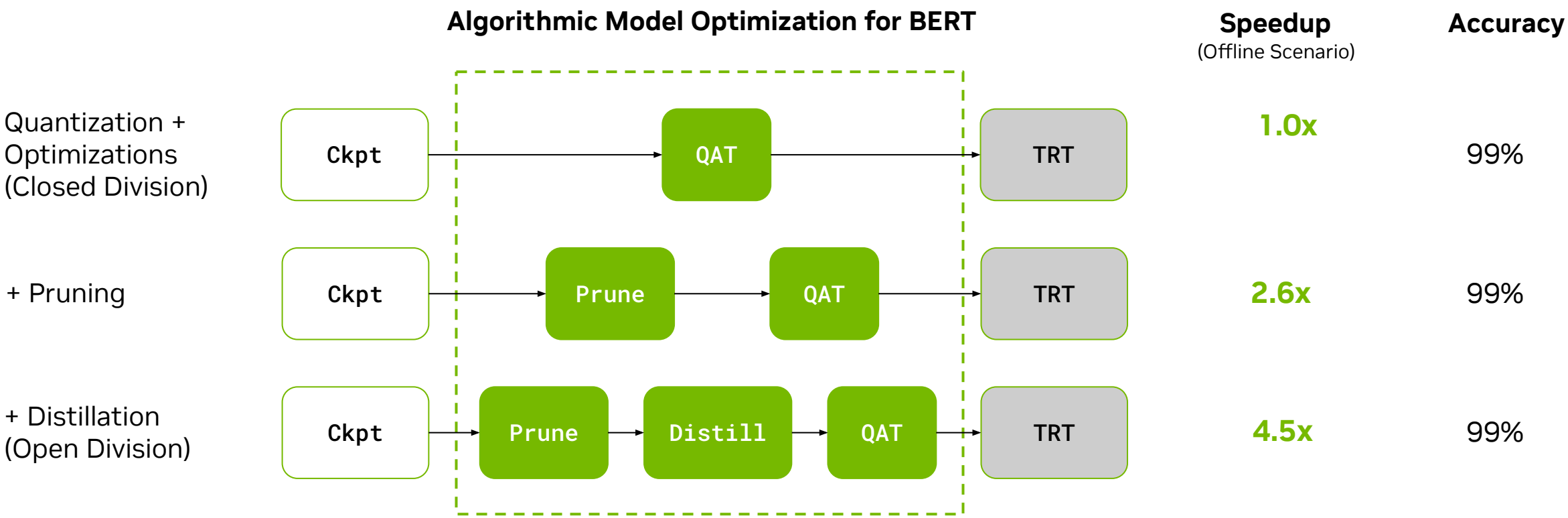


Llama-2 70B, A100 compared to H100 with and w/o TensorRT-LLM

Source: NVIDIA TensorRT-LLM Supercharges Large Language Model Inference on NVIDIA H100 GPUs (link)

HW & SW Co-optimization Yields the Best Results

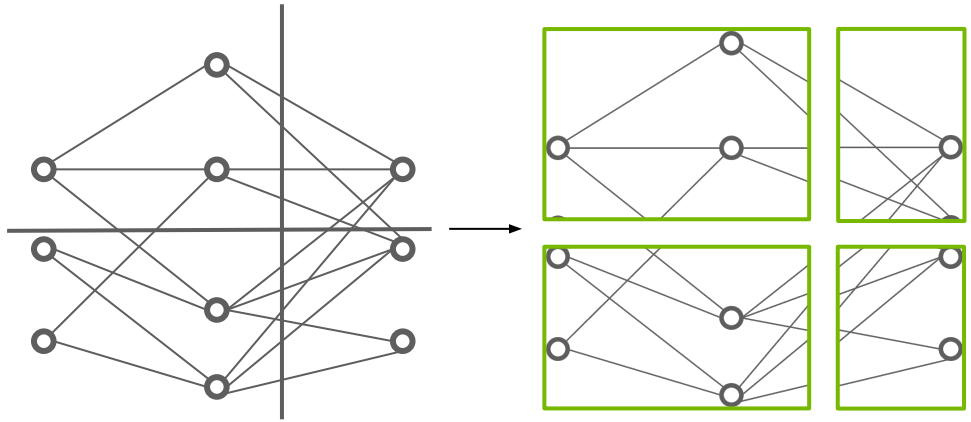
Stacking of model optimizations leads to 4x fewer params, 5.6x less FLOPs, and 3.4x smaller TensorRT-LLM engine size



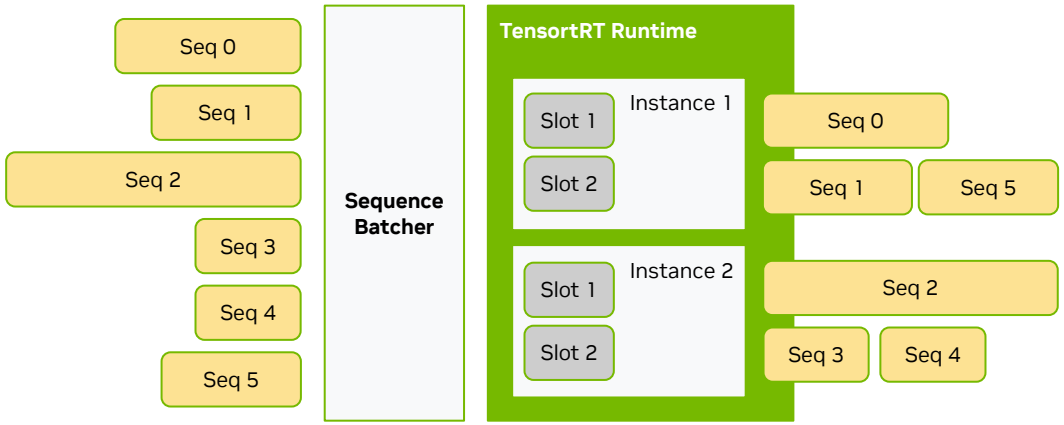
Source: Leading MLPerf Inference v3.1 Results with NVIDIA GH200 Grace Hopper Superchip Debut (link)

HW & SW Co-optimization Yields the Best Results

Model parallelism and in-flight batching maximize throughput and GPU utilization



Tensor Parallelism (TP) and **Pipeline Parallelism (PP)** allow inference of larger LLMs optimally across many GPUs



In-flight batching (IFB) yields high request throughput in settings with lots of traffic by optimally scheduling inputs to GPUs

NVIDIA NIM Streamlines the Path to Production

Easiest and most performant way to deploy generative AI and LLM models coupled with industry-standard APIs

Prebuilt container and helm chart tested and validated across infrastructure

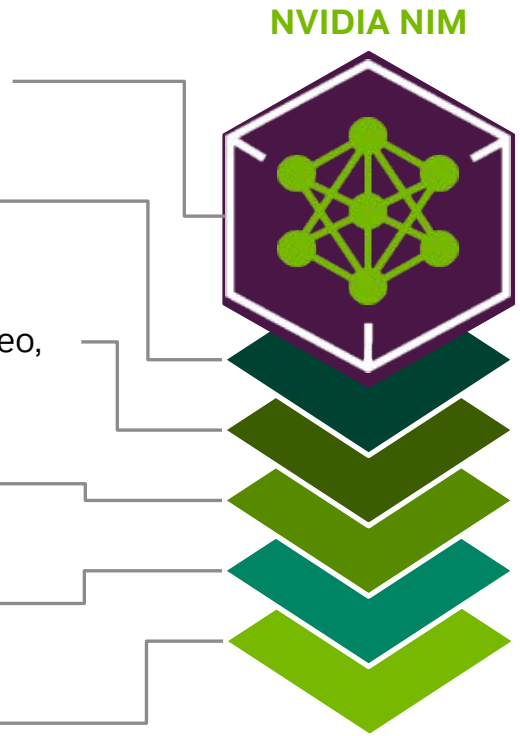
Industry standard APIs with NVIDIA Unified Cloud Standards

Domain specific code for each NIM domain category, including LLMs, Images, VLMs, video, healthcare, biology, genomics, and more

Optimized inference engines for each model and hardware SKU

Support for custom models built by users targeted use cases

NVIDIA AI Enterprise approved base container



Deploy anywhere and maintain control of generative AI applications and data

Simplified development of AI application that can run in enterprise environments

Day 0 support for state-of-the-art generative AI models providing choice across the ecosystem

Improved TCO with best latency and throughput running on accelerated infrastructure

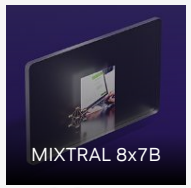
Best accuracy for enterprise by enabling tuning with proprietary data sources

Enterprise software with feature branches, validation and support

NVIDIA NIM Streamlines the Path to Production

Fastest way to deploy AI models on any accelerated infrastructure across cloud, data center, and PC

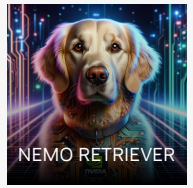
NVIDIA API Catalog (build.nvidia.com)



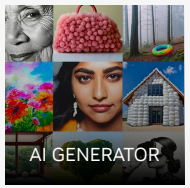
MISTRAL AI_

Google

ADEPT



NVIDIA



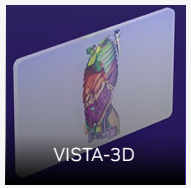
gettyimages

Microsoft

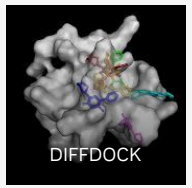
shutterstock

NVIDIA

Meta



NVIDIA



MIT

NVIDIA

NVIDIA NIM

Microsoft Azure

aws

Google Cloud

ORACLE

DELL Technologies

Hewlett Packard Enterprise

Lenovo

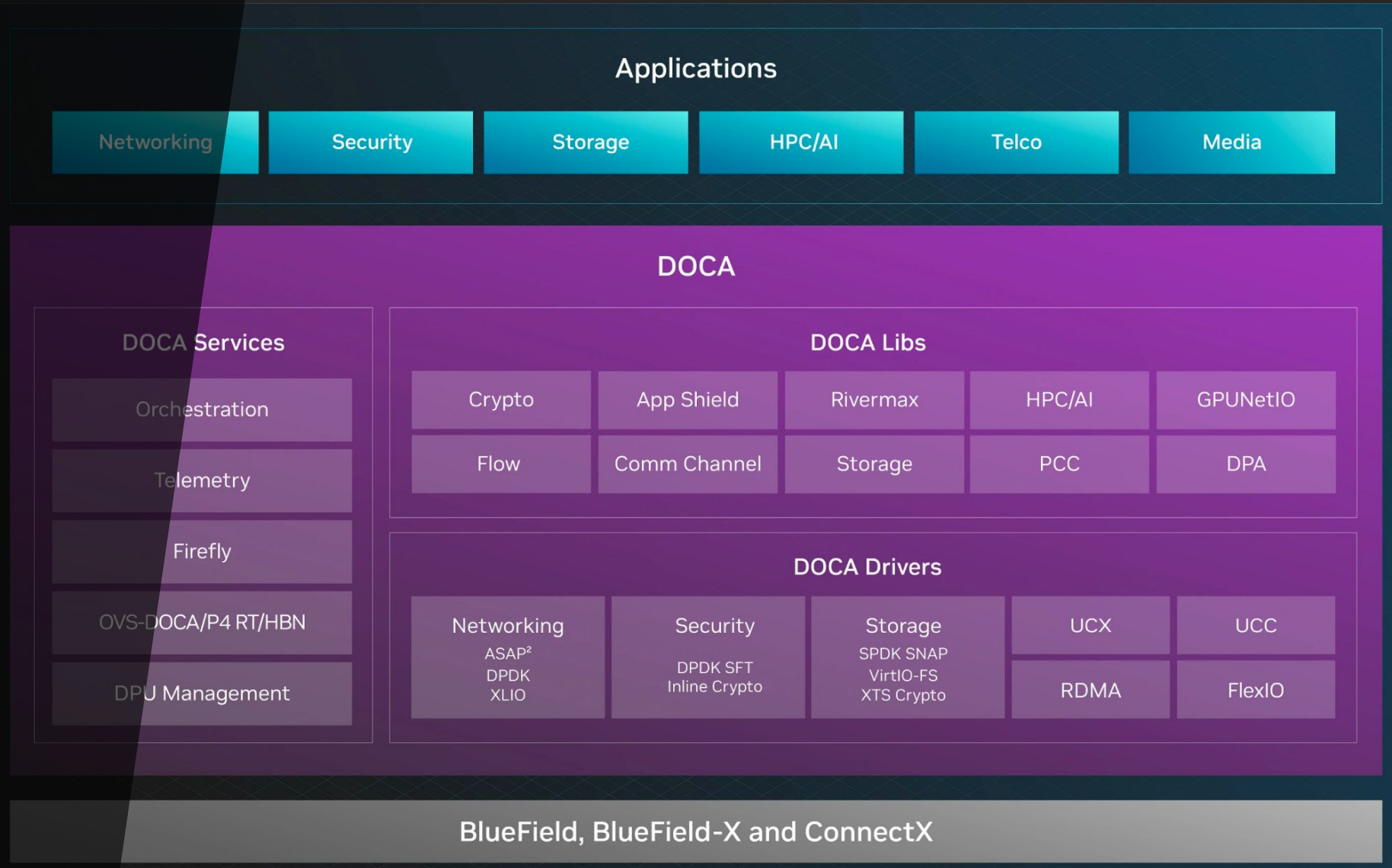
SUPERMICRO

LLM Assistant for DOCA SDK

The NVIDIA DOCA™ SDK enables developers to rapidly create applications and services on top of NVIDIA® hardware and networking platform, leveraging industry-standard APIs.

Today, the chatbot generates C/C++ simple application using DOCA GPUNetIO SDK based on user prompt and summarizes DOCA info taken from the online DOCA documentation.

It may help developers to familiarize with the SDK and create generic application layout for basic tasks that can be reapplied to the real use cases.



Summarization and Q&A for DOCA SDK

Code Generation for DOCA SDK in C/C++

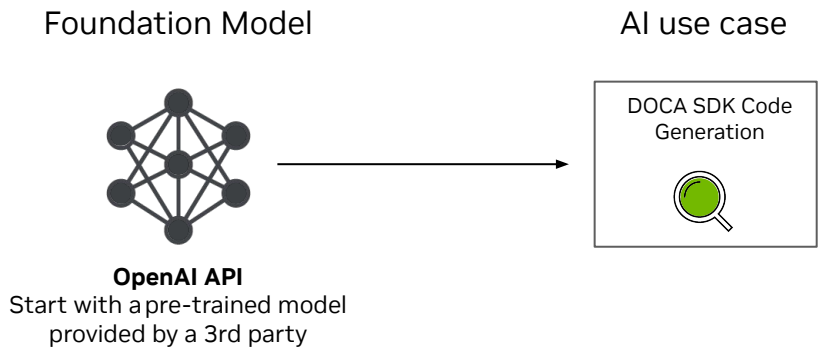
Business KPIs for DOCA SDK Assistant

Optimizing time-to-MVP and ability to quickly proof value and feasibility

Accuracy	Percentage of DOCA source code, which is **built correctly** Percentage of **functionally correct instruction flows**
Latency	N/A
Cost	**Engineering effort** (time it takes to build an app) **Engineering level** (skills required to build an app) **Development** (total initial $ investment on Cloud LLM API) **Operations** ($ per interaction with a Cloud LLM API)

Use an LLM Cloud API for an MVP

Building an MVP is easy and doesn't require any specialized skills



Prompt Engineering with ChatGPT

DOCA is a new SDK making it challenging for LLM Cloud APIs generate working code

ChatGPT

As of my last update in January 2022, DOCA SDK stands for "Data Center Infrastructure Developer Kit." It is a software development kit developed by NVIDIA. DOCA SDK is designed to provide developers with tools and libraries to build and optimize software for networking, storage, and security applications in data center infrastructure, particularly in environments utilizing NVIDIA's data processing units (DPUs).

DOCA SDK offers APIs, libraries, and tools to enable developers to efficiently utilize the capabilities of NVIDIA's DPUs for tasks such as accelerating networking, enhancing storage performance, and implementing security features. This SDK is intended to empower developers to create high-performance, scalable, and efficient software solutions tailored for modern data center infrastructure needs.

It's worth noting that developments may have occurred since my last update, so for the most current and detailed information about DOCA SDK, I recommend visiting NVIDIA's official website or consulting their documentation.

Here's a very basic and hypothetical example to give you a rough idea:

cpp — Copy code

```cpp
#include <doca_sdk/gpu.h>
#include <doca_sdk/gpunetio.h>

int main() {
    // Initialize DOCA SDK
    doca_sdk::init();

    // Initialize GPU device
    doca_sdk::gpu::Device device = doca_sdk::gpu::getDevice(0); // Assuming GPU 0
    device.initialize();

    // Initialize GPUNetIO
    doca_sdk::gpunetio::NetIO netIO(device);
    netIO.initialize();

    // Perform data transfer operations...

    // Cleanup
    netIO.shutdown();
    device.shutdown();

    // Shutdown DOCA SDK
    doca_sdk::shutdown();

    return 0;
}
```

This code is highly simplified and serves only as a conceptual example. You should consult the official documentation and API reference of DOCA SDK for the actual functions, data structures, and usage patterns required to work with GPUNetIO and GPU devices in your C++ application.

Selecting the Solution That is More Likely to Meet Business KPIs

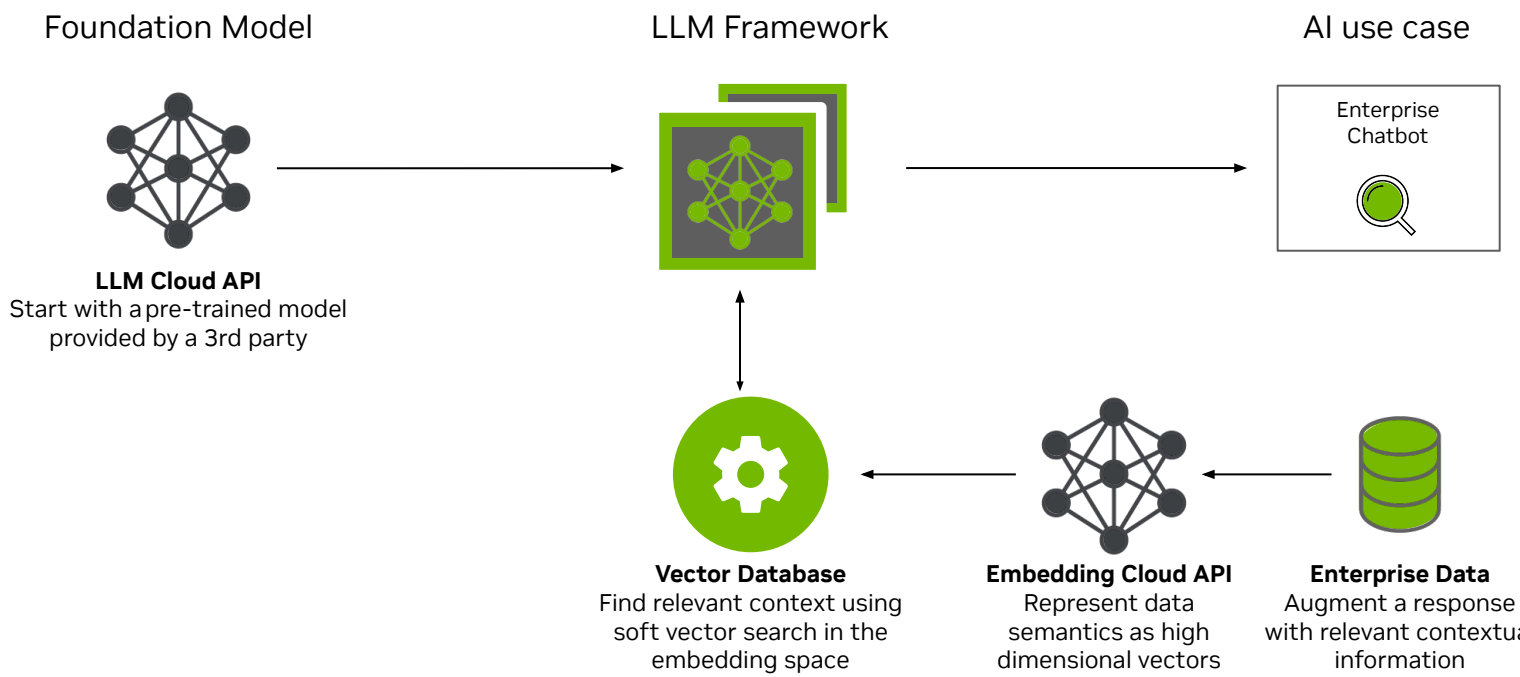
Accuracy vs. Time-to-market and engineering effort/complexity

Business KPIs \ Solutions	Fine-tuned LLM Cloud API	LLM Cloud API + Cloud Embedding API (RAG)
Accuracy	Can reach **very high accuracy** with a well defined dataset	**Accuracy may vary** depending on the quality of the documentation, source code comments, etc
Flexibility & Extensibility	Medium / High	Low / Medium
Engineering effort	Need to produce **more DOCA samples** Requires an **huge dataset** for a great accuracy in the result Can't avoid a **great human effort** for meaningful Q/A Dataset needs to be **maintained every new DOCA releases**	DOCA public **source code and documentation already available** for embeddings Can be tested with several LLM with **minimal effort** **Easily reproducible** for every DOCA release Does **not need any human** intervention
Engineering level	Low / Medium	Low / Medium
Development cost	Medium/High for each fine-tuning round	Low for embeddings creation
Operational cost	Low / Medium	Low / Medium

[S62163] The Goldilocks Approach to LLMs: Balancing Accuracy, Latency, and Cost by Janaki Vamaraju, Elena Agostini, Nik Spirin

Use Retrieval-Augmented Generation (RAG)

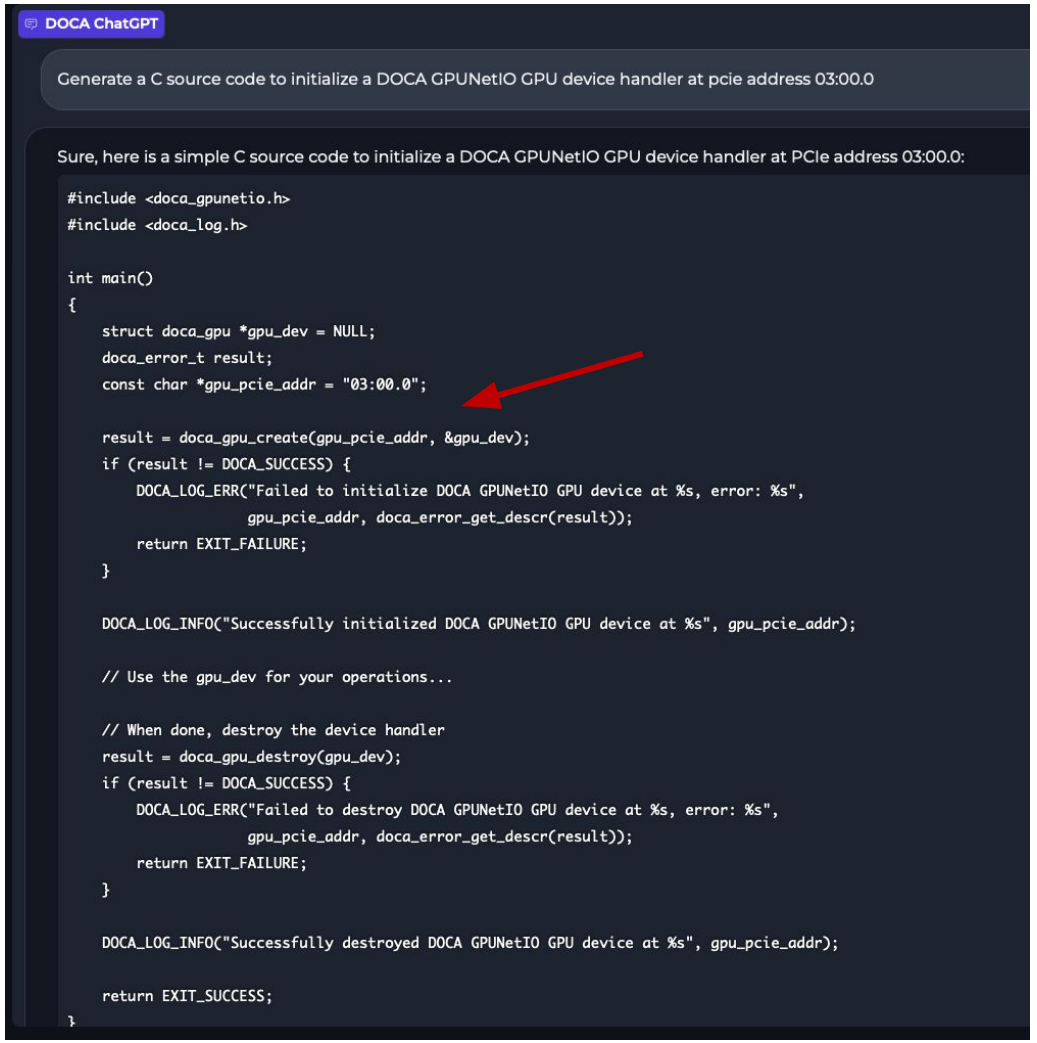
Represent DOCA SDK docs as embeddings, the use vector similarity search to surface relevant context



Retrieval-Augmented Generation (RAG)

Simple scenario

- Simple queries worked well, can build generated code
  - Reusable layout for different type of applications
- Accuracy:
  - **Build**: 100%
    - This code compiles with no errors
  - **Execute**: 66%
    - 3 functions needed, 1 missing
- Missing DPDK function not included in DOCA SDK
  - Function role not clear in DOCA applications code
  - Mentioned only once in DOCA documentation



Retrieval-Augmented Generation (RAG)

Complex scenario

- Query: Generate a DOCA GPUNetIO C source code application to initialize and destroy an handler for the GPU at PCIe address 03:00.0. Additionally the application should allocate GPU memory.

- Good layout for a more complex scenario

- Accuracy:
  - **Build**: ~95% ↓
    - 1 error for 22 lines of code
  - **Execute**: 80% ↑
    - 5 functions needed, 1 missing

- Same problem as previous sample
  - Difficult to fix it with RAG, easier with fine-tuning

```c
#include <stdio.h>
#include "doca_gpu.h"

int main() {
    struct doca_gpu *gpu_dev;
    const char *pcie_address = "03:00.0";

    // Initialize the DOCA GPUNetIO GPU device handler
    doca_error_t error = doca_gpu_create(pcie_address, &gpu_dev);
    if (error != DOCA_SUCCESS) {
        printf("Failed to create DOCA GPUNetIO GPU device handler. Error code: %d\n", error);
        return 1;
    }

    // Allocate GPU memory
    size_t mem_size = 1024; // Allocate 1KB of memory
    void *mem_gpu, *mem_cpu;
    error = doca_gpu_mem_alloc(gpu_dev, mem_size, 0, DOCA_GPU_MEM_TYPE_GPU_CPU, &mem_gpu, &mem_cpu);
    if (error != DOCA_SUCCESS) {
        printf("Failed to allocate GPU memory. Error code: %d\n", error);
        doca_gpu_destroy(gpu_dev);
        return 1;
    }

    // Use the allocated memory for further operations
    // ...

    // Free the allocated GPU memory
    doca_gpu_mem_free(gpu_dev, mem_gpu);

    // Destroy the DOCA GPUNetIO GPU device handler
    doca_gpu_destroy(gpu_dev);

    return 0;
}
```

LLM Assistant for Chip Design - ChipNeMo

AI **copilot** built by NVIDIA research to assist one of the most complex engineering efforts, designing semiconductors.

Responds to **questions about GPU architecture and design** while helping engineers quickly find technical documents in early tests. It will also **create snippets of about 10-20 lines of software** in two specialized languages chip designers use, making it easier to develop new code.

Using proprietary data to customize a foundation model, researchers found that a much **smaller 13B parameter model could out preform larger general purpose LLMs.**

Q&A for GPU ASIC Architecture

Bug Analysis & Reports

Code Generation for VLSI Tools

Business KPIs for ChipNeMo LLM Assistant

Three chip design use cases: EDA Code Generation, Bug Summarization, Design-assist Chatbot

Accuracy

Correctness on wide range of **domain-specific tasks**

Avoid security risks with third party APIs

Model groundedness in the chip domain (e.g. retrieval hit-rate)

Latency

Fast batch evaluation on domain-specific benchmarks

Real-time responses for NVIDIA engineers

Cost

Development (GPU training time, number of data samples and pre-training tokens)

Operations (reduced inference cost at scale)

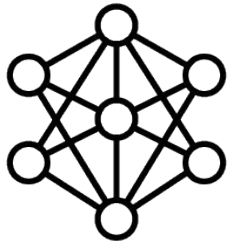
End-2-end ChipNeMo Customization Workflow

Domain-specific models lead to higher accuracy and lower cost

Pretraining

Trillions of tokens of internet data

10^5-10^6 GPU hrs

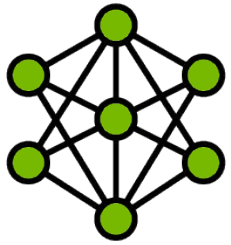


Foundation Model
Llama-2
(70B, 13B, 7B)

Domain Adaptive Pretraining

24B tokens of chip design docs/code

~5000 GPU hrs

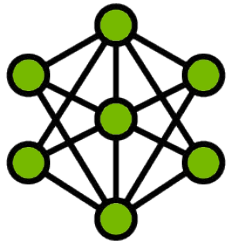


ChipNeMo Foundation Model
(13B, 7B)

Supervised Fine-Tuning

128K chat instructions + 1.1K task instructions

~100 GPU hrs



ChipNeMo Chat Model
(13B, 7B)

Pre-trained Models | Training and Customization | Deployment

ChipNeMo Data Curation

Balanced datasets combining NVIDIA-proprietary chip design specific data and publicly available datasets

Data Source Type	Data Percentage (%)	Data Tokens (B)	Training Percentage (%)	Training Tokens (B)
Bug Summary	9.5%	2.4	10.0%	2.4
Design Source	47.0%	11.9	24.5%	5.9
Documentation	17.8%	4.5	34.0%	8.2
Verification	9.1%	2.3	10.4%	2.5
Other	7.9%	2.0	12.0%	2.9
Wikipedia	5.9%	1.5	6.2%	1.5
Github	2.8%	0.7	3.0%	0.7
Total	100.0%	25.3	100.0%	24.1

Breakdown of DAPT data for ChipNeMo after filtering (**24.1 billion tokens**)

Domain Source	Number of Samples
Design Knowledge	280
EDA Script Generation	480
Bug summarization and analysis	392
Total	1152

Breakdown of Domain SFT data (**128000 samples**)

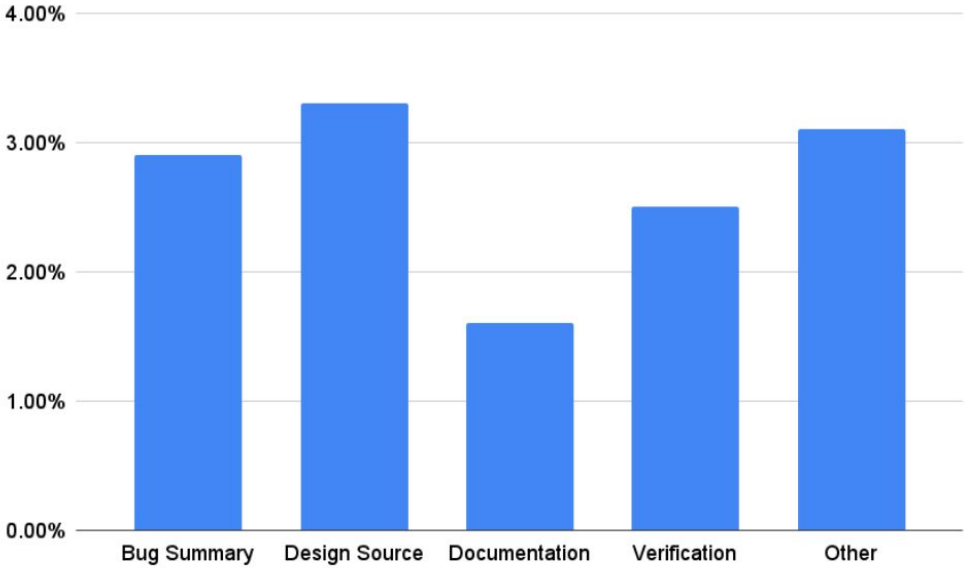
- ✔ Data relevance and quality > quantity
- ✔ Data anonymization and privacy should be considered in dataset compilation
- ✔ Continuous data updating process critical to keep the training set relevant
- ✔ Data curation & management play important role

[S62163] The Goldilocks Approach to LLMs: Balancing Accuracy, Latency, and Cost by Janaki Vamaraju, Elena Agostini, Nik Spirin

Domain-adaptive Foundation Model Pretraining

Custom Tokenization

ChipNeMo's tokenizer enhancements (**9k new tokens**) improved tokenization efficiency (**1.6% to 3.3% improvement**) across various design datasets without significant accuracy decline on public benchmarks



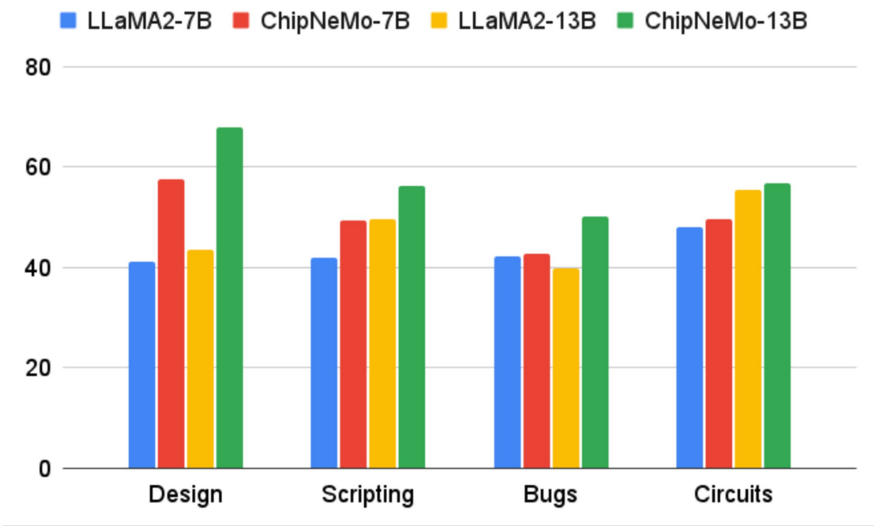
ChipNeMo Tokenizer Augmentation Improvements

- ✔ Balance between generic language understanding and domain-specific nuances
- ✔ Iterative refinement of tokenizer based on feedback and model performance
- ✔ Collaboration between domain experts and ML engineers to identify critical tokens

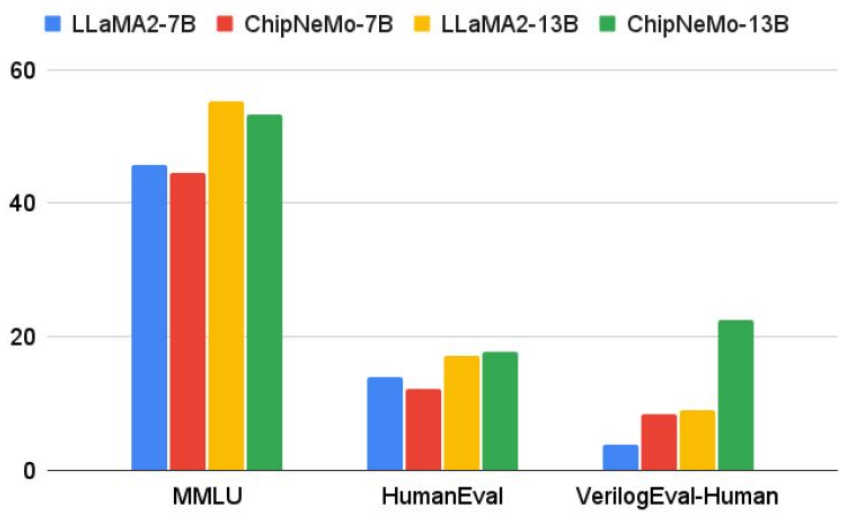
[S62163] The Goldilocks Approach to LLMs: Balancing Accuracy, Latency, and Cost by Janaki Vamaraju, Elena Agostini, Nik Spirin

Domain-adaptive Foundation Model Pretraining

ChipNeMo uses domain-adaptive pre-training to better understand chip design contexts



Chip Design Domain Benchmarks



Academic Benchmarks

- ✔ Balancing between continuing pre-training and overfitting risks. Smaller learning rate plays a dual role.
- ✔ Larger and more performant foundational models yielded better zero shot results on domain-specific tasks.
- ✔ In-domain Improvements from DAPT positively correlated with model size

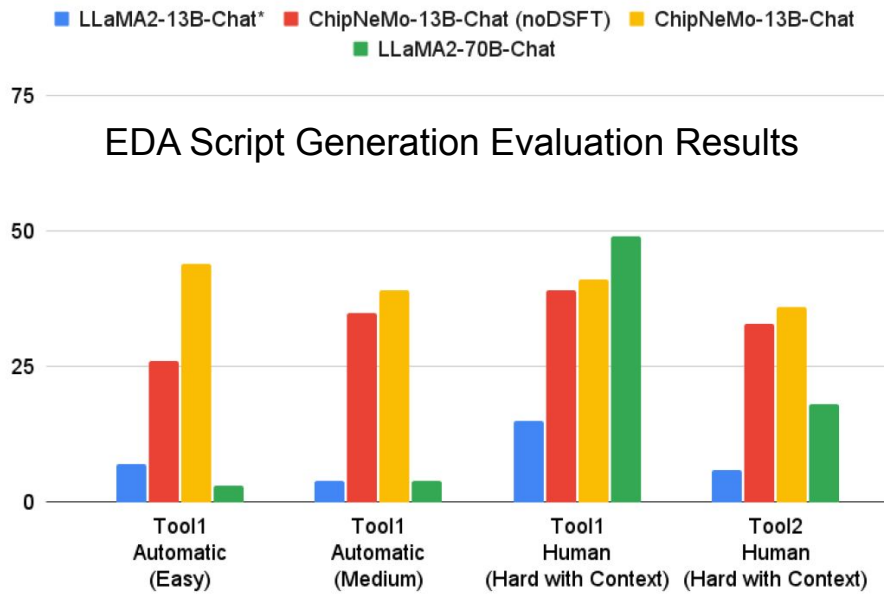
[S62163] The Goldilocks Approach to LLMs: Balancing Accuracy, Latency, and Cost by Janaki Vamaraju, Elena Agostini, Nik Spirin

Supervised Fine-Tuning

Customization of model behaviour for high performance on specific tasks

Model Size	Pretraining	DAPT	SFT
7B	184,320	2,620	90
13B	368,640	4,940	160
70B	1,720,320	-	-

Training cost of LLaMA2 models in GPU hours.



ChipNeMo-Chat: Models fine-tuned with both domain and general chat data
ChipNeMo-Chat (noDSFT): Models fine-tuned with general chat data exclusively.

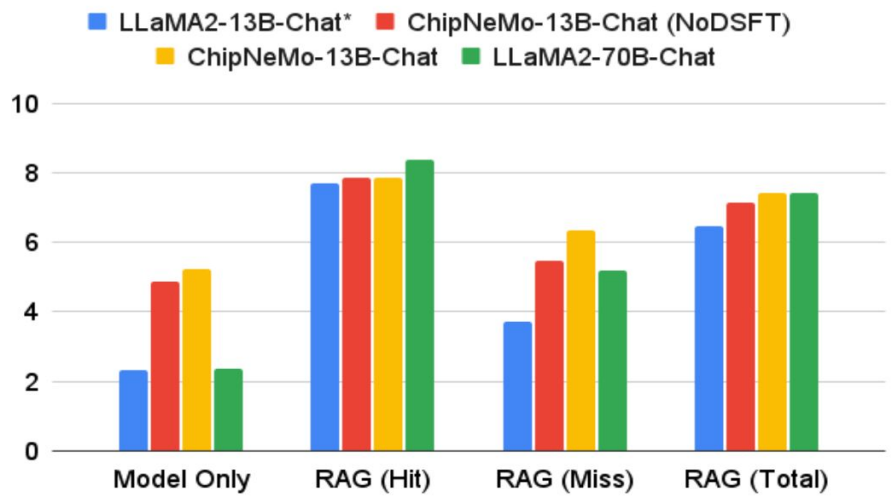
- ✔ Importance of quality and relevance of labeled data for fine-tuning
- ✔ Adopt techniques for efficient fine-tuning without compromising model generalizability
- ✔ Evaluation metrics tailored to specific tasks to gauge SFT success

Engineering chatbot assistant: ⬆️ 0.33 out of 10 point scale
EDA script generation: ⬆️ 18% correctness
Bug summarization: ⬆️ 0.79 out of 7 point scale

[S62163] The Goldilocks Approach to LLMs: Balancing Accuracy, Latency, and Cost by Janaki Vamaraju, Elena Agostini, Nik Spirin

Retrieval-Augmented Generation (RAG)

Fine-tuning ChipNeMo retrieval model + domain-specific data improves the hit rate by 30% leading to better RAG



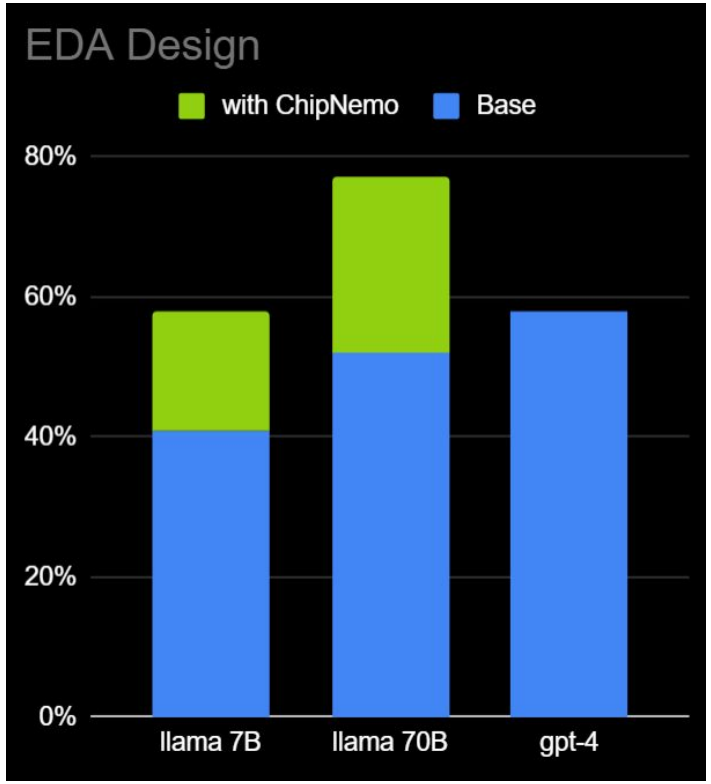
Human Evaluation of Different Models

- ✔ Addition of in-domain context through **RAG** significantly boosts human scores
- ✔ **ChipNeMo (DAPT+SFT)** models outperforms fine-tuned same size LLaMa chat model.
- ✔ ChipNeMo-13b-Chat with RAG achieves same score as the 5X larger model LLaMA2-70B-Chat with RAG. **Domain adaptation however makes up for the misses.**
- ✔ Domain SFT improves performance of ChipNeMo-13B-Chat with/without RAG.

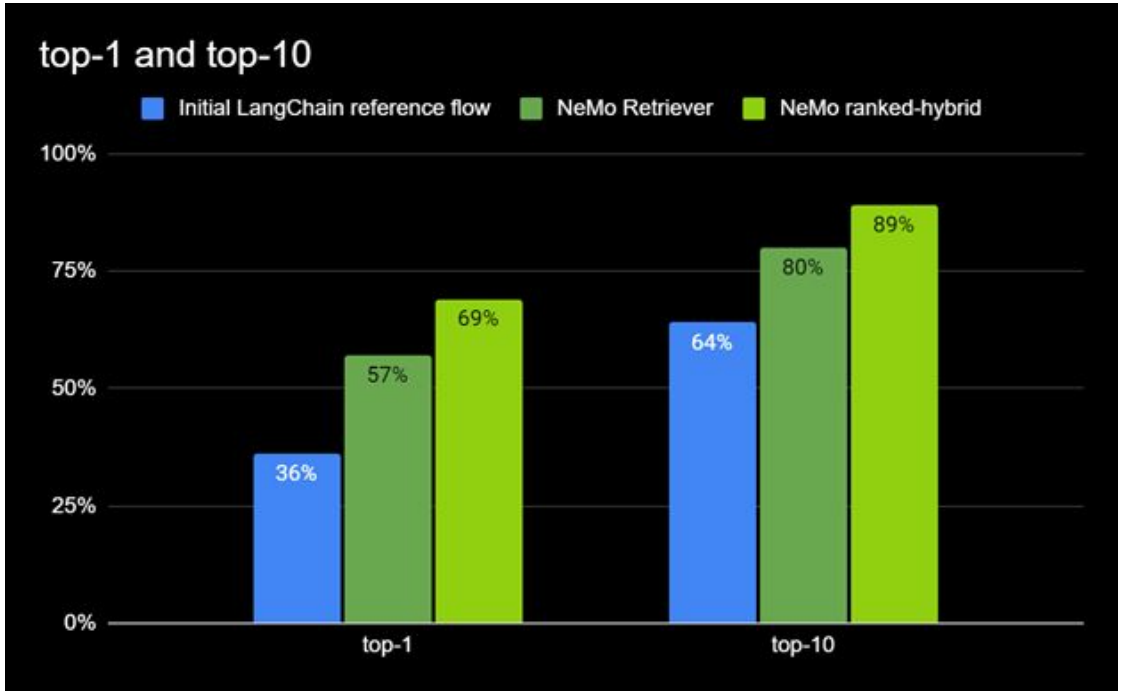
[S62163] The Goldilocks Approach to LLMs: Balancing Accuracy, Latency, and Cost by Janaki Vamaraju, Elena Agostini, Nik Spirin

Customization Lead to Large Performance Improvement

Domain-adapted ChipNeMo significantly outperforms OOTB solutions



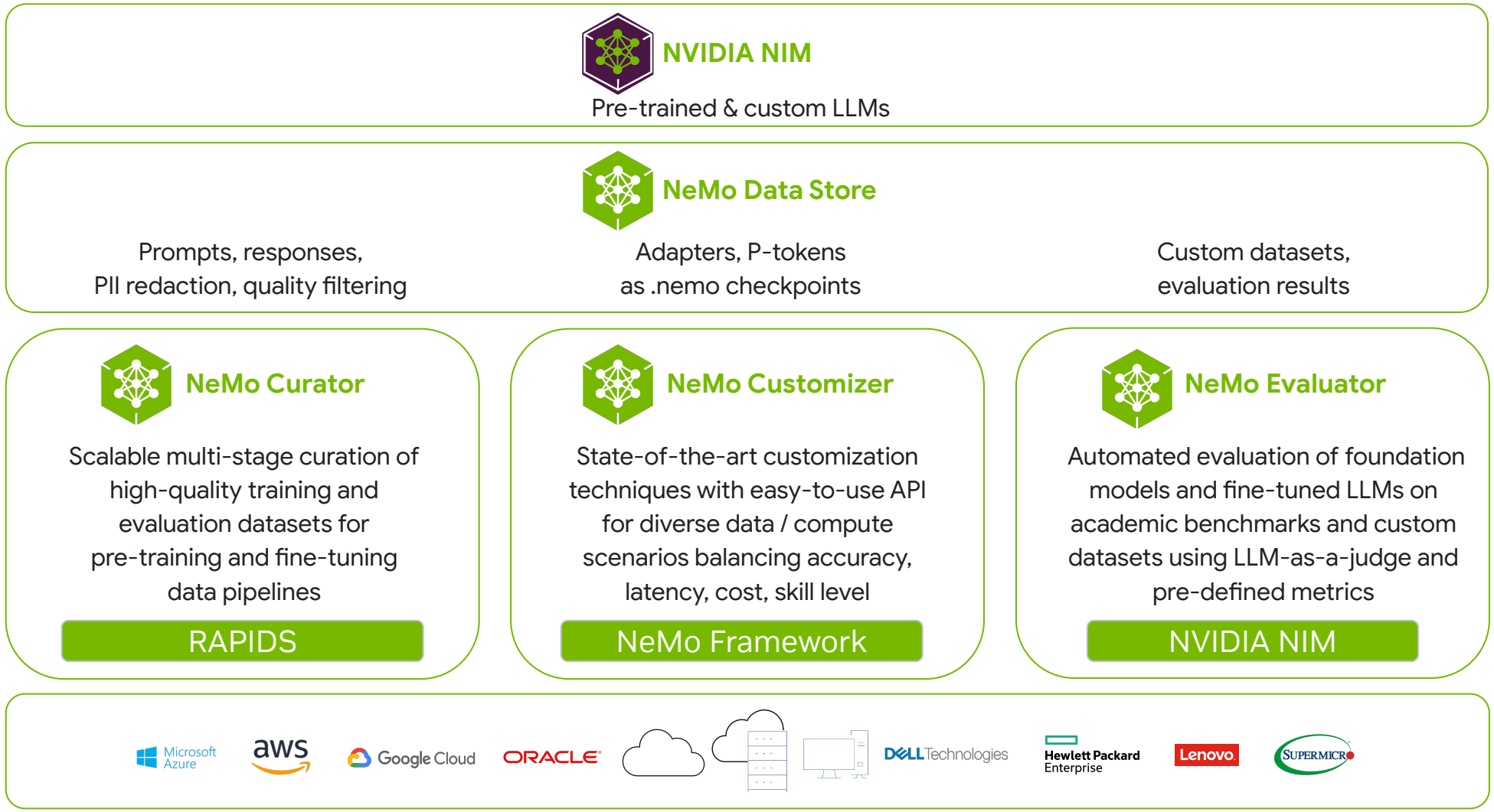
Customized Llama-2 7B achieves GPT-4 accuracy, while Llama-2 70B demonstrates state-of-the-art results



Domain-specific embedding, ranking, and re-ranking models lead to higher context relevance resulting in more downstream customer value

Unified Stack to Accelerate Generative AI Adoption for Enterprises

Enabling end-2-end generative AI journey from data curation to model customization, optimization, evaluation, inference



Q&A / Contacts / Next Steps

Don’t Miss This Transformative Moment in AI

Janaki Vamaraju
Deep Learning Architect and Scientist
NVIDIA
jvamaraju@nvidia.com

Elena Agostini
Senior Software Engineer
NVIDIA
eagostini@nvidia.com

Nik Spirin
Director of Generative AI and LLMOps
NVIDIA
nspirin@nvidia.com

Related GTC 2024 Talks from NVIDIA and Partners:

- **[S62458]** LLMOps: The New Frontier of Machine Learning Operations
- **[S61934]** Enterprise MLOps 101
- **[S62797]** LLM Inference Sizing: Benchmarking End-to-End Inference Systems
- **[S61190]** The Small Models Revolution
- **[S63103]** Considerations for Choosing LLM Serving Technologies (Presented by Run:ai)
- **[S62251]** Harmony in Diversity: Decoupled Values Alignment of Large Language Models
- **[S61604]** Optimizing AI-powered NPCs Cost-Efficiency Using TRT-LLM Without Sacrificing Quality
- **[DLIW62596]** Efficient Large Language Model (LLM) Customization
- **[S61775]** Optimizing and Scaling LLMs With TensorRT-LLM for Text Generation