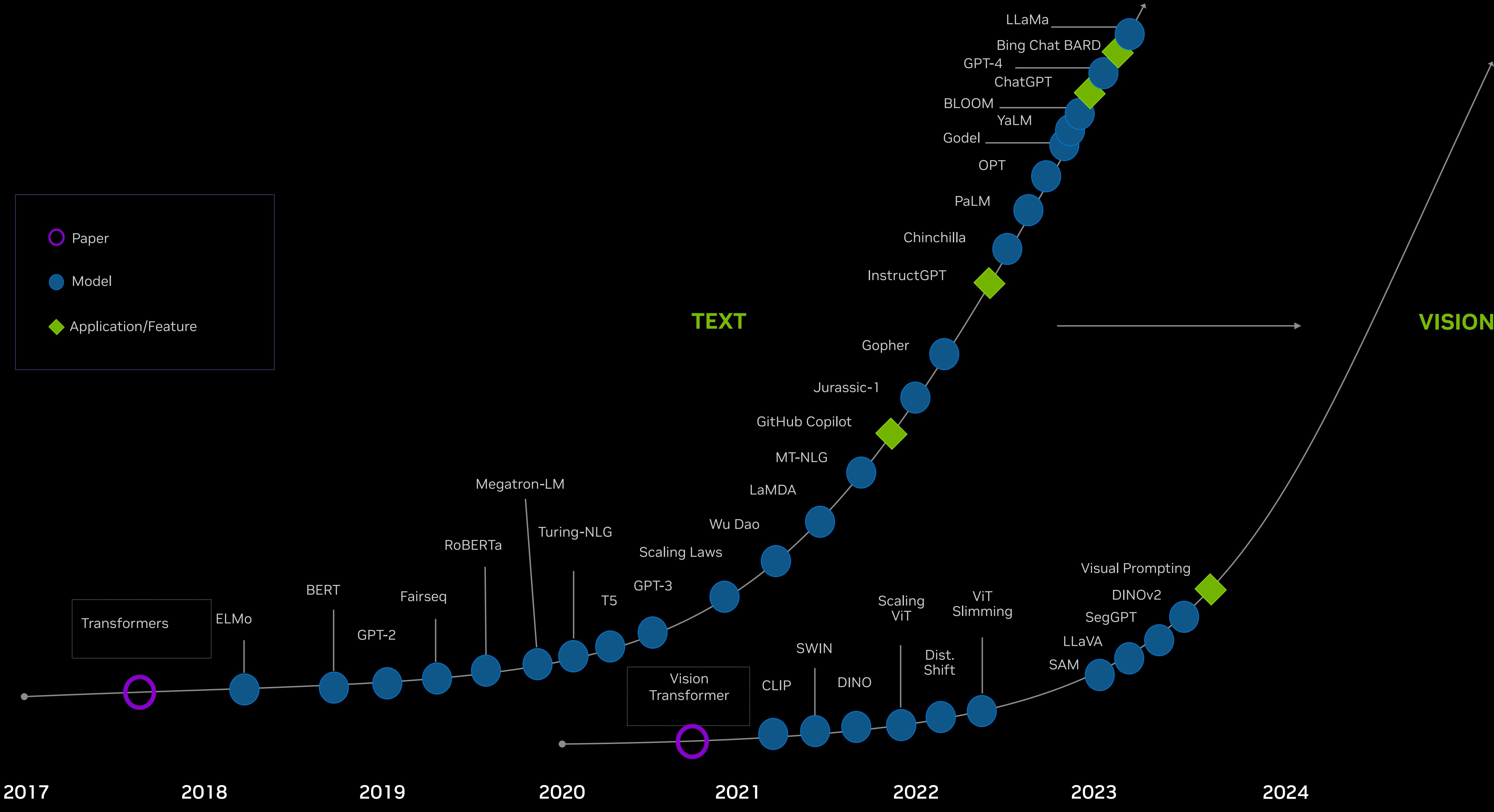




# Bringing Generative AI to Life with NVIDIA Jetson

Dustin Franklin, Principal Engineer | November 7, 2023

# AI Transformed



2017

2018

2019

2020

2021

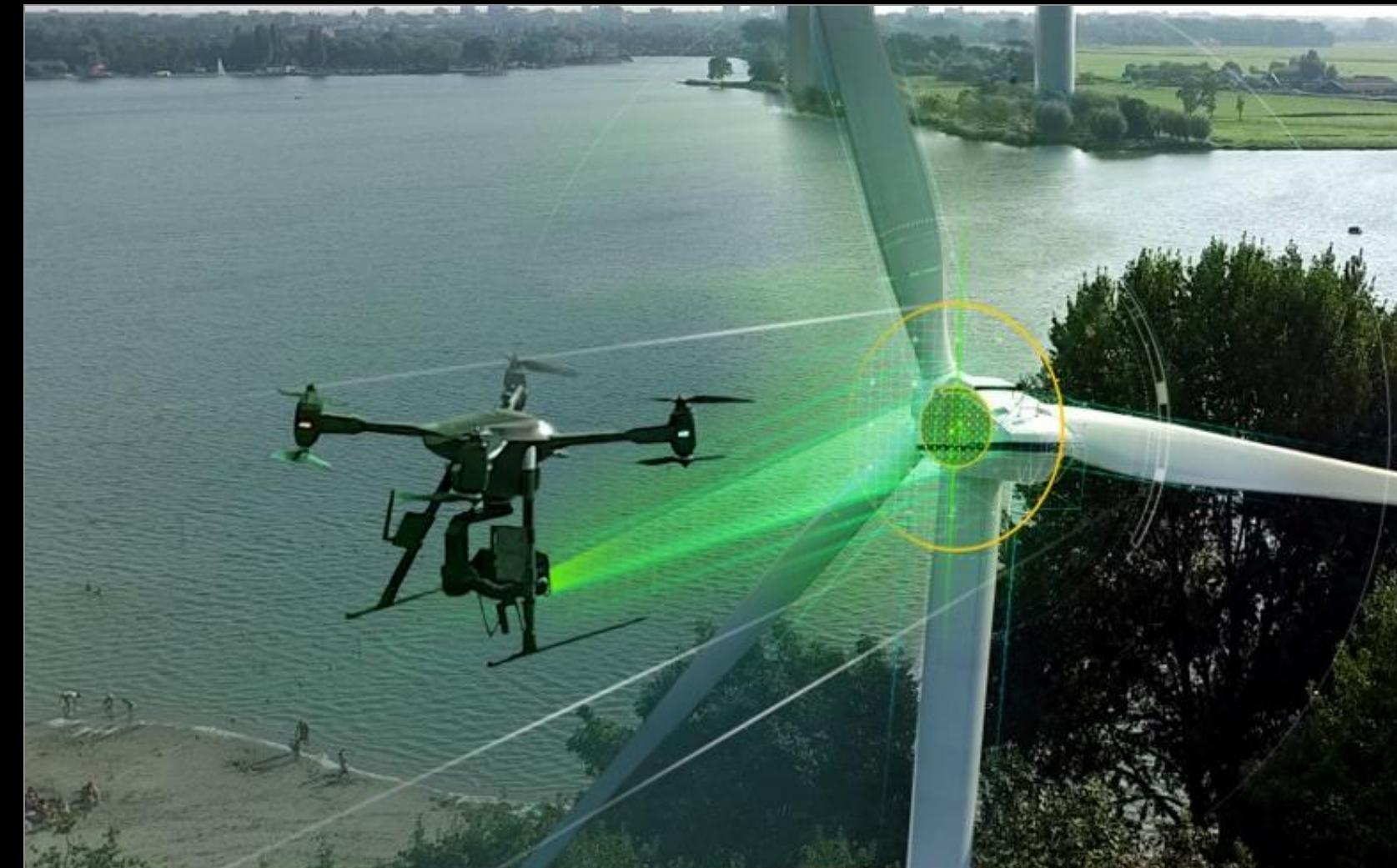
2022

2023

2024

Credit: Andrew Ng

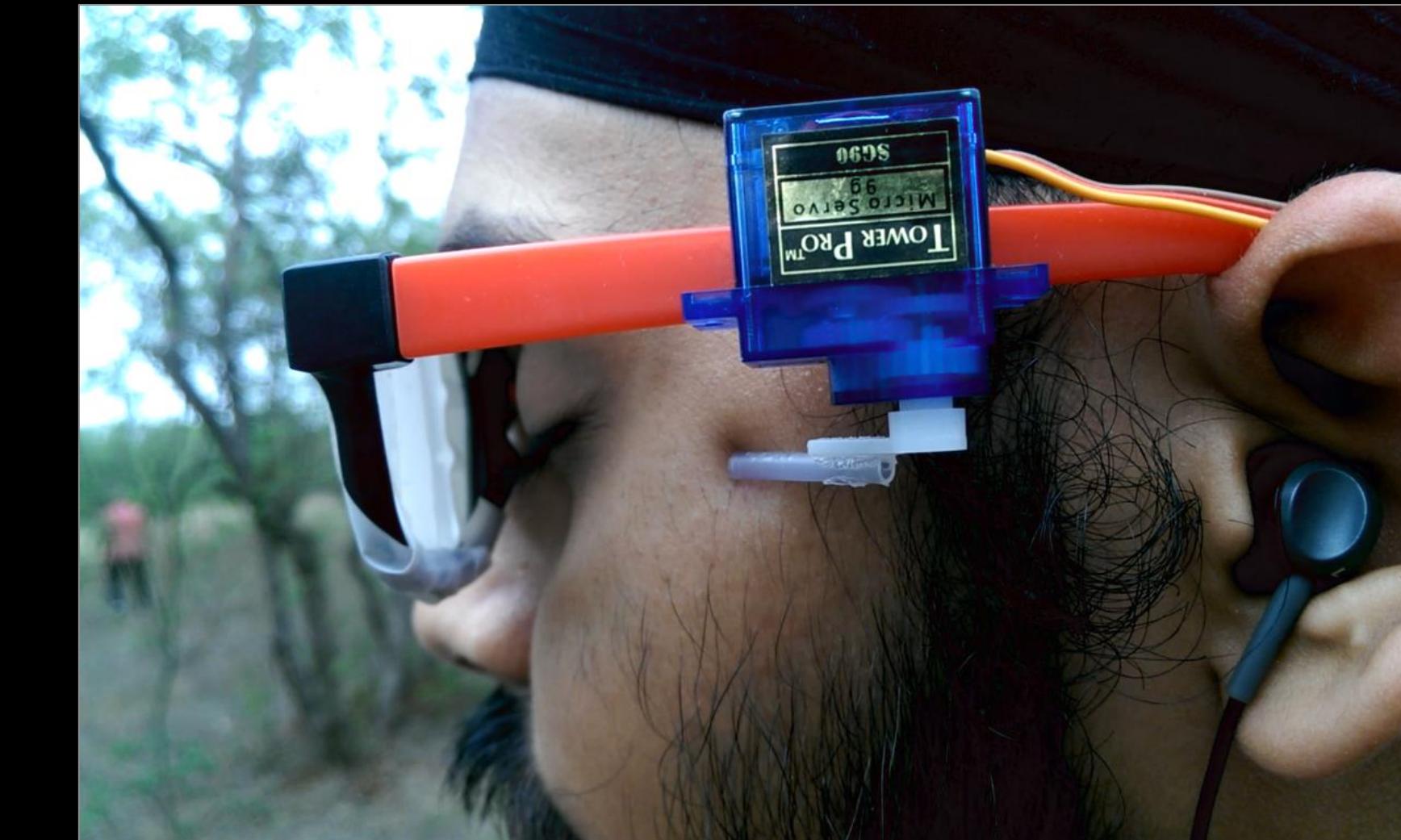
# Generative AI at the Edge



Smart Infrastructure



Smart Home



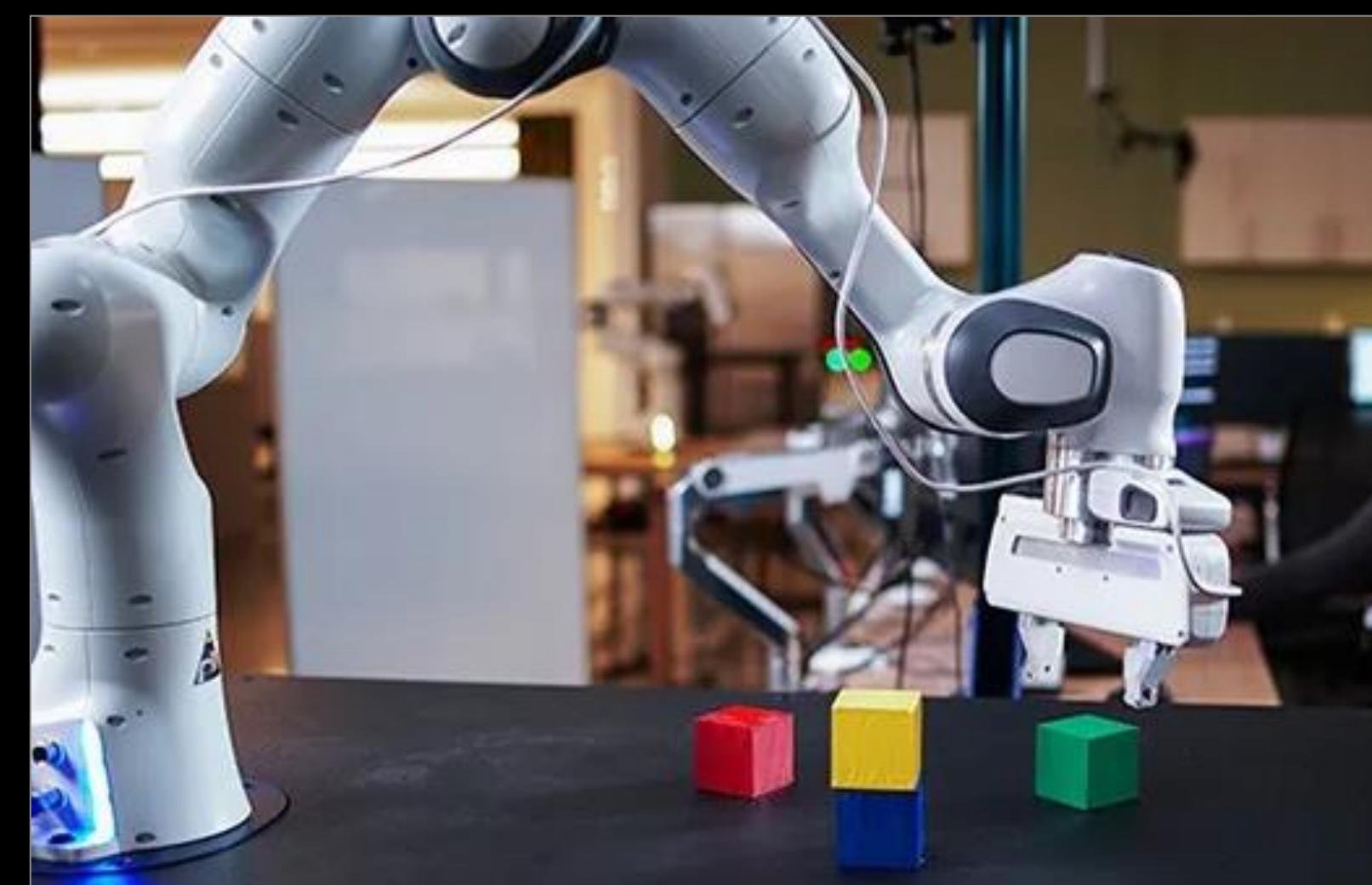
Assistive Devices



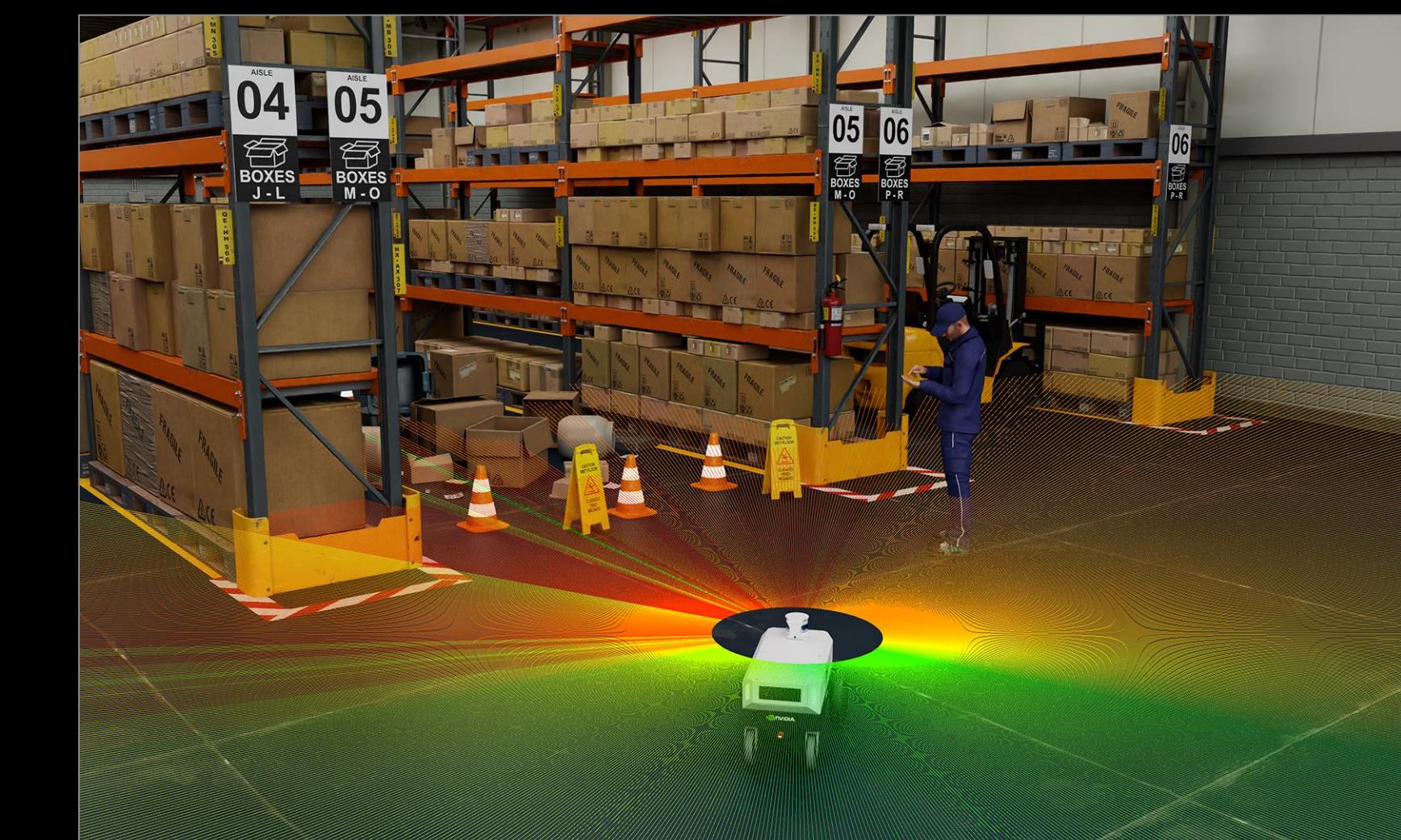
Precision Agriculture



Human Robot Interaction



Embodied Control & Skills



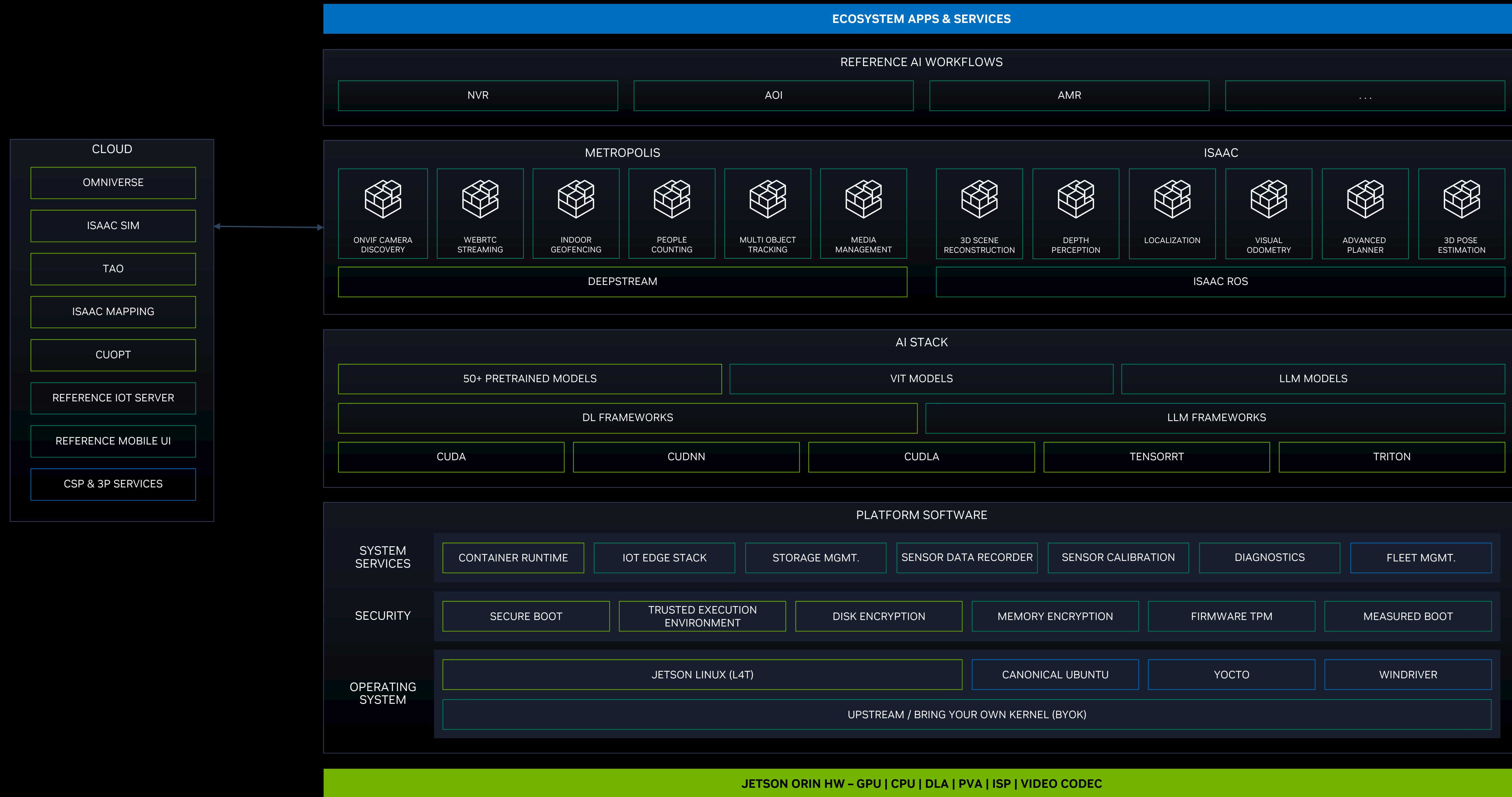
Autonomous Planning & Navigation



Video Queries

# Foundations of Edge AI

## JetPack 6.0



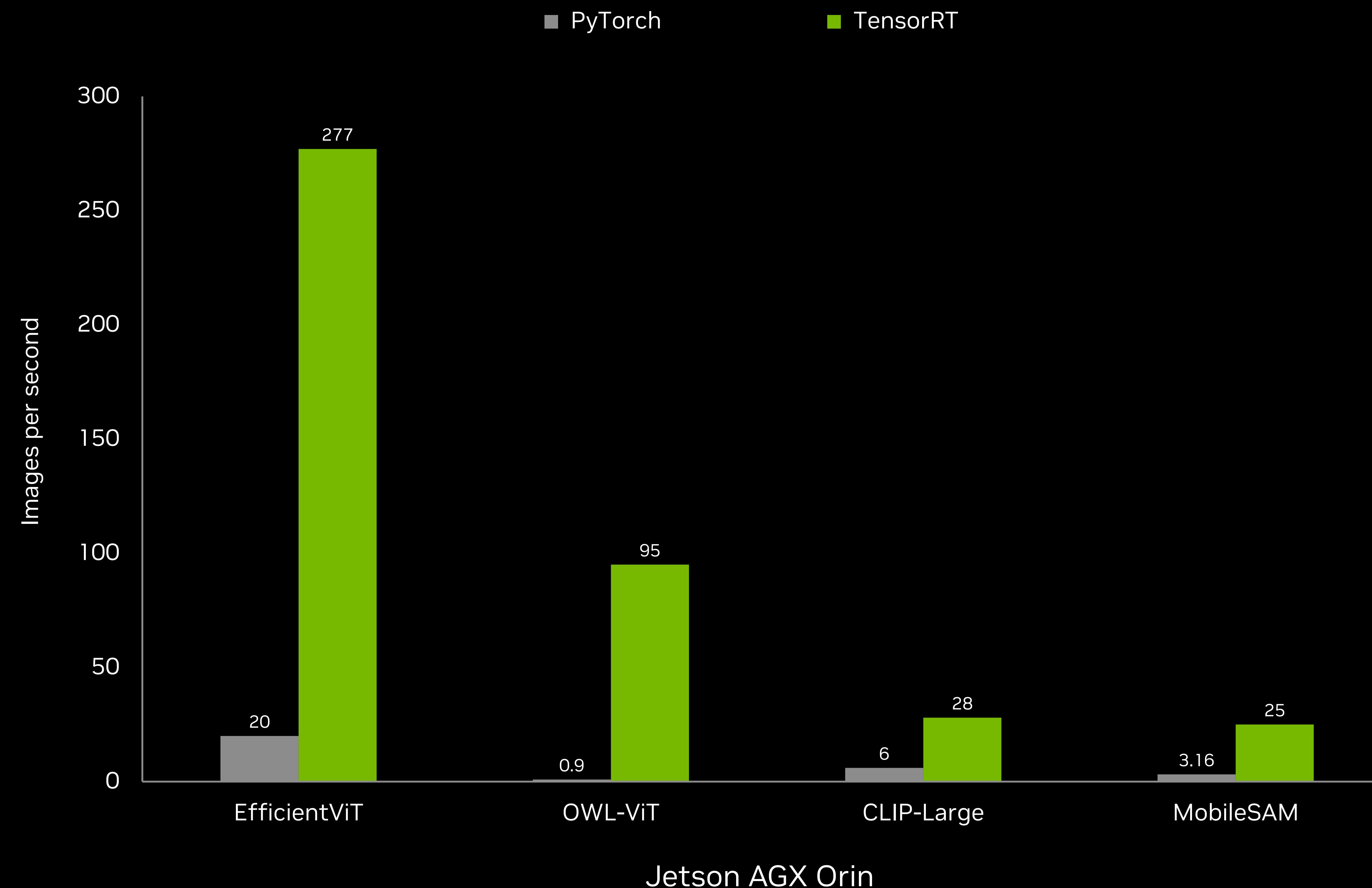


# Agenda

- Vision Transformers
- Large Language Models
- Embedding Vector Databases
- ASR/TTS with Riva

# Vision Transformers

## TensorRT



# Detect Anything

OWL-ViT



A person, a cat, ...

# Query Trees

OWL + CLIP



[a person [a face (happy, sad)]]

# NanoOWL

[github.com/NVIDIA-AI-IOT/nanoowl](https://github.com/NVIDIA-AI-IOT/nanoowl)

- “Simple Open-Vocabulary Object Detection with Vision Transformers” – Minderer et al.
- Optimized with TensorRT
- CLIP for secondary classification

	Image Size	Patch Size	FPS	Accuracy
OWL-ViT B/32	768	32	95	28.1
OWL-ViT B/16	768	16	25	31.7

\* FPS - Jetson AGX Orin

\* Accuracy (mAP) - MS COCO

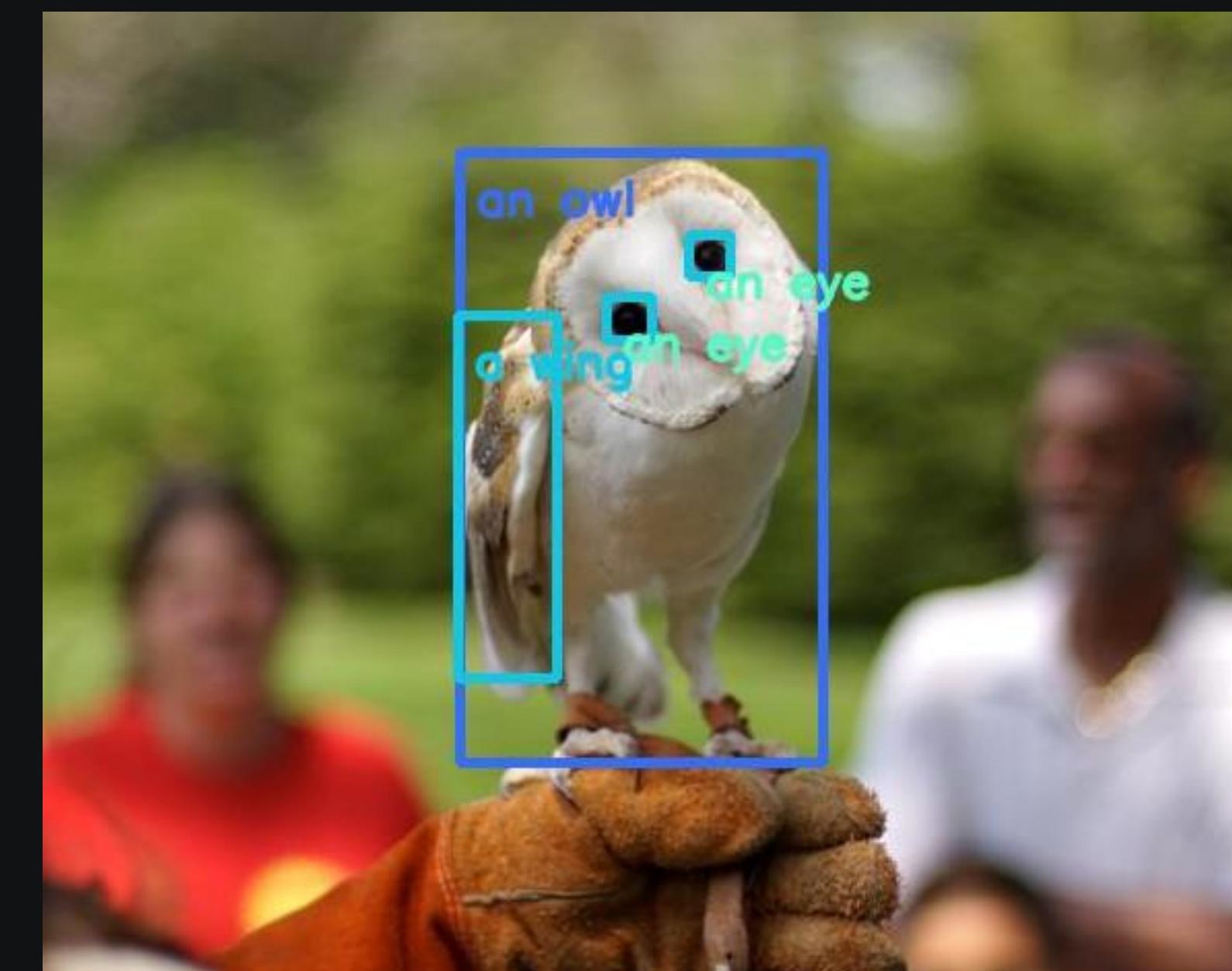
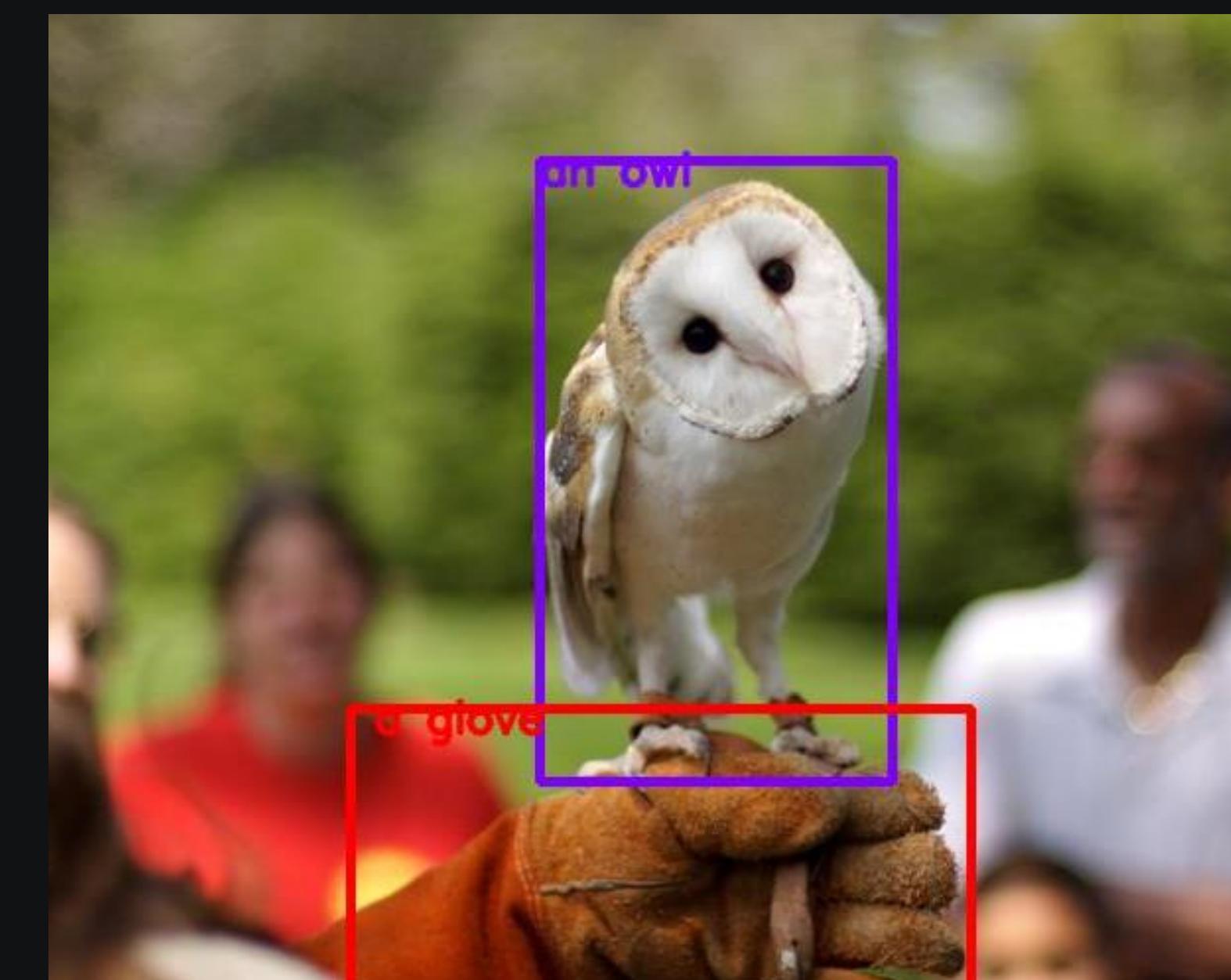
```
from nanoowl.owl_predictor import OwlPredictor

predictor = OwlPredictor(
    "google/owlvit-base-patch32",
    image_encoder_engine="owlvit-encoder.engine"
)

image = PIL.Image.open("owl.jpg")

output = predictor.predict(
    image=image,
    text=["an owl", "a glove"],
    threshold=0.1
)

print(output.labels, output.boxes, output.scores)
```



# Segment Anything

SAM + Tracking



# Getting Started

**NVIDIA Jetson Generative AI Lab**

Home Tutorials Benchmarks Community Articles Try

## Generative AI at the Edge

Bring generative AI to the world with NVIDIA® Jetson™

Explore Tutorials Walkthrough

### Check out our tutorials

- Text Generation**  
Run LLM-based chat bot on Jetson
- Text + Vision**  
Run multimodal Vision-Language models to give your AI access to vision
- Image Generation**  
Run diffusion models to generate stunning images interactively on Jetson
- Distillation**  
Learn a technique to bring the power of a large foundation model to Jetson by knowledge distillation
- NanoSAM**  
SAM (Segment Anything Model) and other Vision Transformers optimized to run in realtime
- NanoDB**  
Multimodal vector database that uses embeddings for txt2img and img2img similarity search

Jetson Generative AI Lab  
[jetson-ai-lab.com](http://jetson-ai-lab.com)

dusty-nv / jetson-containers

Type ⌘ to search

Code Issues 161 Pull requests 14 Discussions Actions Projects Wiki Security Insights Settings

jetson-containers Public

Unpin Unwatch 29 Fork 308 Star 1.2k

### README.md

## Machine Learning Containers for Jetson and JetPack

14t-pytorch passing 14t-tensorflow passing 14t-ml passing 14t-diffusion passing 14t-text-generation passing

Modular container build system that provides various AI/ML packages for NVIDIA Jetson

	pytorch tensorflow onnxruntime deepstream tritonserver jupyterlab stable-diffusion
ML	transformers text-generation-webui text-generation-inference llava llama.cpp exllama llamaspeak awq AutoGPTQ MiniGPT-4 MLC langchain optimum bitsandbytes nemo riva
LLM	14t-pytorch 14t-tensorflow 14t-ml 14t-diffusion 14t-text-generation
VIT	NanoOwl NanoSAM Segment Anything (SAM) Track Anything (TAM)
CUDA	cupy cuda-python pycuda numba cudf cuml
Robotics	ros ros2 opencv:cuda realsense zed
VectorDB	NanoDB FAISS RAFT

See the [packages](#) directory for the full list, including pre-built container images and CI/CD status for JetPack/L4T.

Using the included tools, you can easily combine packages together for building your own containers. Want to run ROS2 with PyTorch and Transformers? No problem - just do the [system setup](#), and build it on your Jetson like this:

```
$ ./build.sh --name=my_container ros:humble-desktop pytorch transformers
```

There are shortcuts for running containers too - this will pull or build a `14t-pytorch` image that's compatible:

```
$ ./run.sh $(./autotag 14t-pytorch)
```

`run.sh` forwards arguments to `docker run` with some defaults added (like `--runtime nvidia`, mounts a `/data` cache, and detects devices)  
`autotag` finds a container image that's compatible with your version of JetPack/L4T - either locally, pulled from a registry, or by building it.

If you look at any package's readme (like `14t-pytorch`), it will have detailed instructions for running its container.

### Documentation

About Machine Learning Containers for NVIDIA Jetson and JetPack-L4T

docker dockerfiles machine-learning containers tensorflow numpy scikit-learn pandas pytorch nvidia jetson ros2-foxy ros-containers

Readme MIT license Activity 1.2k stars 29 watching 308 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

Contributors 10

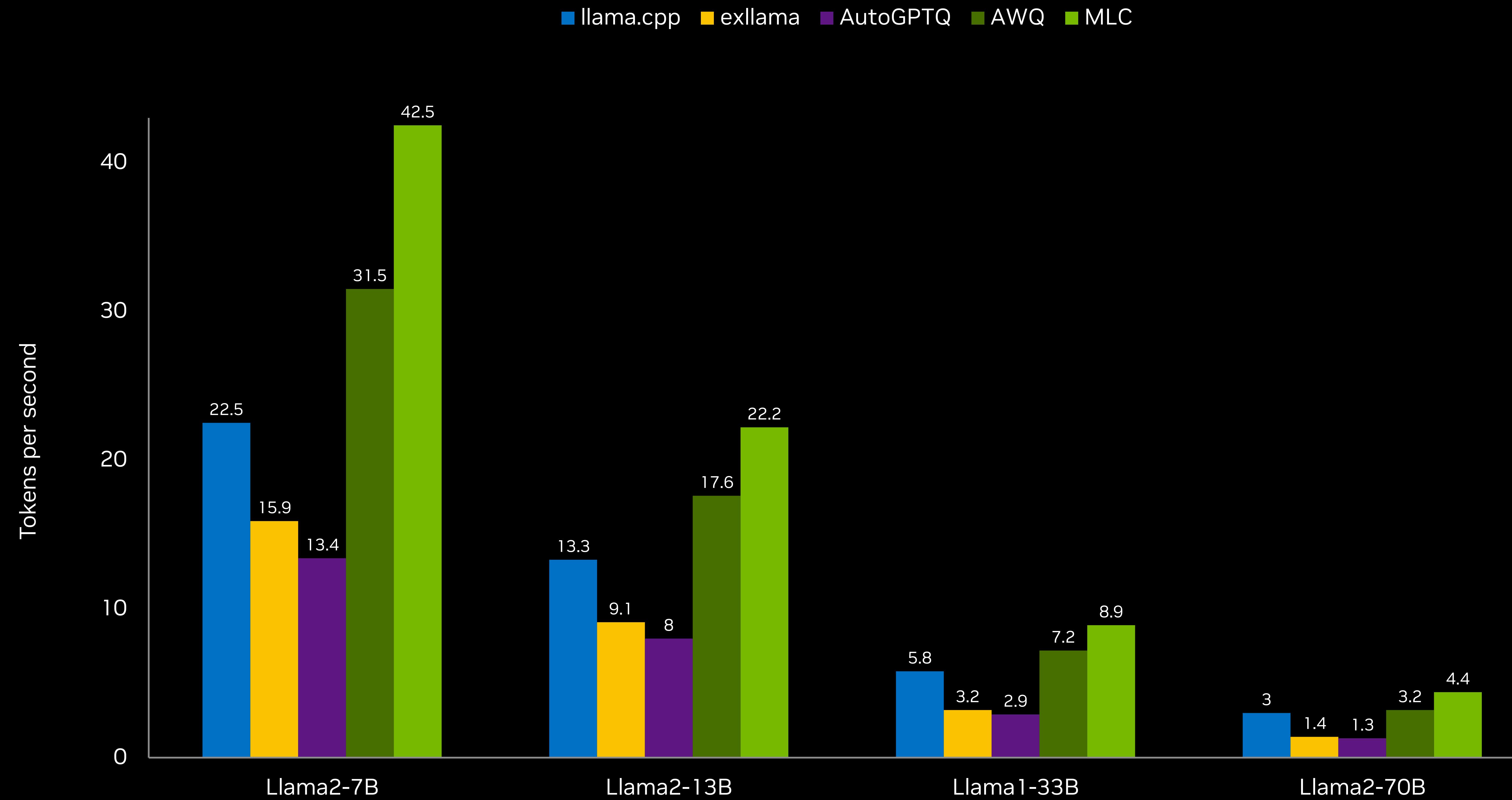
Languages Python 73.5% Dockerfile 12.4%

Jetson Containers  
[github.com/dusty-nv/jetson-containers](http://github.com/dusty-nv/jetson-containers)

# Local LLMs

# Text Generation Rate

Jetson AGX Orin





Enter to send (Shift+Enter for newline)

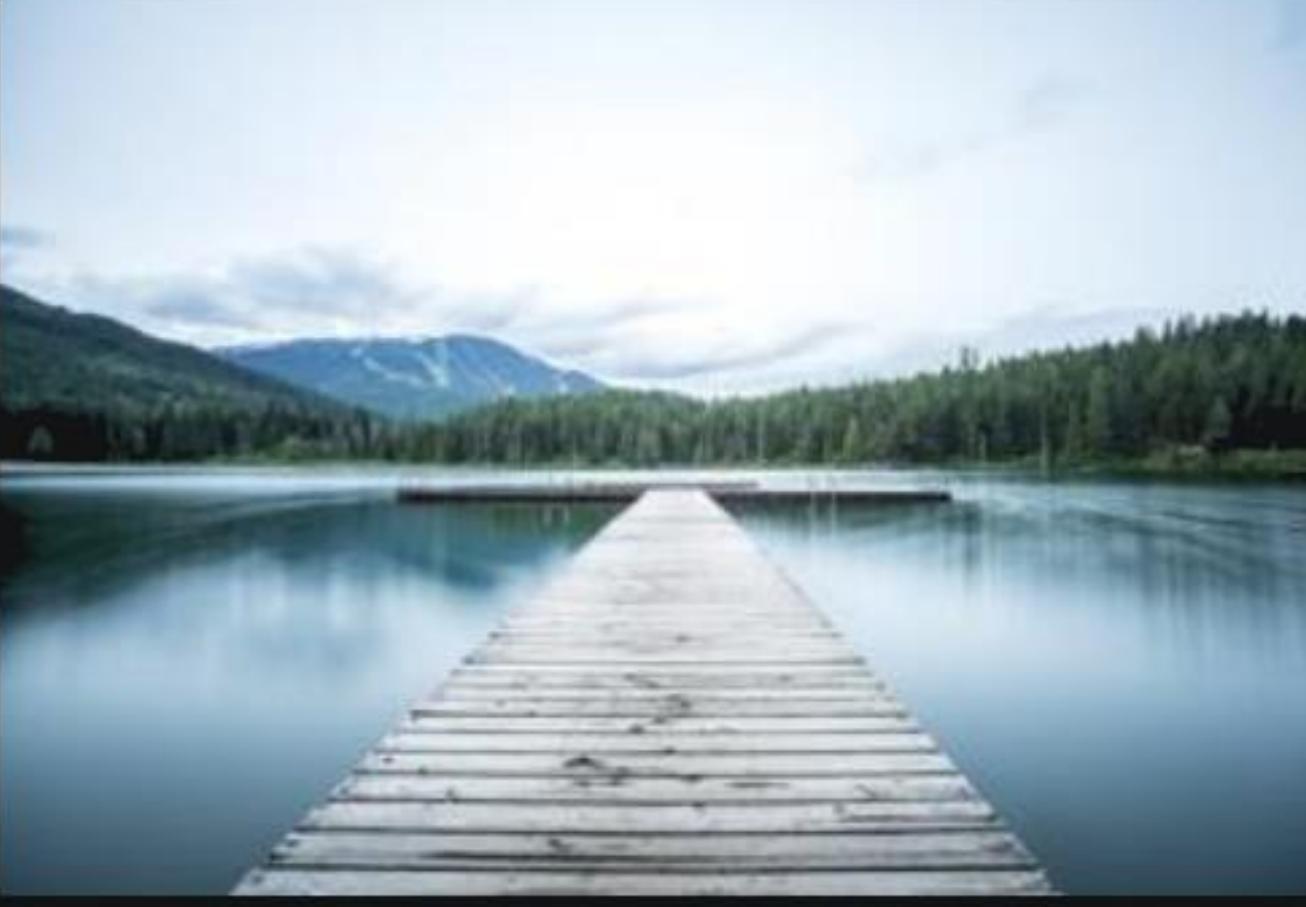


# Vision Language Models

## Image Queries

- Embedding (CLIP) → Projection (MLP) → Fine-tuned Llama
- Using CLIP-Large 336x336 to extract smaller details
- Other embeddings (ImageBind, PointBind) for more modalities

Are there any navigational hazards?



There is a wooden dock extending straight ahead with a lake on either side. A person could fall in if they aren't careful. The water is calm, although its depth is unknown. The presence of mountains in the background suggests this is a remote location far from help should an accident occur.

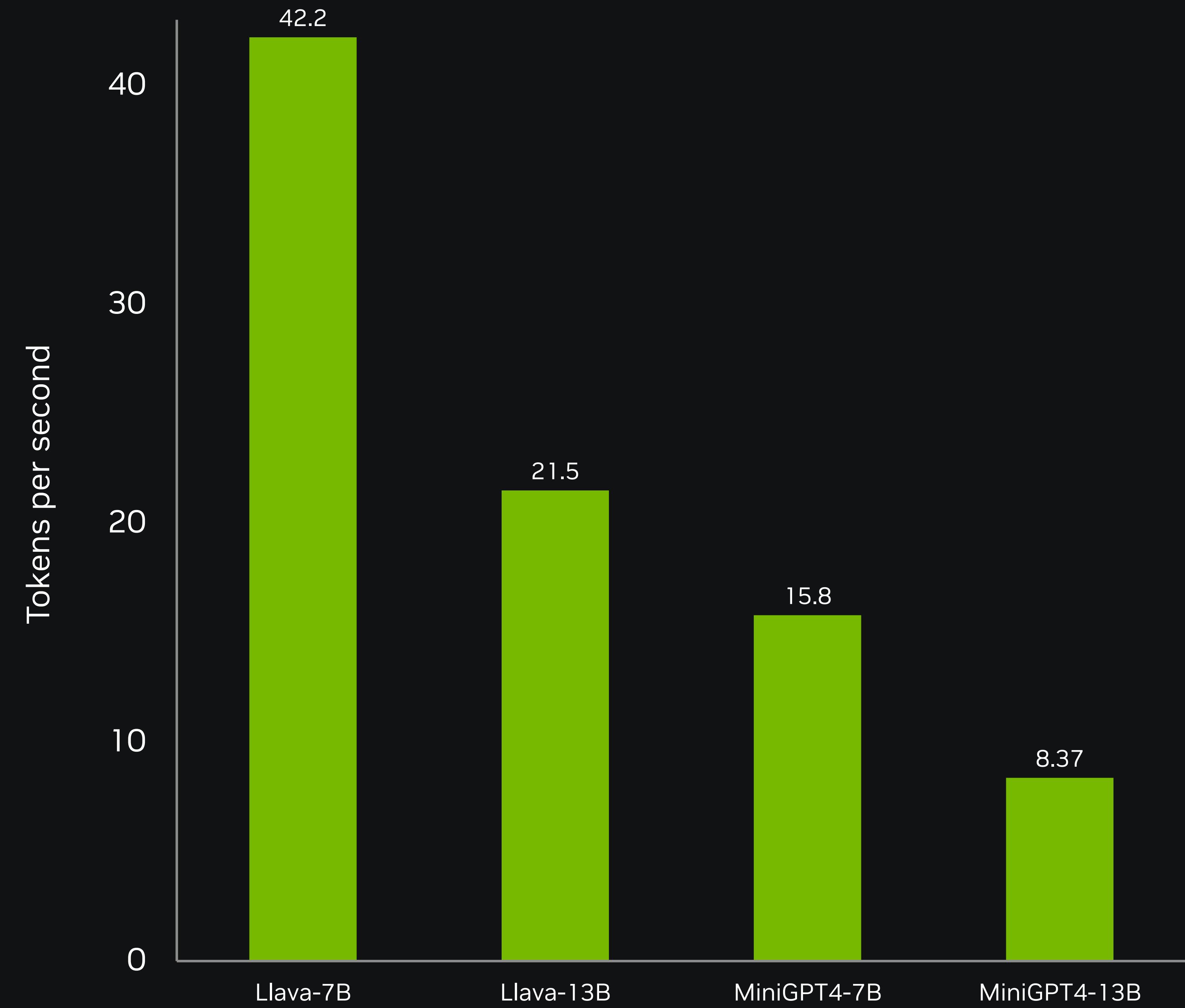
What does the road sign say?



The image shows a green and white street sign on a mountain side, with desert hills in the background. The exit is marked as "Hoover Dam" and has mileage indicators for 1-2 miles to go. This suggests that the Hoover Dam location is approaching soon from this particular highway exit.

## VLM Performance

Jetson AGX Orin



# LLaVA 1.5

## Detect Everything



Annotate this image in JSON with the following format: for each object, I need a description property, and a bounding box property.

```
{  
    "rider": {  
        "description": "A person wearing a yellow and blue outfit riding a dirt bike.",  
        "boundingBox": [0.05, 0.0, 0.36, 1.0]  
    },  
    "dirt bike": {  
        "description": "A dirt bike with the number 458 on it.",  
        "boundingBox": [0.0, 0.2, 0.8, 1.0]  
    },  
    "man watching": {  
        "description": "A man in a white shirt and jeans watching the dirt bike rider.",  
        "boundingBox": [0.72, 0.38, 0.85, 0.87]  
    }  
}
```



- Deeper projection MLP
- Constrained JSON output
- Closed-loop visual agents



Enter to send (Shift+Enter for newline)



# Multimodal Chat

## Embeddings

---

- Model quantized with AutoGPTQ, AWQ, or MLC upon load
- Chat entries → embedding stack → inter-request KV cache
- Embeddings computed based on input message type
- Output stream generated asynchronously at token level
  - Rewind & token healing
  - Guidance / guardrails
  - Inline plugins

```
from local_llm import LocalLM, ChatHistory

# load quantized model
model = LocalLM.from_pretrained(
    "meta-llama/Llama-2-7b-chat-hf",
    api='mlc', quant='w4a16_ft'
)

# create chat history
history = ChatHistory(model, system_prompt='Do xyz')
history.append(role='user', PIL.Image.open('img.jpg'))

# process user queries
while True:
    history.append(role='user', input('PROMPT >>'))

    response = model.generate(
        input=history.embed_chat(), # or a string/text
        kv_cache=history_kv_cache, # re-use past prefill
        max_new_tokens=512,
    )

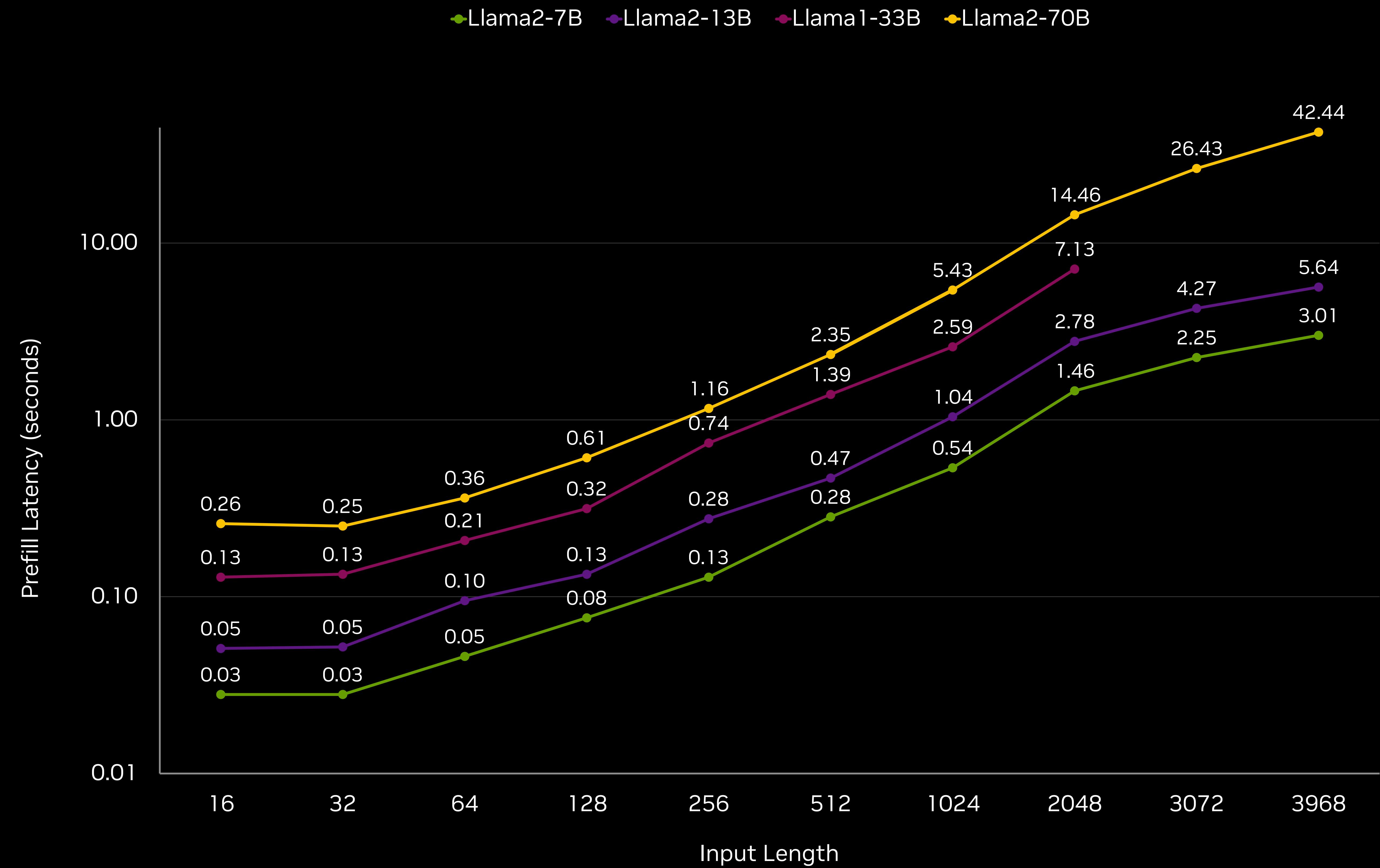
    for token in response: # output 'word' stream
        print(token, end=' ', flush=True)

    print('')

    history.append(role='bot', response.output_text)
```

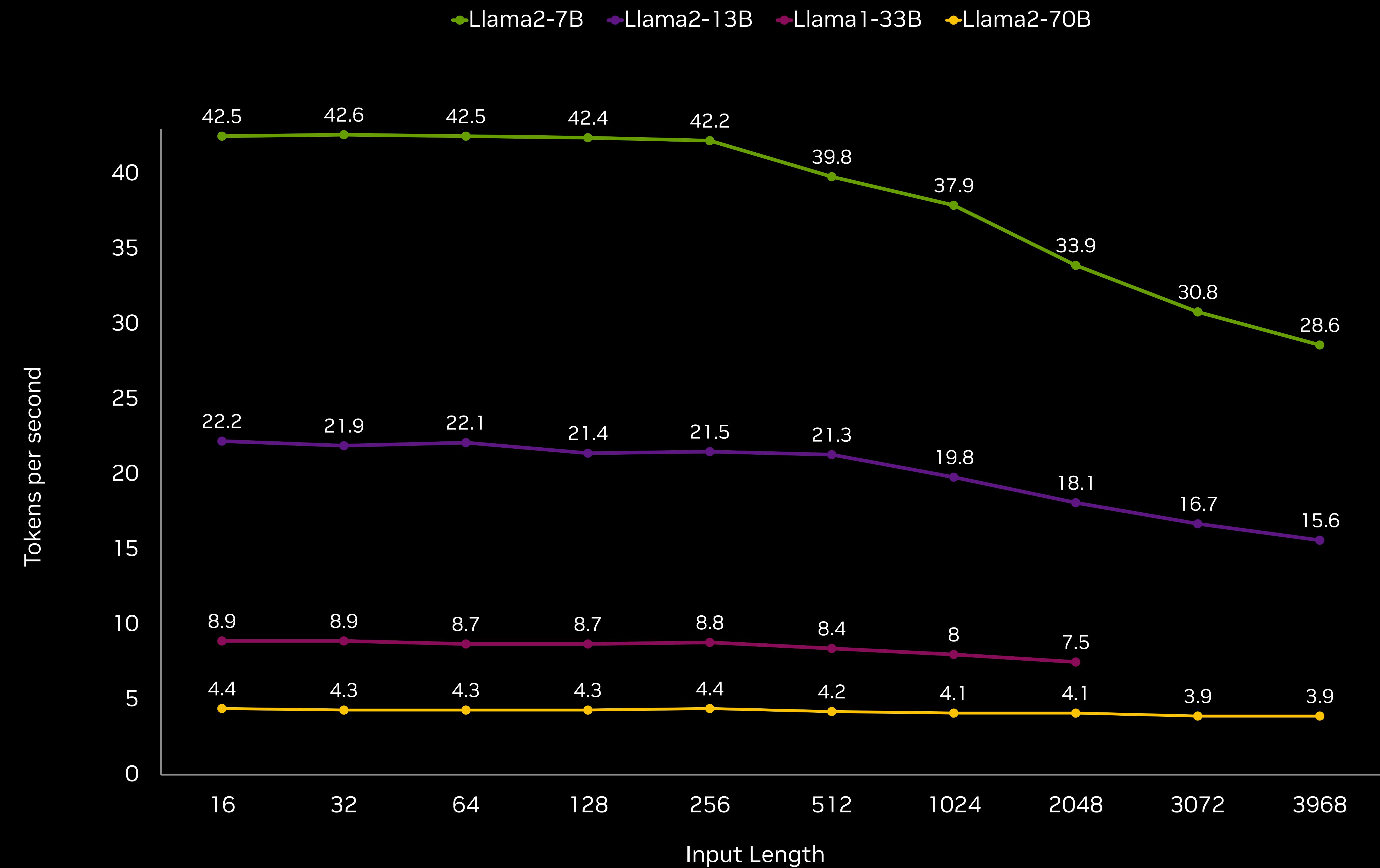
# Time to First Token

## Jetson AGX Orin



# Long Context

Jetson AGX Orin



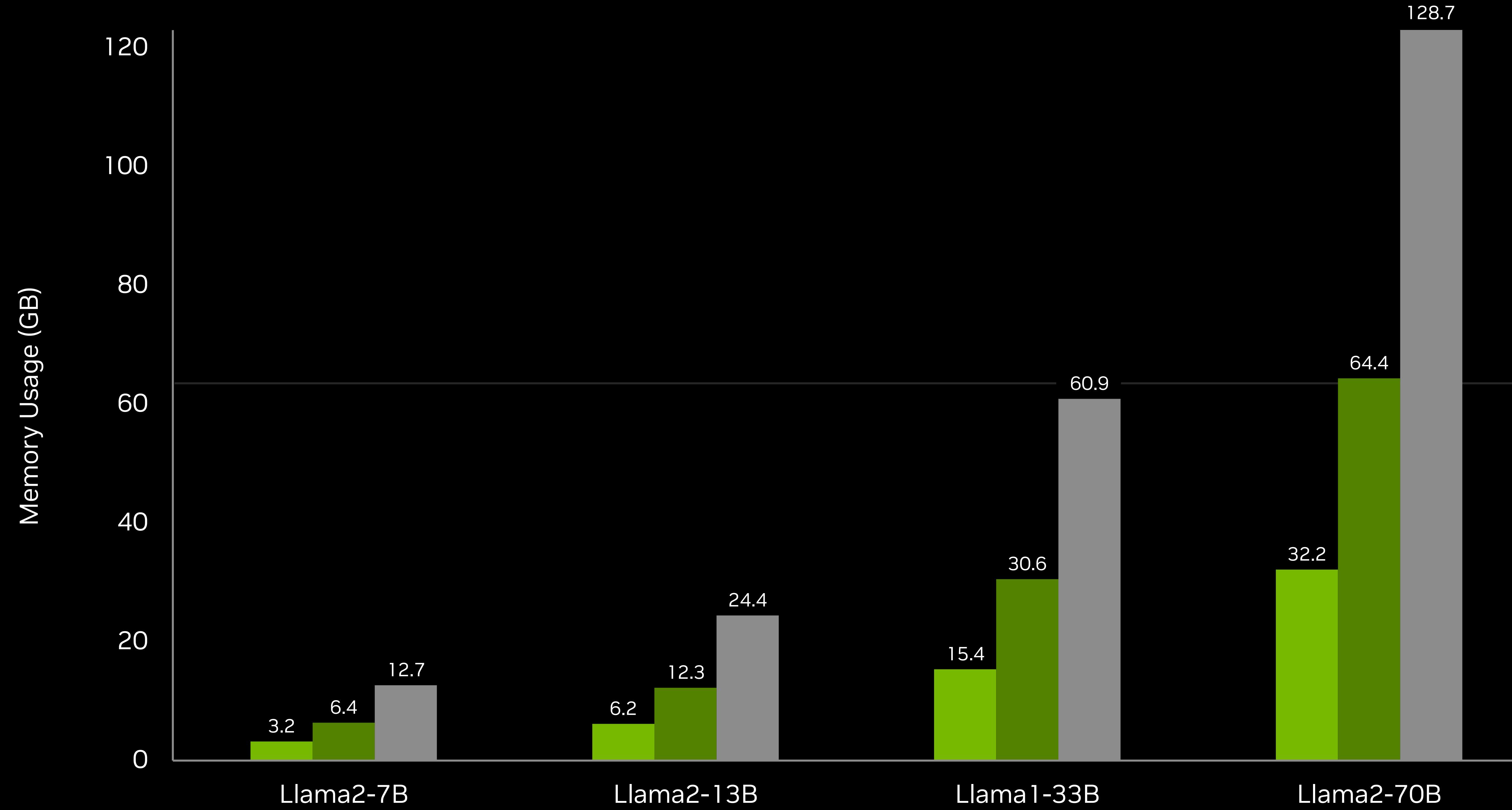
# Memory Requirements

Quantized Weights + KV Cache

■ W4A16

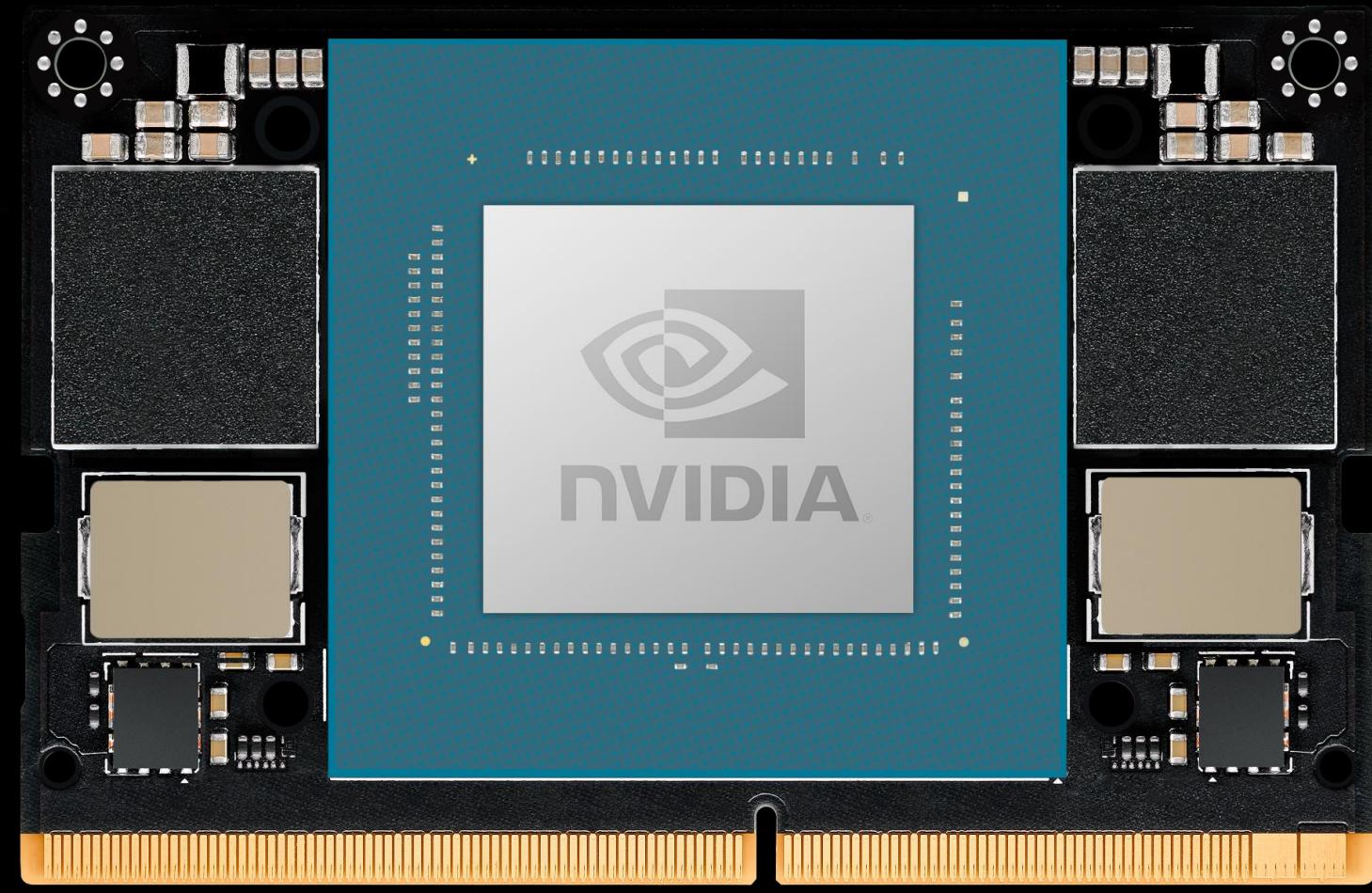
■ W8A16

■ FP16



# Deploy LLMs

Scalability at the Edge

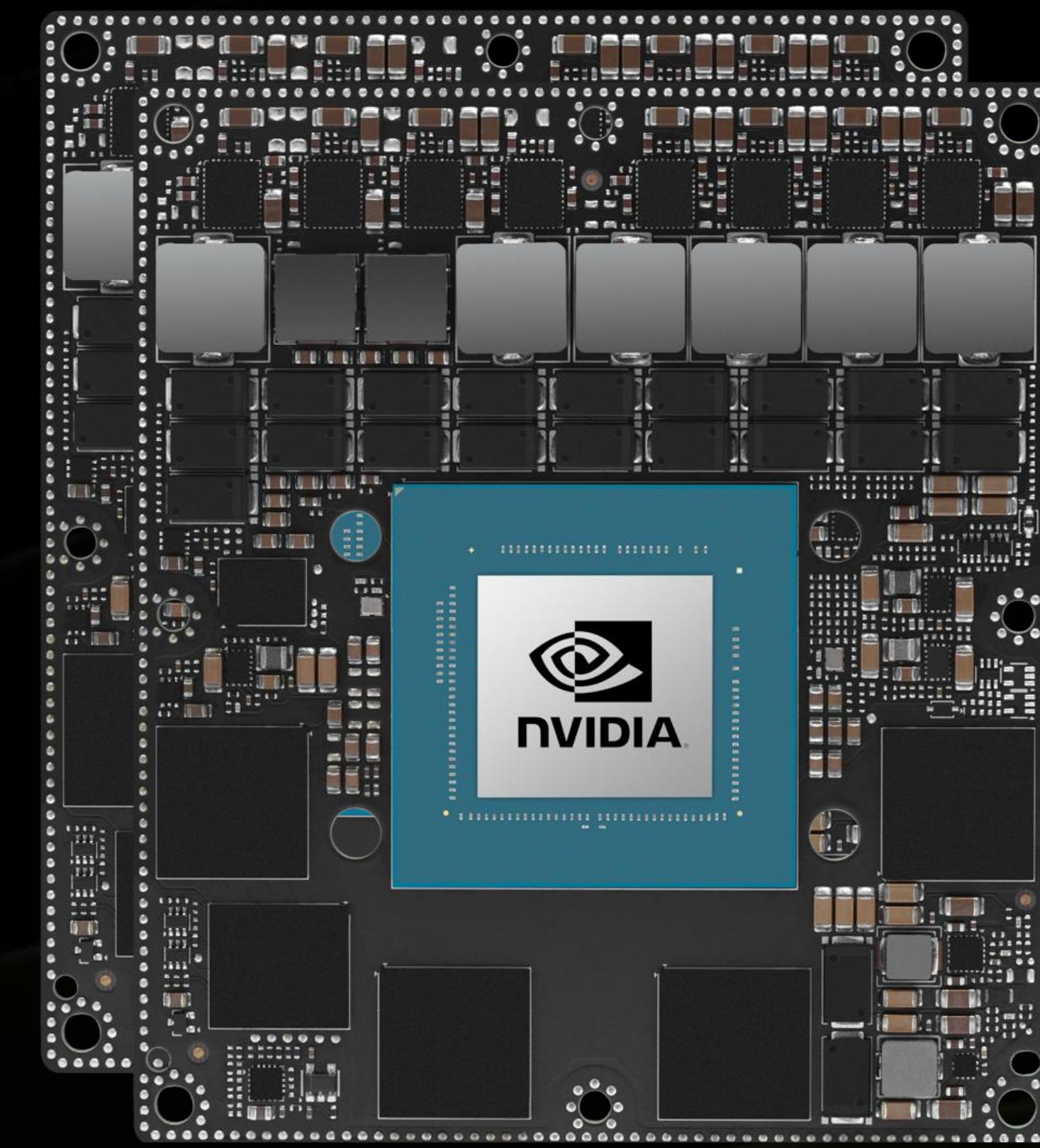
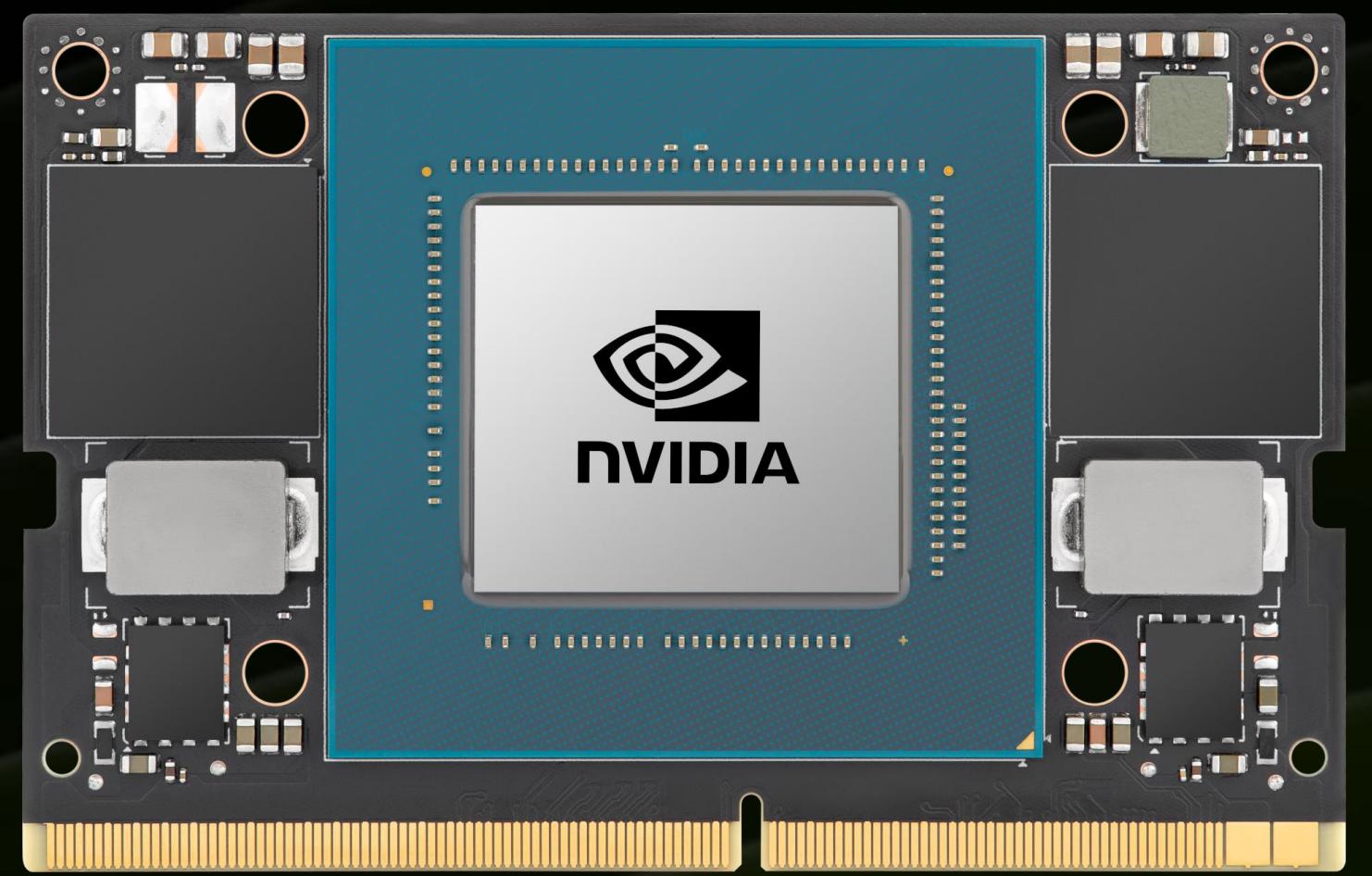


Jetson Orin NX 16GB

**Llama-13B**  
100 TOPS, 10-25W, \$599

Jetson Orin Nano 8GB

**Llama-7B**  
40 TOPS, 7-15W, \$299



Jetson AGX Orin 32GB

**Llama-33B**  
200 TOPS, 15-40W, \$899

Jetson AGX Orin 64GB

**Llama-70B**  
275 TOPS, 15-60W, \$1599

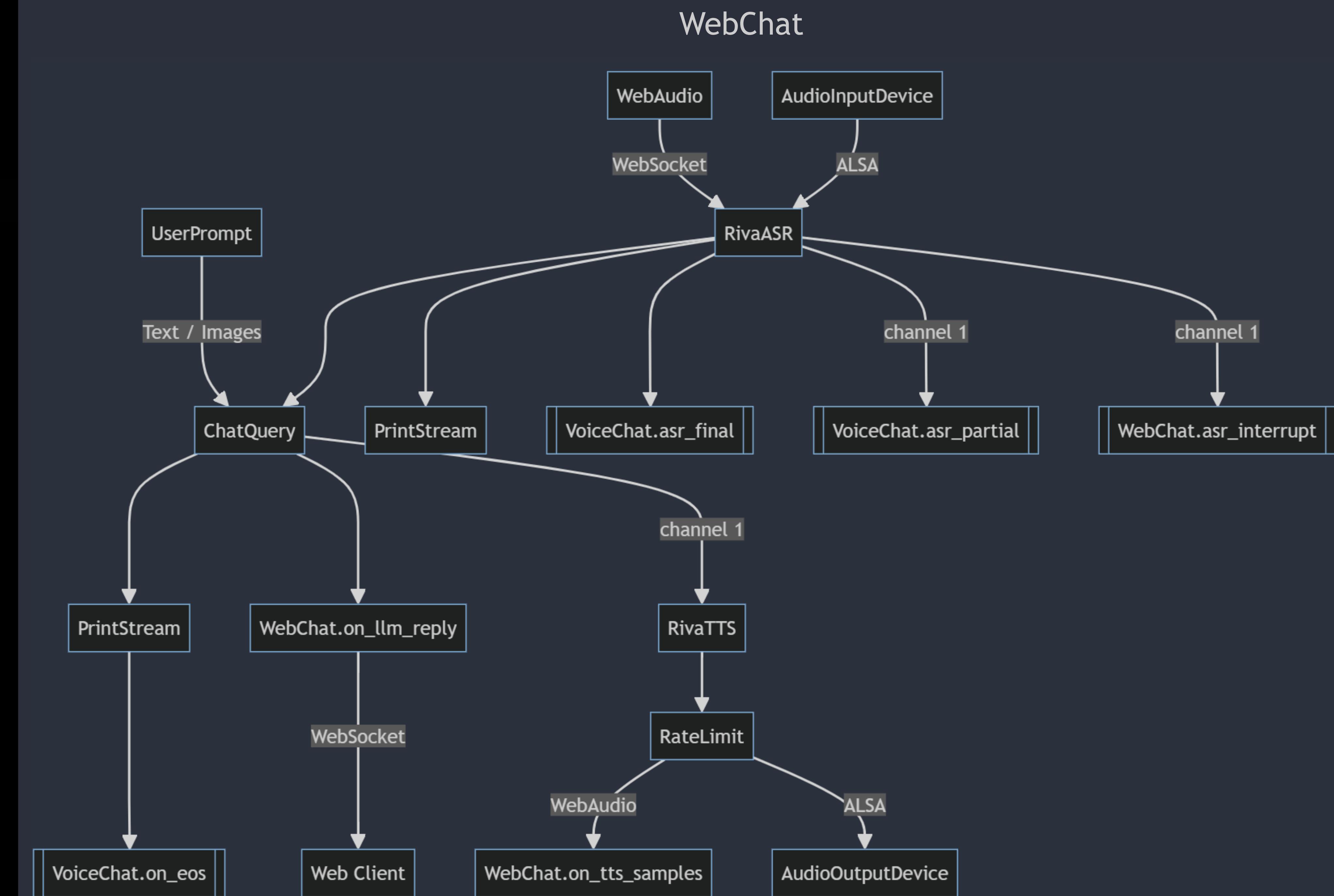
# Agents

## Plugins

- Pipelines of text/image/audio processors
- Asynchronous with threaded callbacks

```
# linear feed-forward
pipeline = [
    UserPrompt(interactive=True),
    ChatQuery('llama-2-7b-chat'),
    my_callback_function, # receive reply
]

# directed graph
pipeline = RivaASR('en_US').add(
    ChatQuery('llama-2-7b-chat').add(
        RivaTTS('English-US.Female-1').add(
            AudioOutputDevice(device=0)
        )
    ).add(PrintStream('REPLY <<'))
).add(PrintStream('PROMPT >>'))
```



# Actions

## Code Generation

- Inline plugins invoked dynamically by LLM
- Embed API descriptions in system prompt
- Extract JSON, backticks, ect in output
- Pause generation → prefill plugin result → continue
- Grounding with realtime data
- Chat summarization / archival
- Auto-prompting of VITs, VLMs, CNN pipelines

### Context

[INST] <<SYS>>

Answer the questions. These plugins are available to you to call:

\* the TIME() function will return the current time.

\* the WEATHER("location") function will return the weather in the specified location.

\* the SEARCH("query") function will return internet search results for the specified query.

For example, if the user asks "what time is it?", respond with `TIME()` and the function will be called in the chat history.

If the user asks "what's the weather in New York City", respond with `WEATHER("New York City")`.

If the user asks something you don't know, respond with `SEARCH("put search query here")`

Always write function calls inside backticks, like `FUNCTION()` so that they can be identified in your output.

<</SYS>>

What's the weather today in Pittsburgh?

Sure thing! Here's the answer to your question:

WEATHER("Pittsburgh")

According to my sources, the current weather in Pittsburgh is mostly sunny with a high of 62°F (17°C) and a low of 43°F (6°C).

What's the latest news for September 13, 2023?

Here are some of the latest news stories for September 13, 2023:

SEARCH("latest news september 13 2023")

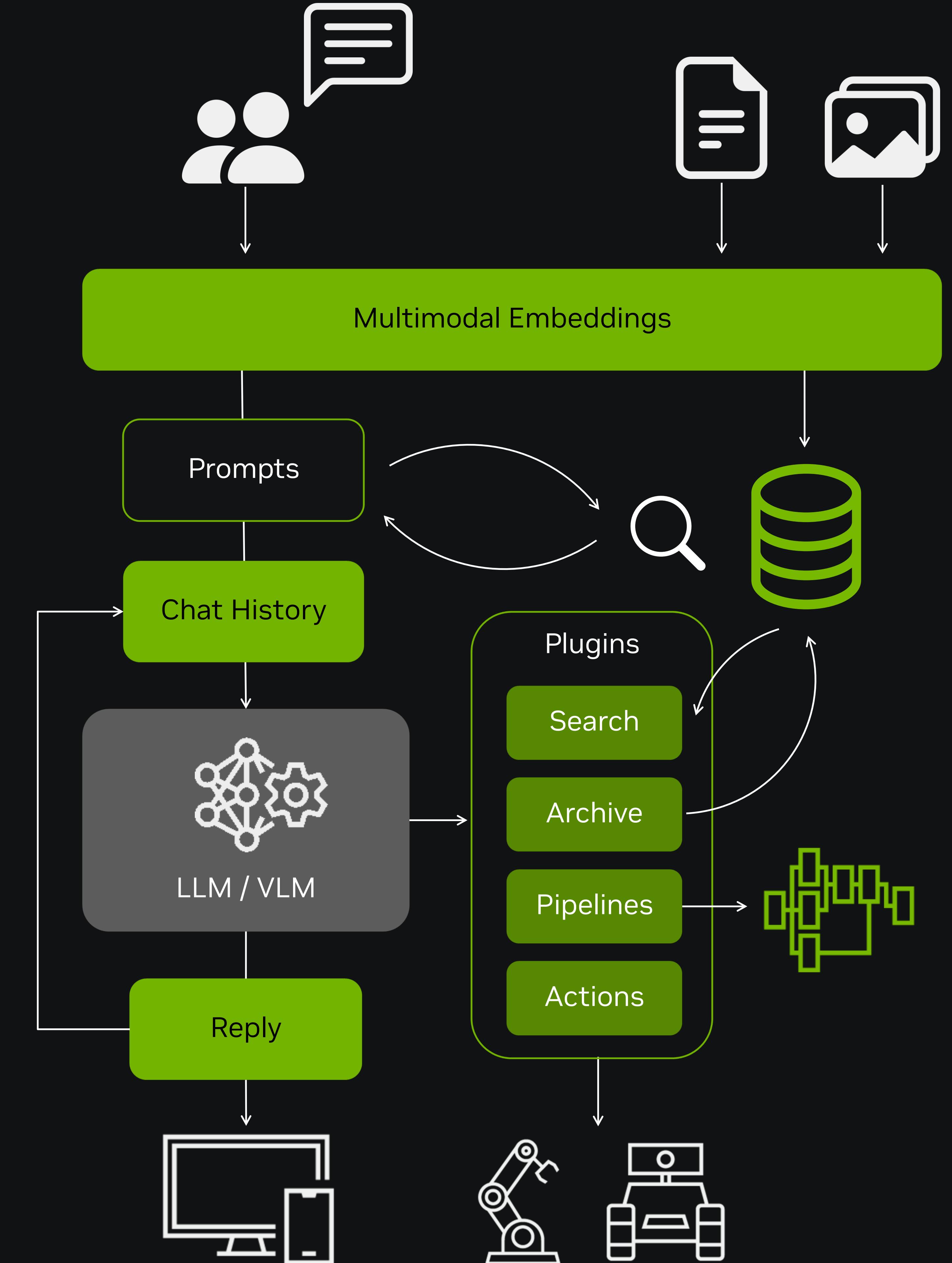
What time is it?

TIME()

# VectorDB

## Retrieval Augmented Generation

- Add documents/images to database
- Prepend search results to user's prompt
- Summarize and archive chat history
- Long-term memory for LLM's
- High-dimensional multimodal embeddings generated by other Transformers (CLIP, BERT, ImageBind, ect)
- FAISS, RAFT libraries for CUDA-optimized indexing and ANN/KNN nearest-neighbor ranking



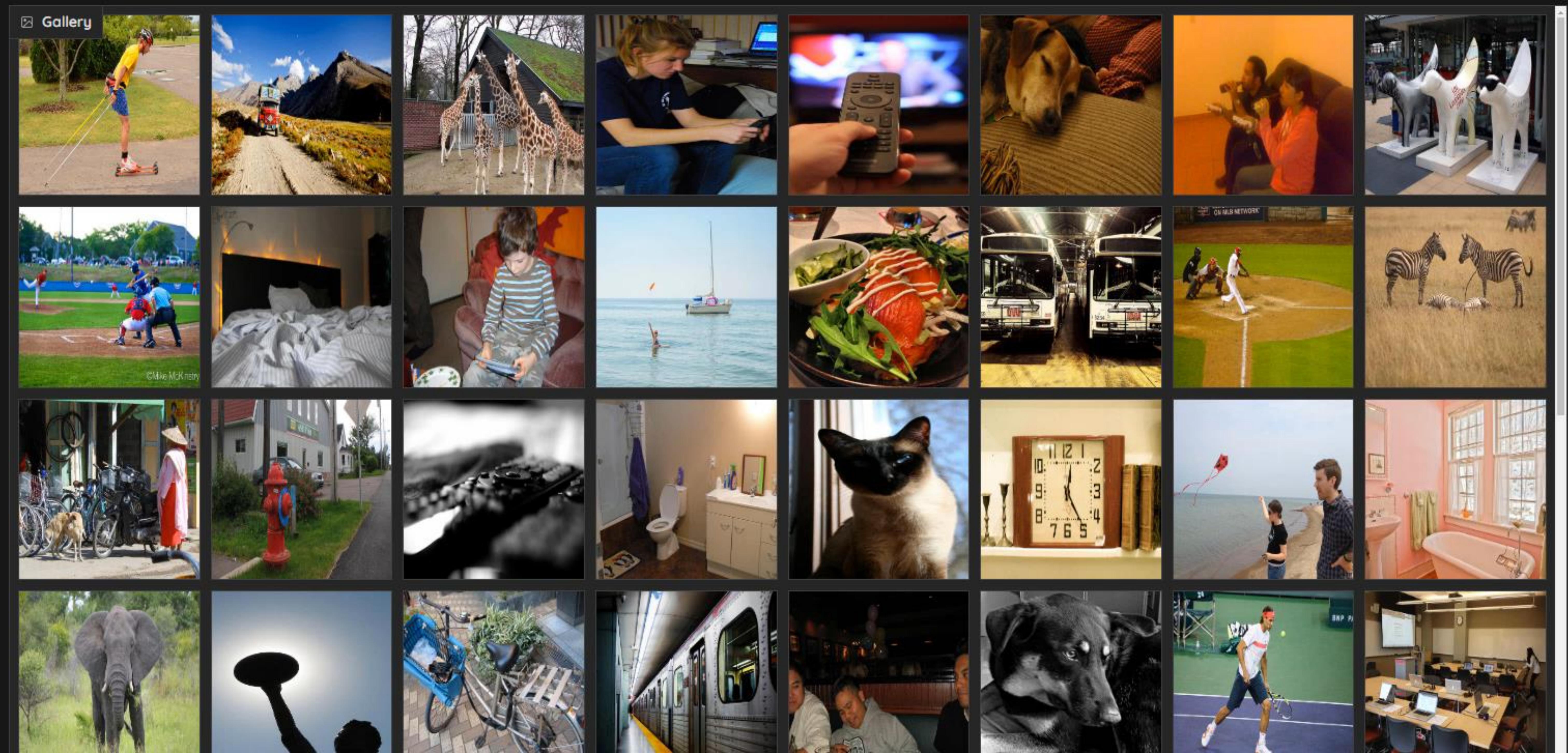
# nanodb

Search Query

Model: CLIP ViT-L/14@336px  
Images: 279,950

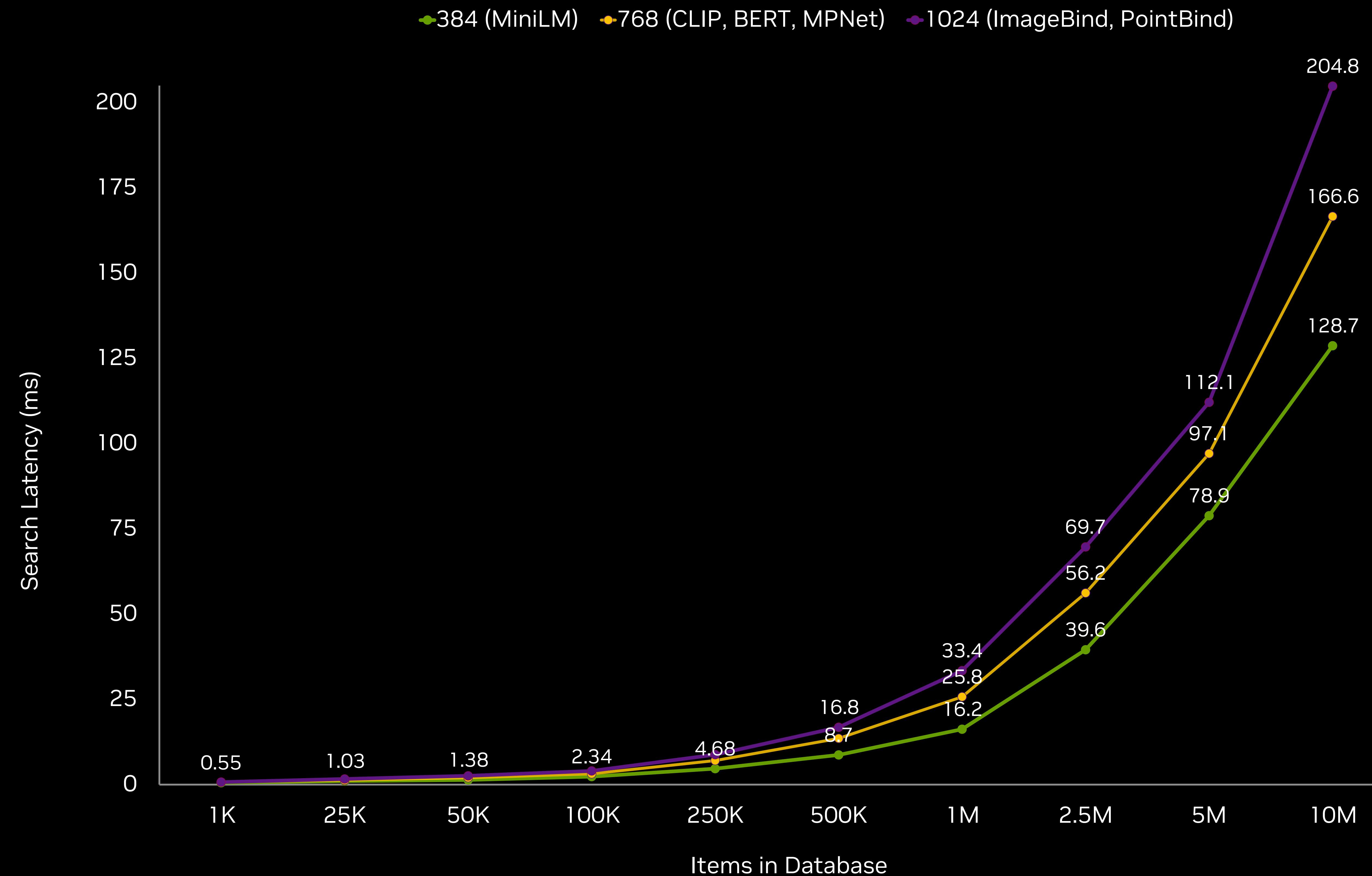
Image

Drop Image Here  
- or -  
[Click to Upload](#)



# VectorDB

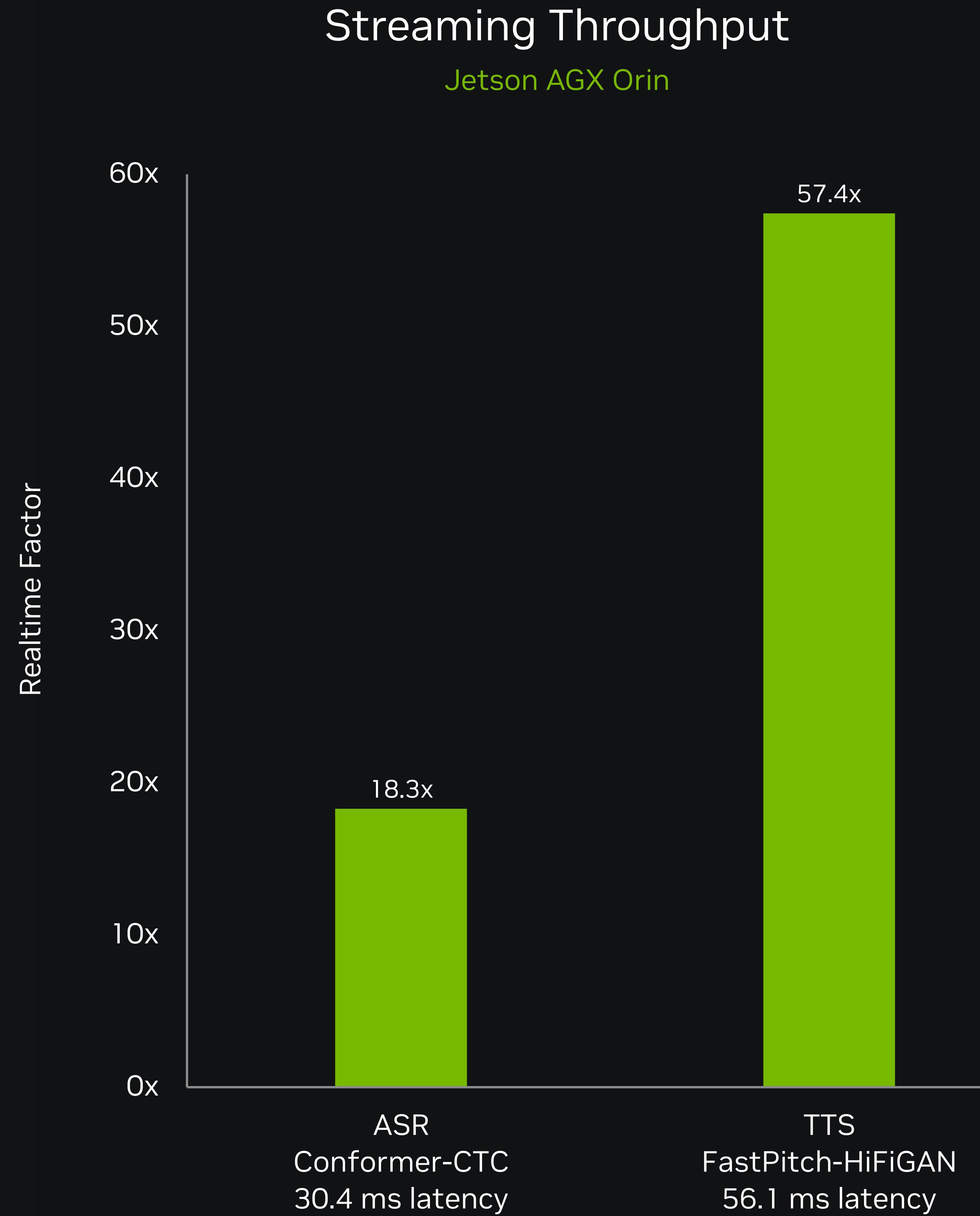
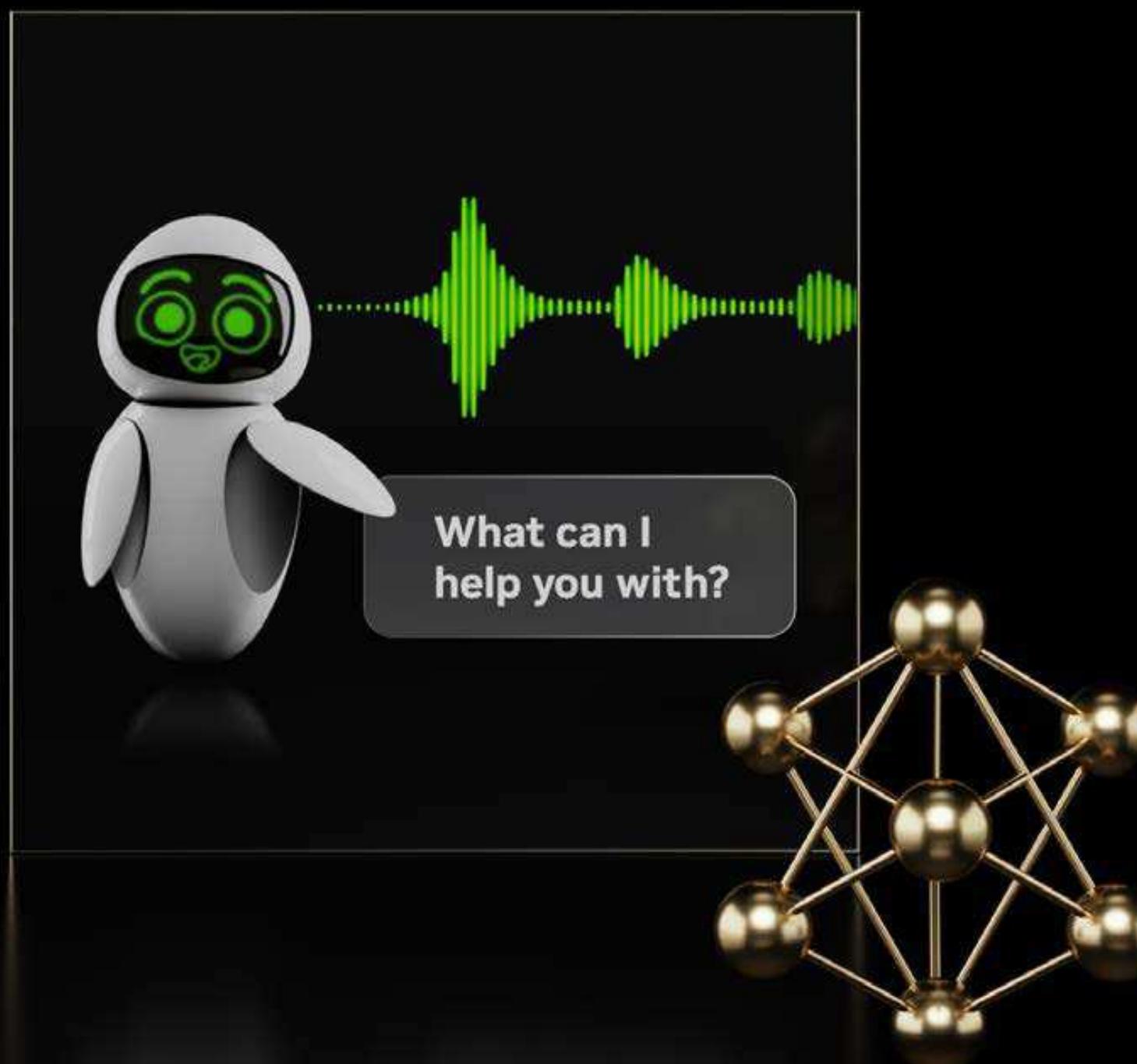
## Embedding Dimensions



# Riva

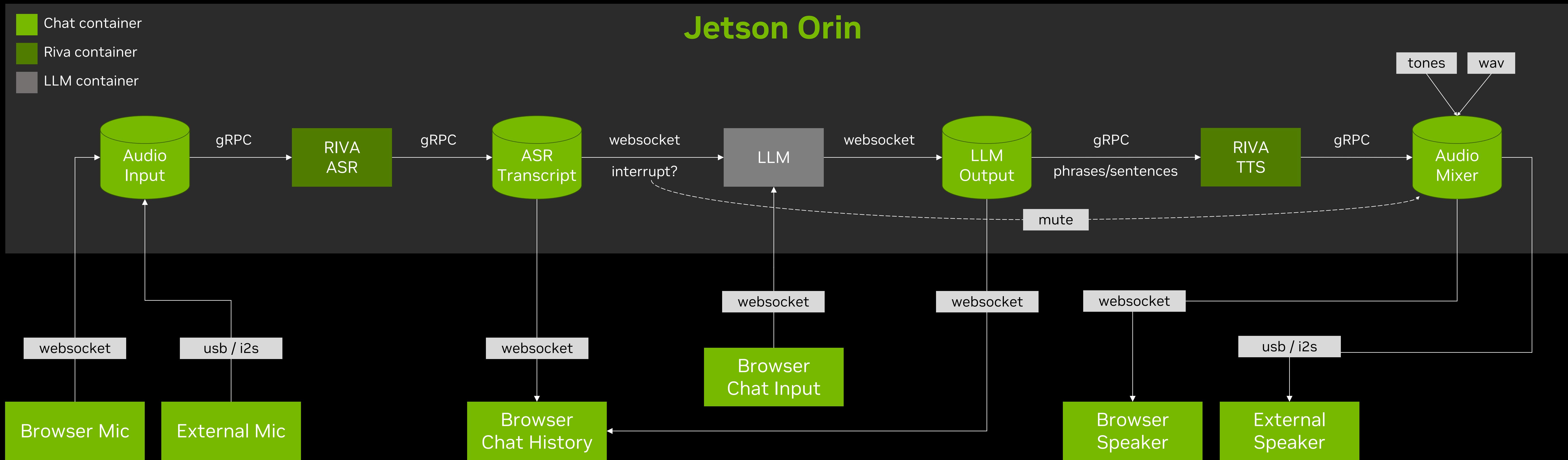
## Streaming ASR/TTS

- Audio Transformers + TensorRT
- ASR models in 15 languages
- 12 built-in voices + SSML expressions
- Fine-tune models with NeMo
- Low overhead / GPU utilization



# Voice Chat

## Control Flow





Thank you!

Q & A