OpenVINO™

CVPR 2025

CVPR Nashville JUNE 11-15, 2025

Cognitive AI for the Future: Multimodal Models and RAG in Vision Language Applications, from Training to Deployment

Dr. Zhuo Wu, Dr. Tiep Le, Dr. Gustavo Lujan, Adrian Boguszewski, Dr. Raymond Lo

intel ai

Outline of the tutorial

- Module 0: Motivations, general overview
- Module 1: Accelerate your dataset preparation locally– OpenVINO fundamentals
- Module 2: Fine-tuning the multimodal embedding and VLM
- Q&A
- Module 3: Optimize and deploy the multimodal RAG pipeline
- Module 4: Build your own cognitive AI assistant with multi-agent workflow and multimodal RAG
- Q&A

Cognitive AI for the Future: Multimodal Models and RAG in Vision Language Applications, from Training to Deployment

OpenVINO™

CVPR 2025

Module 0 :

Cognitive AI: Multimodal RAG in Vision Language Applications

Speaker: Raymond Lo, Zhuo Wu

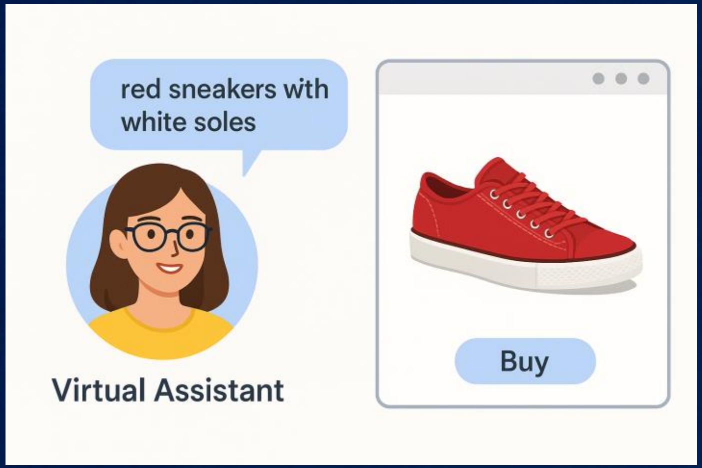
Job title: AI Software Evangelist

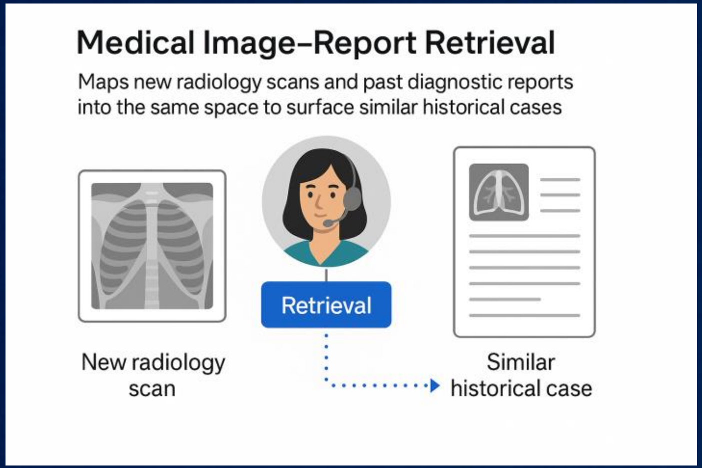
Outline

- What is Cognitive AI? Motivations & Quick Demo
- Multimodal RAG pipeline
- Multimodal RAG with agentic workflow

Cognitive AI for the Future

- Learn from multimodal sources
- Learn understanding and reasoning
- Enhance decision-making
- Automate complex tasks





127.0.0.1:7875

Powered by OpenVINO + MCP Tools

1) Upload or choose a video

2) Build Vector Store

Vector Store is Ready

Agent's Reasoning Log

Your Actions / Cart

Conversation

Type your message...

Send

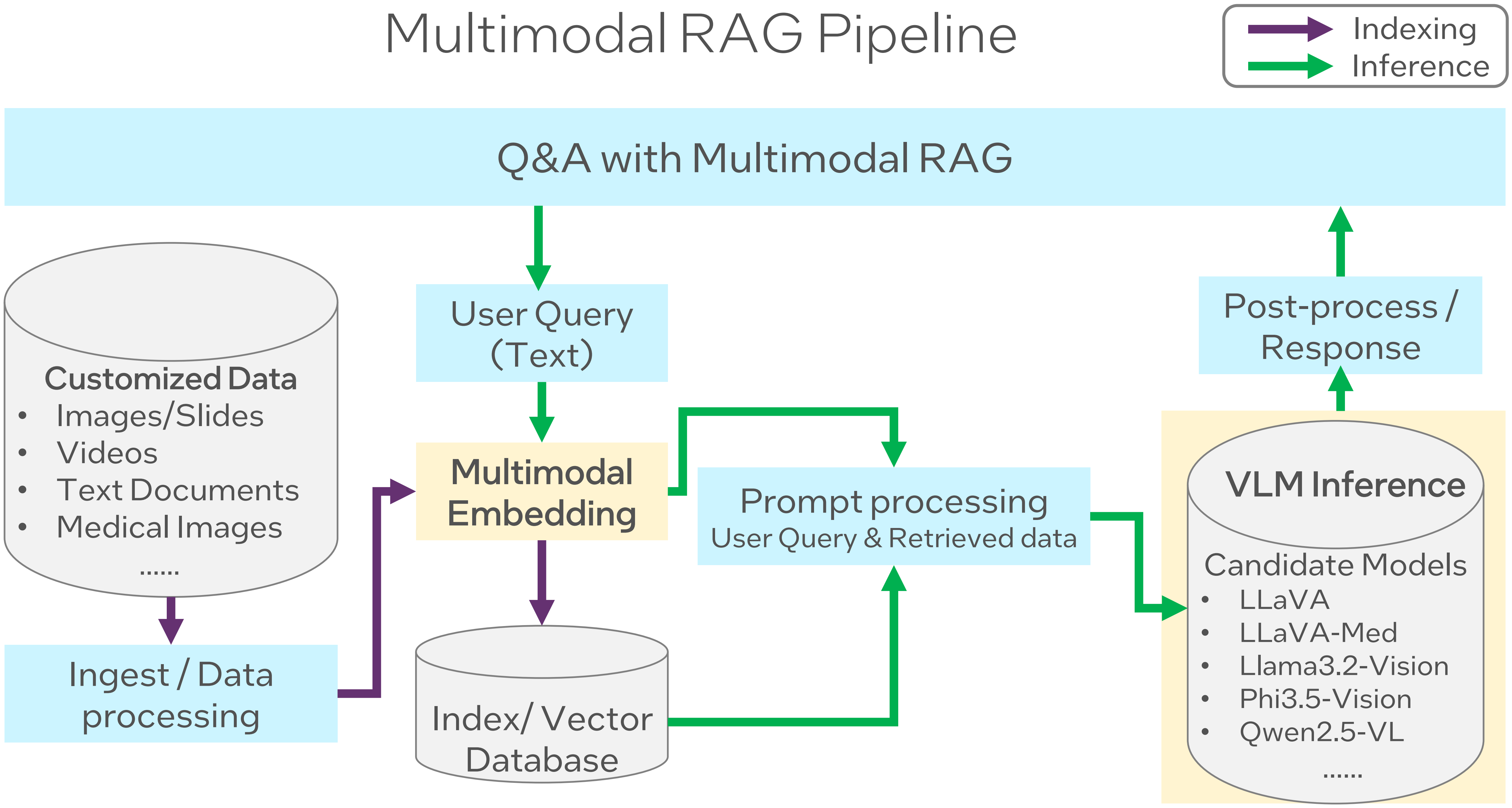
Stop

Clear

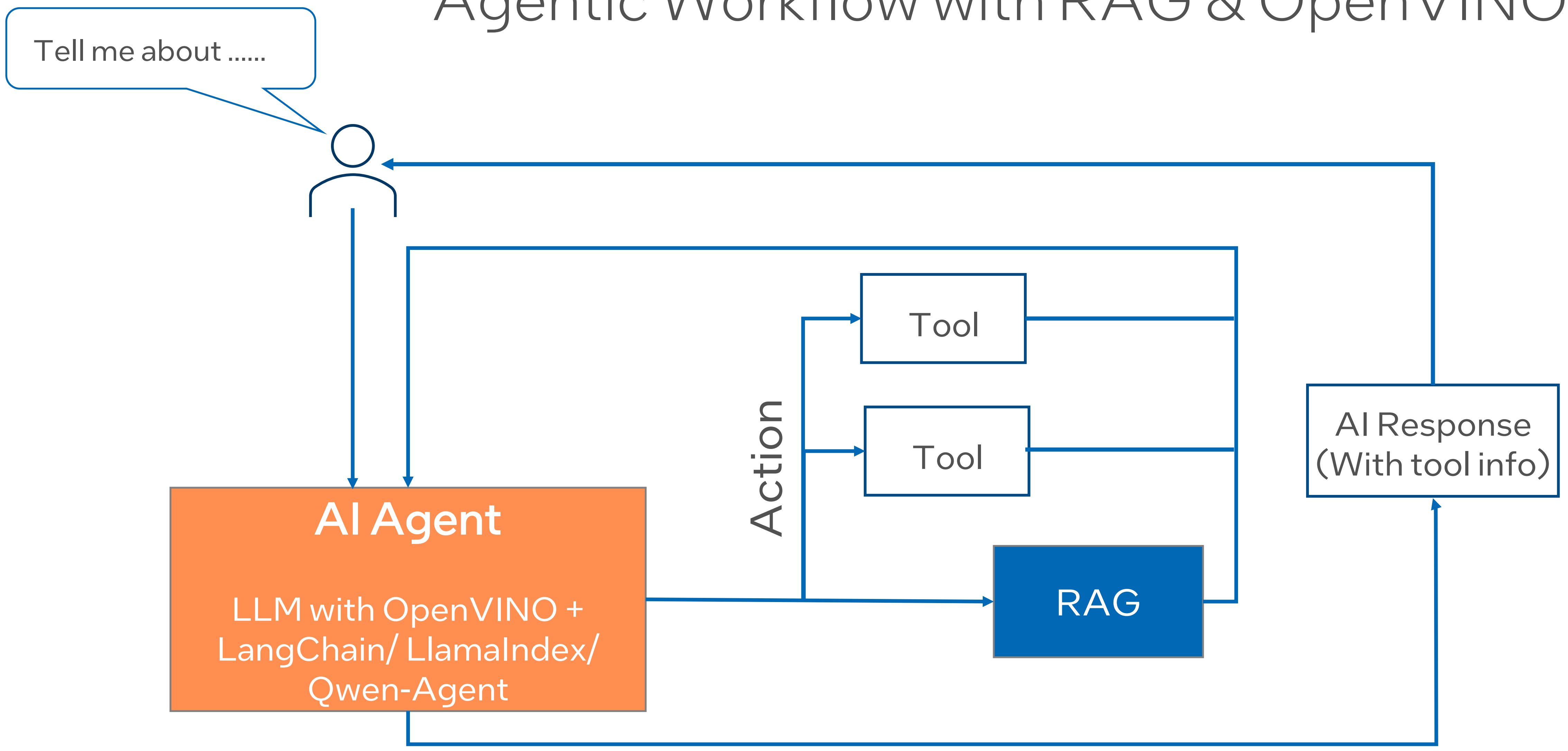
Click example, then Send

What dessert is included in this video?

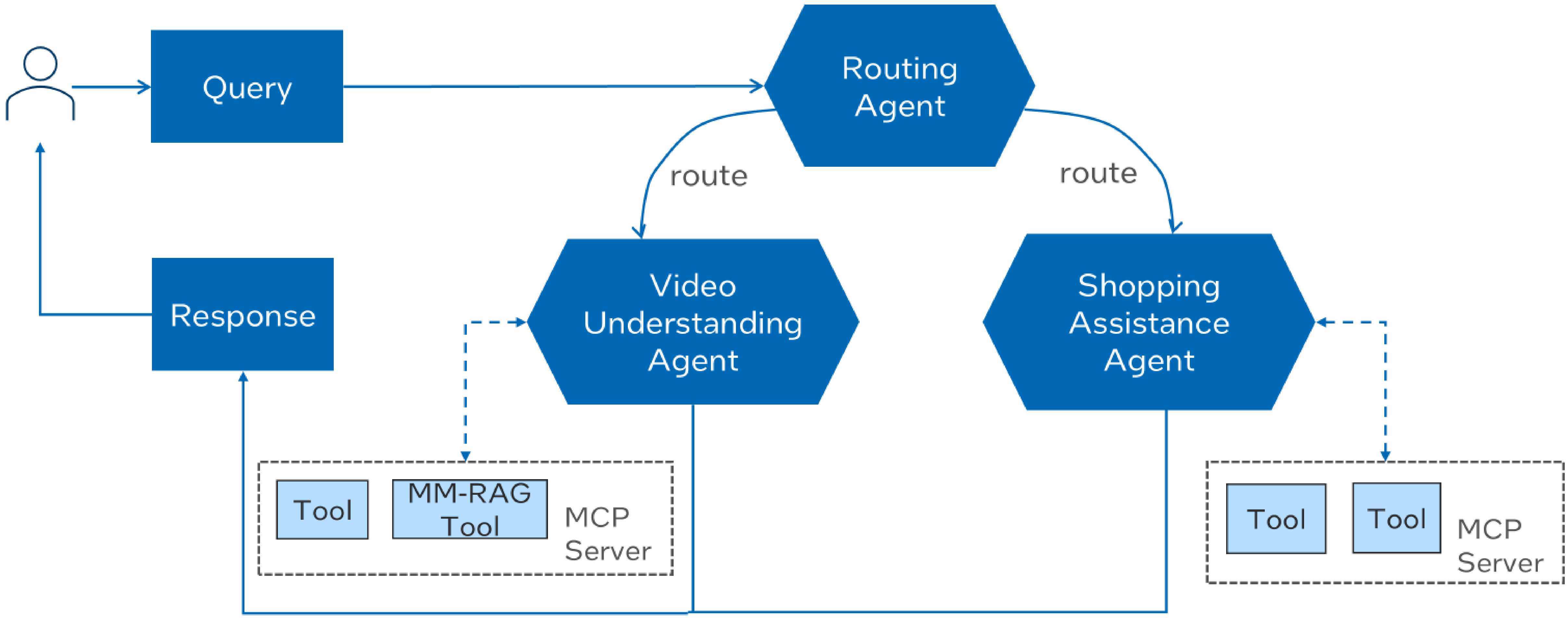
Multimodal RAG Pipeline



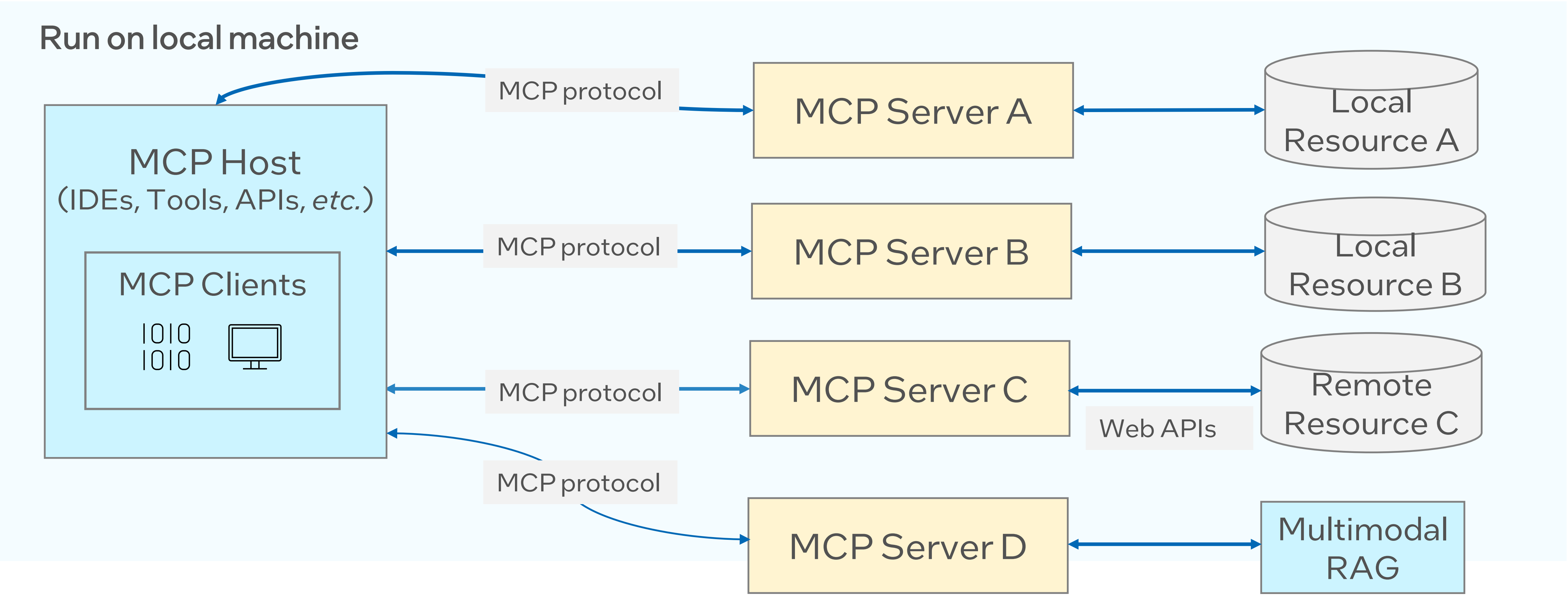
Agentic Workflow with RAG & OpenVINO



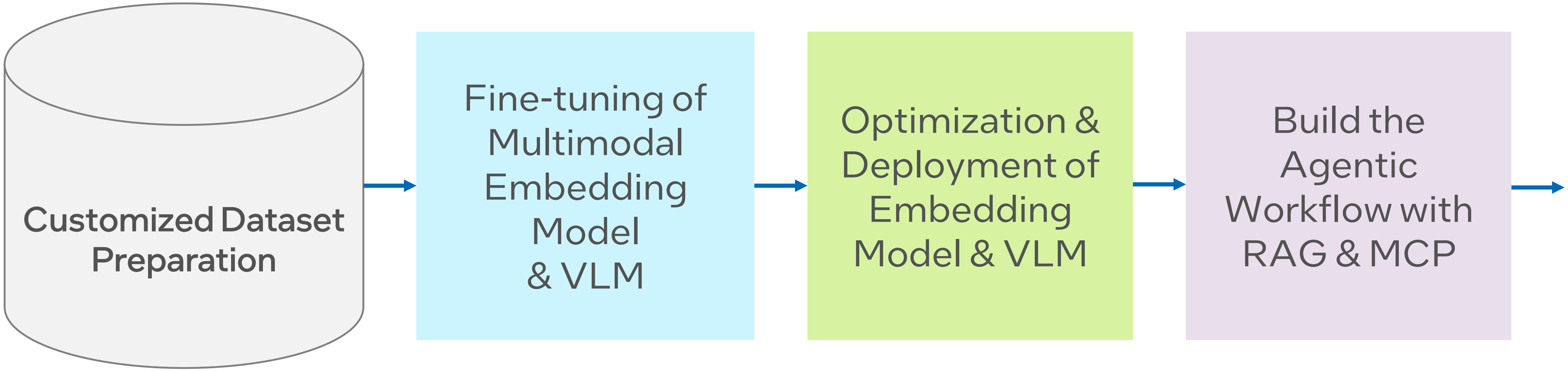
Multi-agent Workflow with RAG & OpenVINO



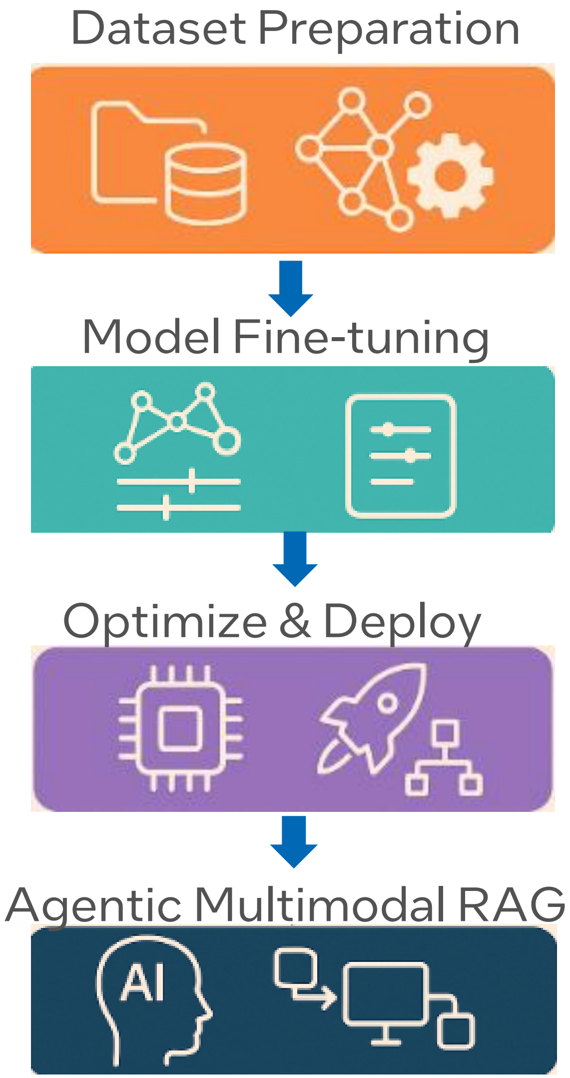
Multimodal RAG with Agentic Workflow



How to Build Such an Agentic Workflow, from Training to Deployment?



What You Will Learn



- Module 1: Tool for accelerating your dataset preparation locally
- Module 2: How to fine-tune the multimodal embedding model and VLM
- Module 3: How to optimize and deploy the multimodal RAG pipeline
- Module 4: How to build Agentic multimodal RAG

Thank You

Cognitive AI for the Future: Multimodal Models and RAG in Vision Language Applications, from Training to Deployment

OpenVINO™

CVPR 2025

Module 1 :

Accelerate your dataset preparation locally: OpenVINO Fundamentals

Speaker: Adrian Boguszewski

Job title: AI Software Evangelist

Deploying Challenges

Infrastructure	Scalability	Performance optimization
Model versioning and updates	Data management and preprocessing	Security and privacy
Monitoring and debugging	Integration with existing systems	User feedback and improvement

Deploying Challenges

Infrastructure	Scalability	Performance optimization
Model versioning and updates	Data management and preprocessing	Security and privacy
Monitoring and debugging	Integration with existing systems	User feedback and improvement

Google Summer of Code

open source initiative
Apache 2.0

OpenVINO™

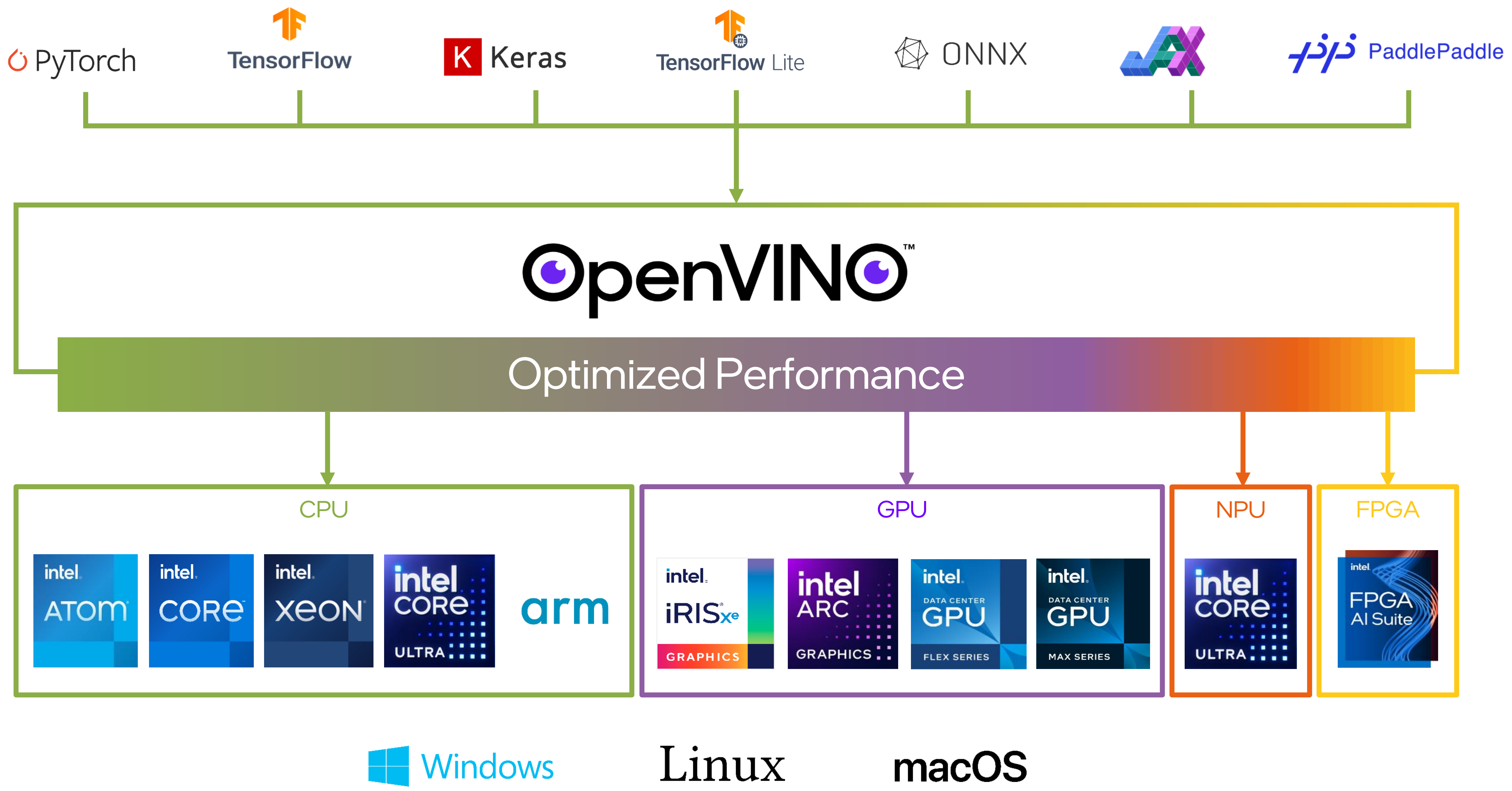
Visual+
NLP, Audio,
LLM, GenAI,
Transformers...

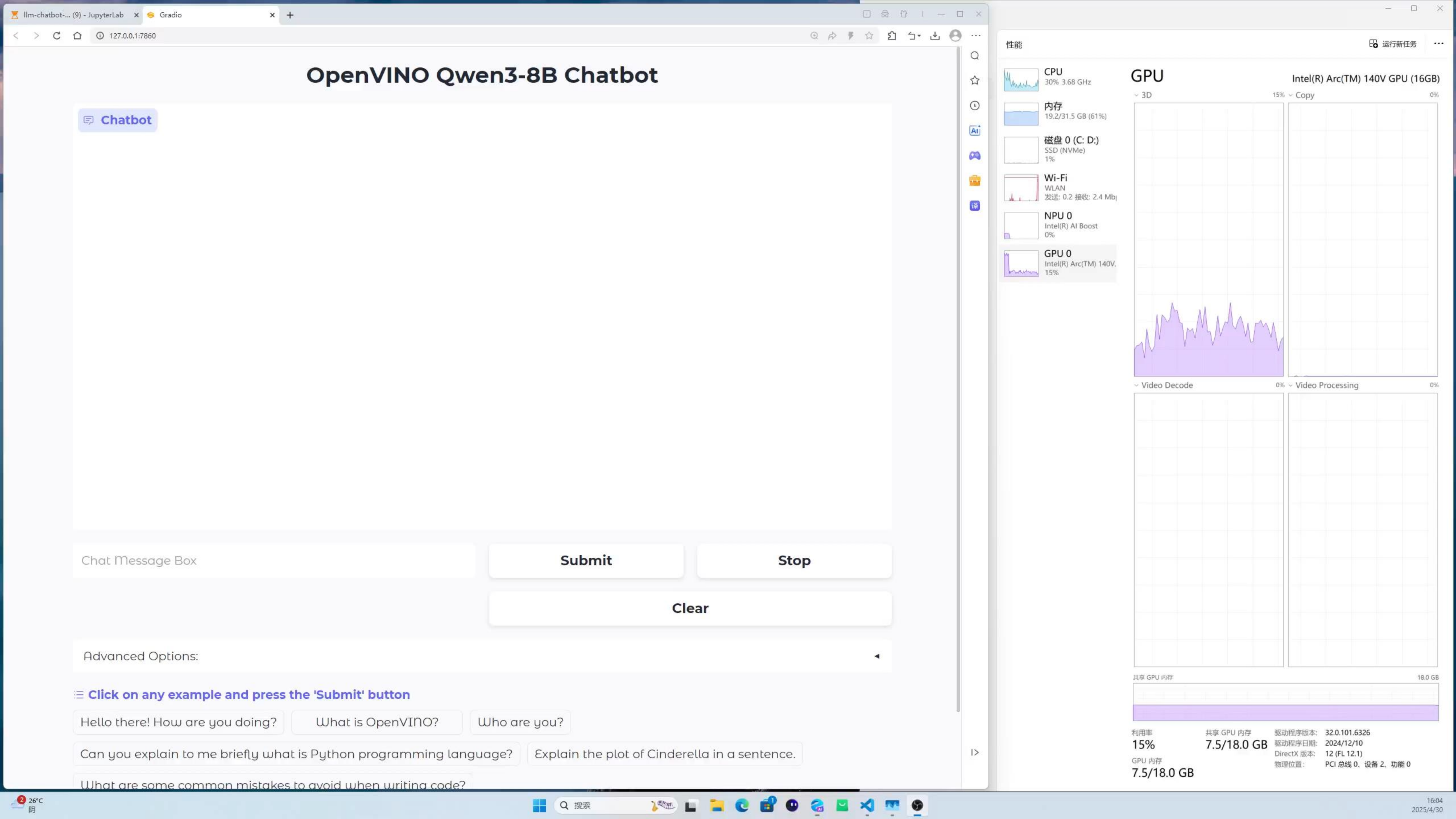
Inference

and **N**eural
Network

Optimization

Powered
by oneAPI


PyTorch
TensorFlow
Keras
TensorFlow Lite
ONNX
PaddlePaddle
OpenVINO™
Optimized Performance
CPU
intel ATOM
intel CORE
intel XEON
intel CORE ULTRA
arm
GPU
intel iRIS Xe GRAPHICS
intel ARC GRAPHICS
intel DATA CENTER GPU FLEX SERIES
intel DATA CENTER GPU MAX SERIES
NPU
intel CORE ULTRA
FPGA
intel FPGA AI Suite
Windows
Linux
macOS


llm-chatbot-... (9) - JupyterLab
Gradio
127.0.0.1:7860
OpenVINO Qwen3-8B Chatbot
Chatbot
Chat Message Box
Submit
Stop
Clear
Advanced Options:
Click on any example and press the 'Submit' button
Hello there! How are you doing?
What is OpenVINO?
Who are you?
Can you explain to me briefly what is Python programming language?
Explain the plot of Cinderella in a sentence.
What are some common mistakes to avoid when writing code?
性能
运行新任务
CPU
30% 3.68 GHz
内存
19.2/31.5 GB (61%)
磁盘 0 (C: D:)
SSD (NVMe)
1%
Wi-Fi
WLAN
发送: 0.2 接收: 2.4 Mb
NPU 0
Intel(R) AI Boost
0%
GPU 0
Intel(R) Arc(TM) 140V
15%
GPU
Intel(R) Arc(TM) 140V GPU (16GB)
3D
15%
Copy
0%
Video Decode
0%
Video Processing
0%
共享 GPU 内存
18.0 GB
利用率
15%
共享 GPU 内存
7.5/18.0 GB
GPU 内存
7.5/18.0 GB
驱动程序版本: 32.0.101.6326
驱动程序日期: 2024/12/10
DirectX 版本: 12 (FL 12.1)
物理位置: PCI 总线 0、设备 2、功能 0
26°C
阴
搜索
16:04
2025/4/30

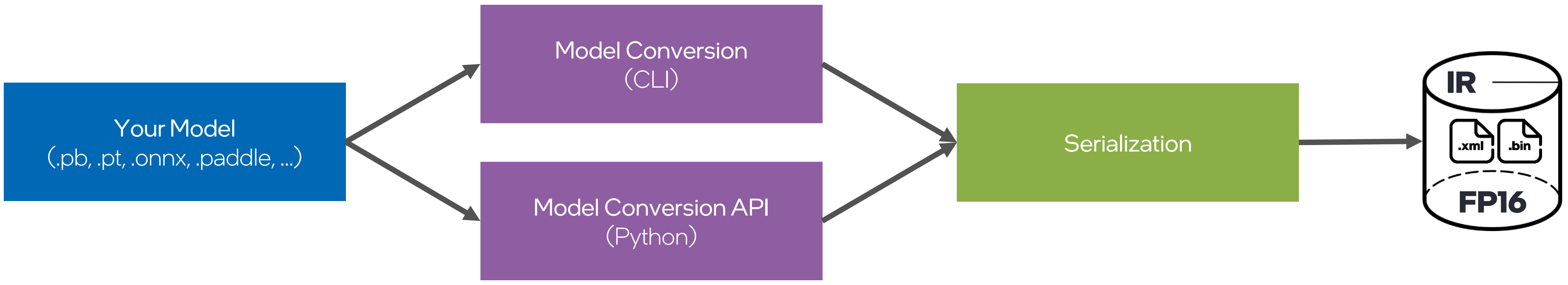
Installation

```
pip install openvino
```

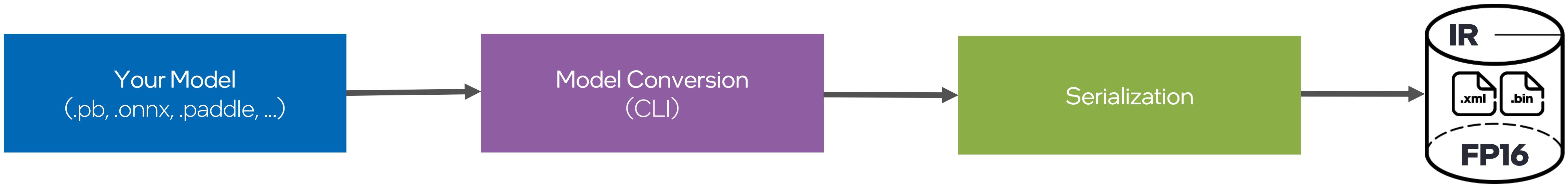
ubuntu
Red Hat
docker
ANACONDA
CONDA-FORGE
CONAN
C/C++ Package Manager
npm

www.openvino.ai

OpenVINO™ Model Converter

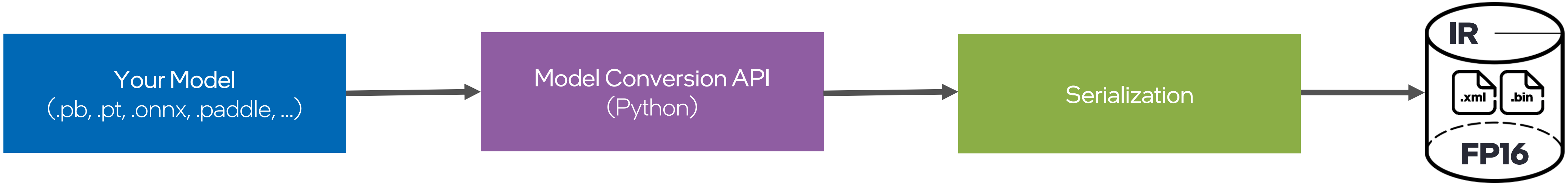


OpenVINO™ Model Converter



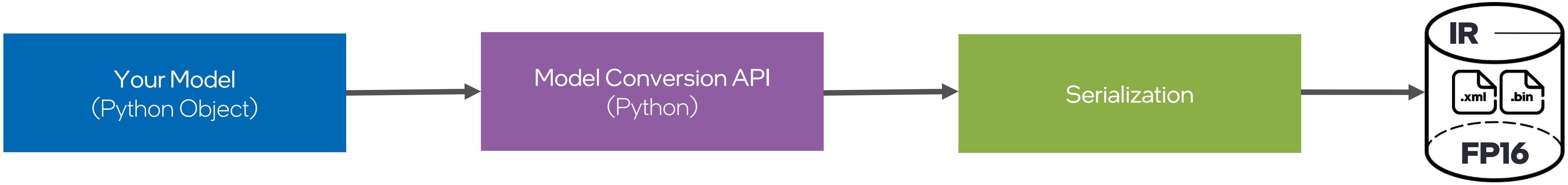

```
ovc "model.onnx" --input "1,3,224,224"
```

OpenVINO™ Model Converter




```
ov_model = ov.convert_model("model.onnx",
                            input={"input1": [1,3,224,224]})

ov.save_model(ov_model, "converted_model.xml")
```

OpenVINO™ Model Converter

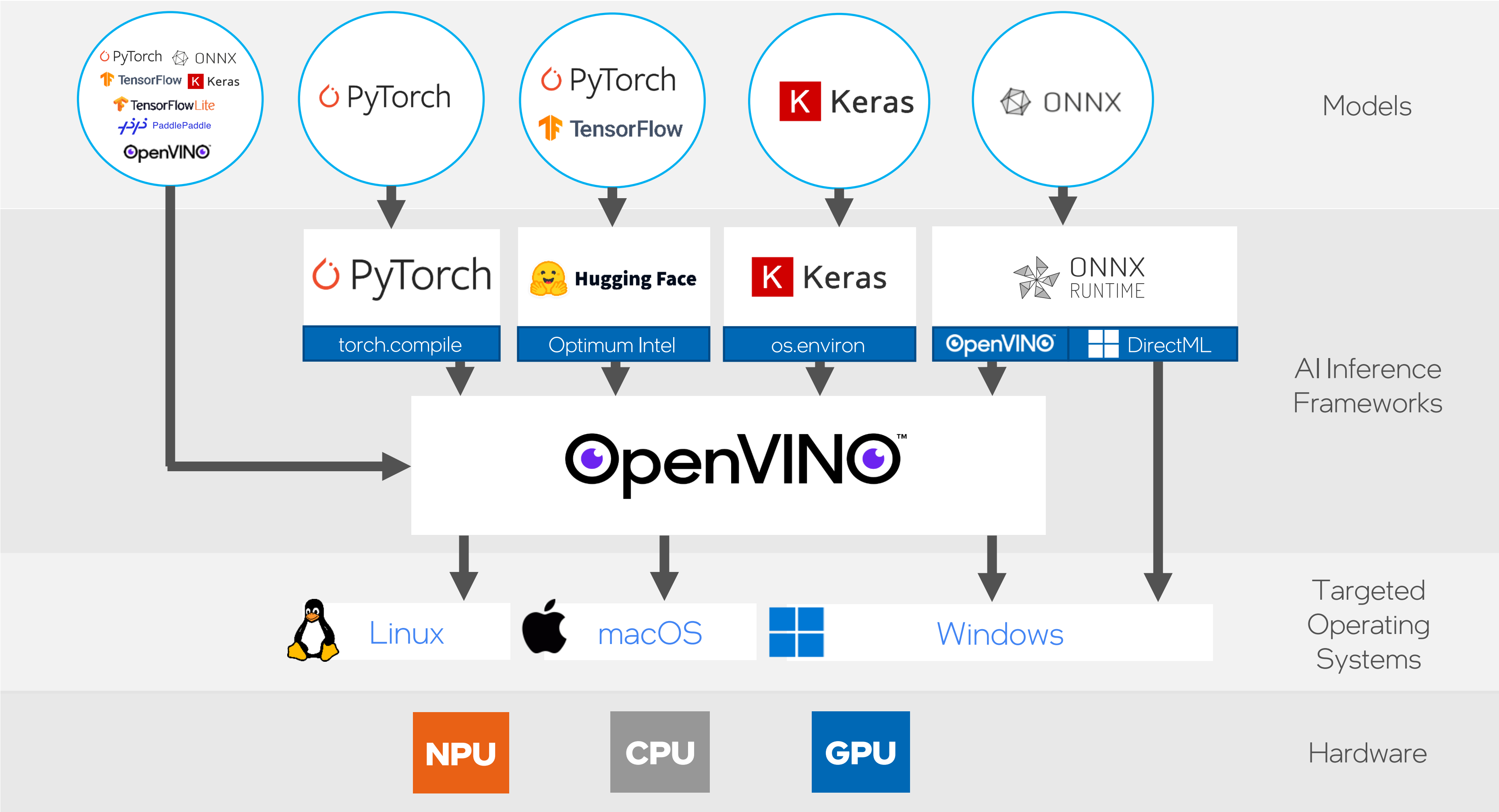



```
pytorch_model = torchvision.models.resnet50(pretrained=True)

ov_model = ov.convert_model(pytorch_model,
                            input={"input1": [1,3,224,224]})

ov.save_model(ov_model, "converted_model.xml")
```

OpenVINO™ Runtime

```python
from openvino import runtime as ov

img = load_img()

core = ov.Core()

model = core.read_model(model="model.xml")
compiled_model = core.compile_model(model=model, device_name="CPU")

output_layer = compiled_model.outputs[0]

result = compiled_model(img)[output_layer]
```

OpenVINO™ as Backend



OpenVINO™ Integration with Optimum

Combine the convenience of Hugging Face API with the efficiency of OpenVINO™!

OpenVINO™ Integration with Optimum

```
pip install optimum-intel[openvino,nncf]
```

```diff
- from transformers import AutoModelForCausalLM
+ from optimum.intel import OVModelForCausalLM
  from transformers import AutoTokenizer, pipeline

  model_id = "OpenVINO/TinyLlama-1.1B-Chat-v1.0-int8-ov"

- model = AutoModelForCausalLM.from_pretrained(model_id)
+ model = OVModelForCausalLM.from_pretrained(model_id)

  tokenizer = AutoTokenizer.from_pretrained(model_id)

  pipe = pipeline("text-generation", model=model, tokenizer=tokenizer)

  results = pipe("He's a dreadful magician and")
```

OpenVINO™ Integration with Optimum

	Hugging Face through OpenVINO	OpenVINO Native API
Model support	Broad set of Models	Broad set of Models
APIs	Python (Hugging Face API)	Python, C++ (OpenVINO API)
Model Format	Source Framework / OpenVINO	OpenVINO
Inference code	Hugging Face based	Custom inference pipelines
Additional dependencies	Many Hugging Face dependencies	Lightweight (e.g. numpy, etc.)
Application footprint	Large	Small
Pre/post-processing and glue code	Available at Hugging Face out-of-the-box	OpenVINO samples and notebooks
Performance	Good	Best

OpenVINO™ Integration with Optimum

Export to OpenVINO

```
optimum-cli export openvino --model gpt2 ov_model
```

```python
from optimum.intel import OVModelForCausalLM

model_id = "OpenVINO/TinyLlama-1.1B-Chat-v1.0-int8-ov"
model = OVModelForCausalLM.from_pretrained(model_id)
```

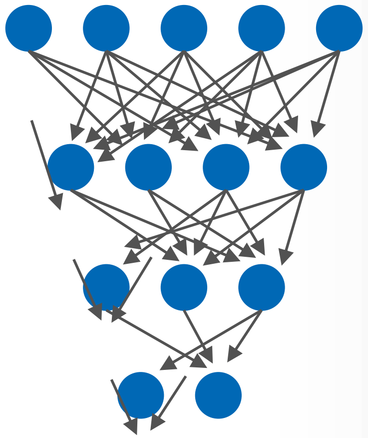
```python
from optimum.intel import OVModelForCausalLM

model_id = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"
model = OVModelForCausalLM.from_pretrained(model_id, export=True)

model.save_pretrained("ov_model")
```

Neural Network Compression Framework

(NNCF)

TensorFlow PyTorch ONNX OpenVINO

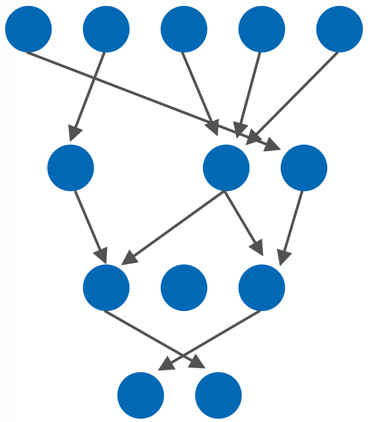
Native Model



Optimization Techniques

- Post-Training Quantization
- Accuracy-Control Quantization
- Quantization-Aware Training
- Weight Compression
- Filter Pruning
- Binarization
- Sparsity

Compressed Model



Broad model support   Easy Installation

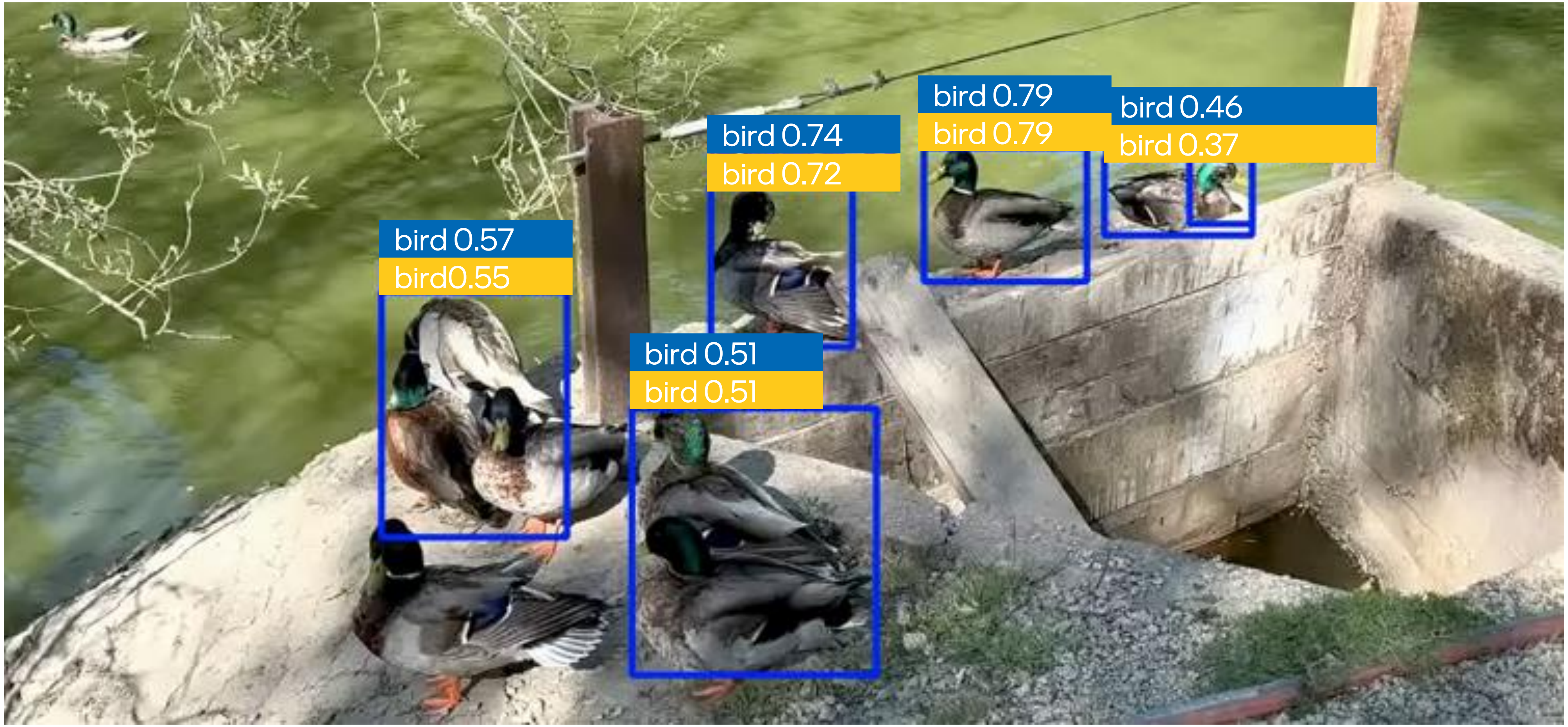
```
pip install nncf
```

Quantization Performance Optimization

YOLOv8: Minimum accuracy drop from FP32 to INT8

FP32

INT8



Quantization Performance Optimization

OpenPose: Reduced file size and faster inference with consistent accuracy

Native Model

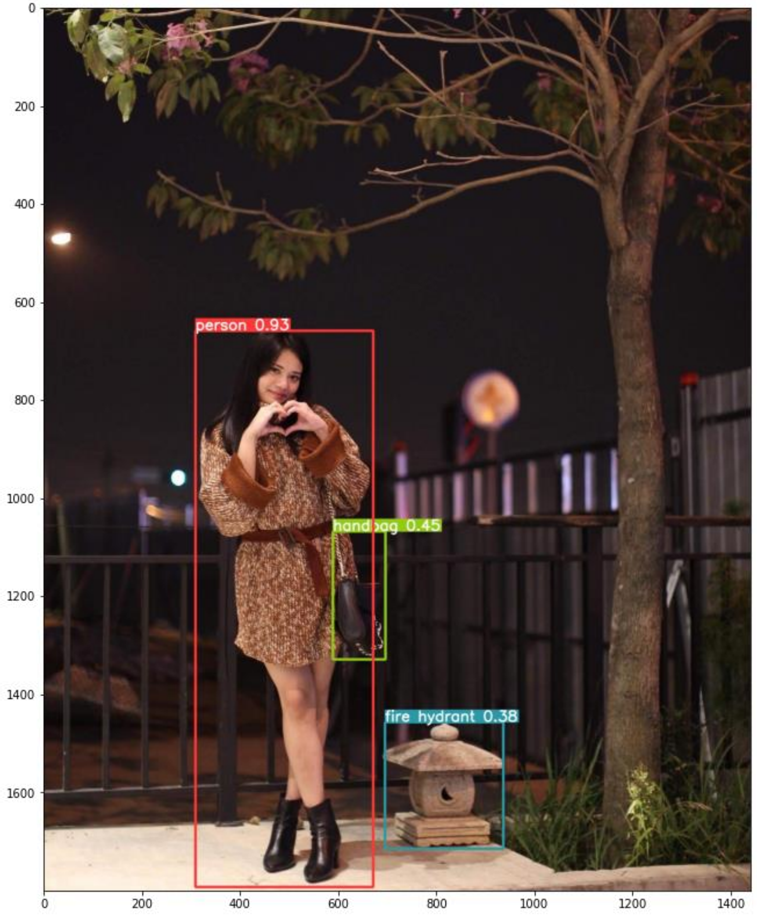
File size: 16.6 MB

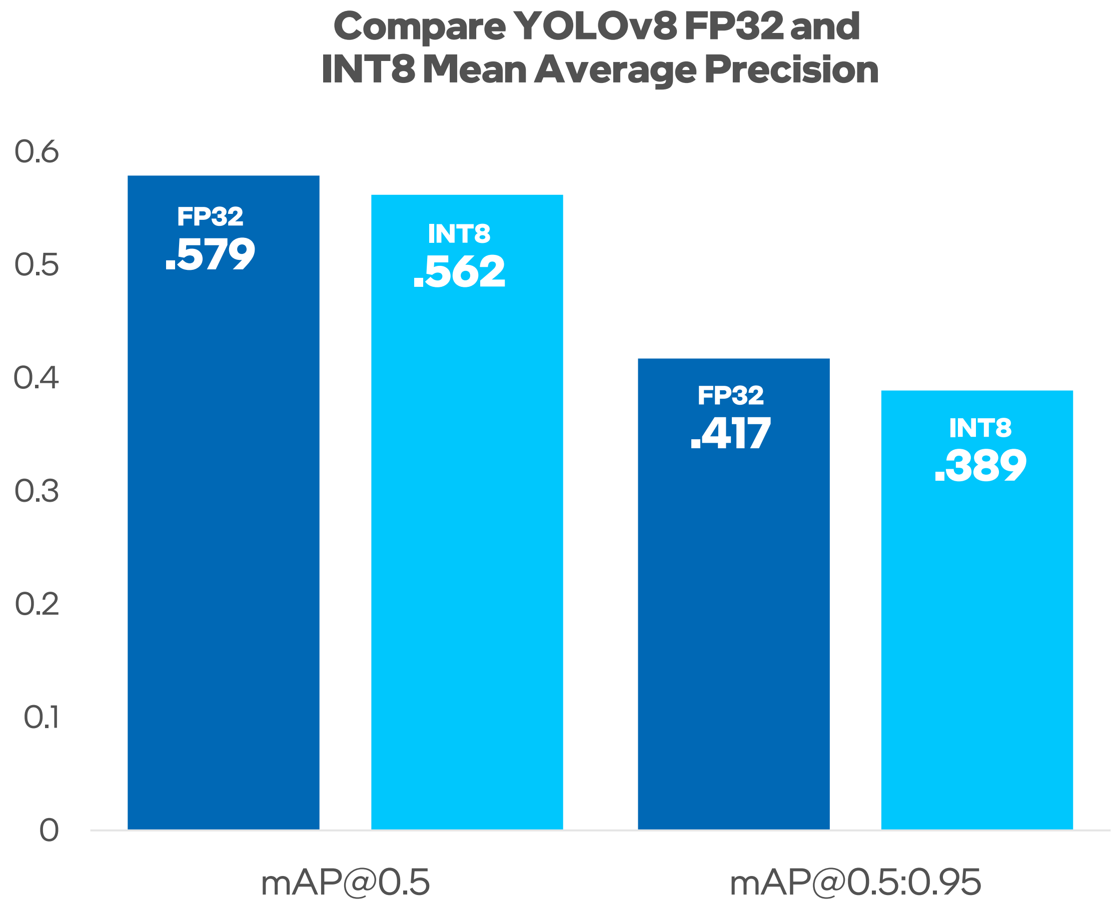
Optimized Model

File size: 4.7 MB

Post-Training Quantization

(NNCF)





Weight Compression for LLMs

(NNCF)

Model	Mode	Perplexity	Perplexity Increase	Model Size (GB)
databricks/dolly-v2-3b	fp32	5.01	0	10.3
databricks/dolly-v2-3b	int8	5.07	0.05	2.6
databricks/dolly-v2-3b	int4_asym_g32_r50	5.28	0.26	2.2
databricks/dolly-v2-3b	nf4_g128_r60	5.19	0.18	1.9
meta-llama/Llama-2-7b-chat-hf	fp32	3.28	0	25.1
meta-llama/Llama-2-7b-chat-hf	int8	3.29	0.01	6.3
meta-llama/Llama-2-7b-chat-hf	int4_asym_g128_r80	3.41	0.14	4.0
meta-llama/Llama-2-7b-chat-hf	nf4_g128	3.41	0.13	3.5
togethercomputer/RedPajama-INCITE-7B-Instruct	fp32	4.15	0	25.6
togethercomputer/RedPajama-INCITE-7B-Instruct	int8	4.17	0.02	6.4
togethercomputer/RedPajama-INCITE-7B-Instruct	nf4_ov_g32_r60	4.28	0.13	5.1
togethercomputer/RedPajama-INCITE-7B-Instruct	int4_asym_g128	4.17	0.02	3.6

Significant Reduction in RAM usage!

OpenVINO™ Integration with Optimum

Quantization with NNCF

```
from optimum.intel import OVModelForCausalLM

# INT8 quantization
quantization_config = OVWeightQuantizationConfig(bits=8)
model = OVModelForCausalLM.from_pretrained(model_id,
                                           quantization_config=quantization_config)
```

```
from optimum.intel import OVModelForCausalLM
from nncf import compress_weights, CompressWeightsMode

# INT4 quantization
quantization_config = OVWeightQuantizationConfig(bits=4, sym=True,
                                                 ratio=0.8, group_size=128)
model = OVModelForCausalLM.from_pretrained(model_id,
                                           quantization_config=quantization_config)
```

GenAI Workflow

Find

+

PyTorch

python™

Optimize

+ intel®

Optimum-Intel

+

OpenVINO™ & NNCF

python™

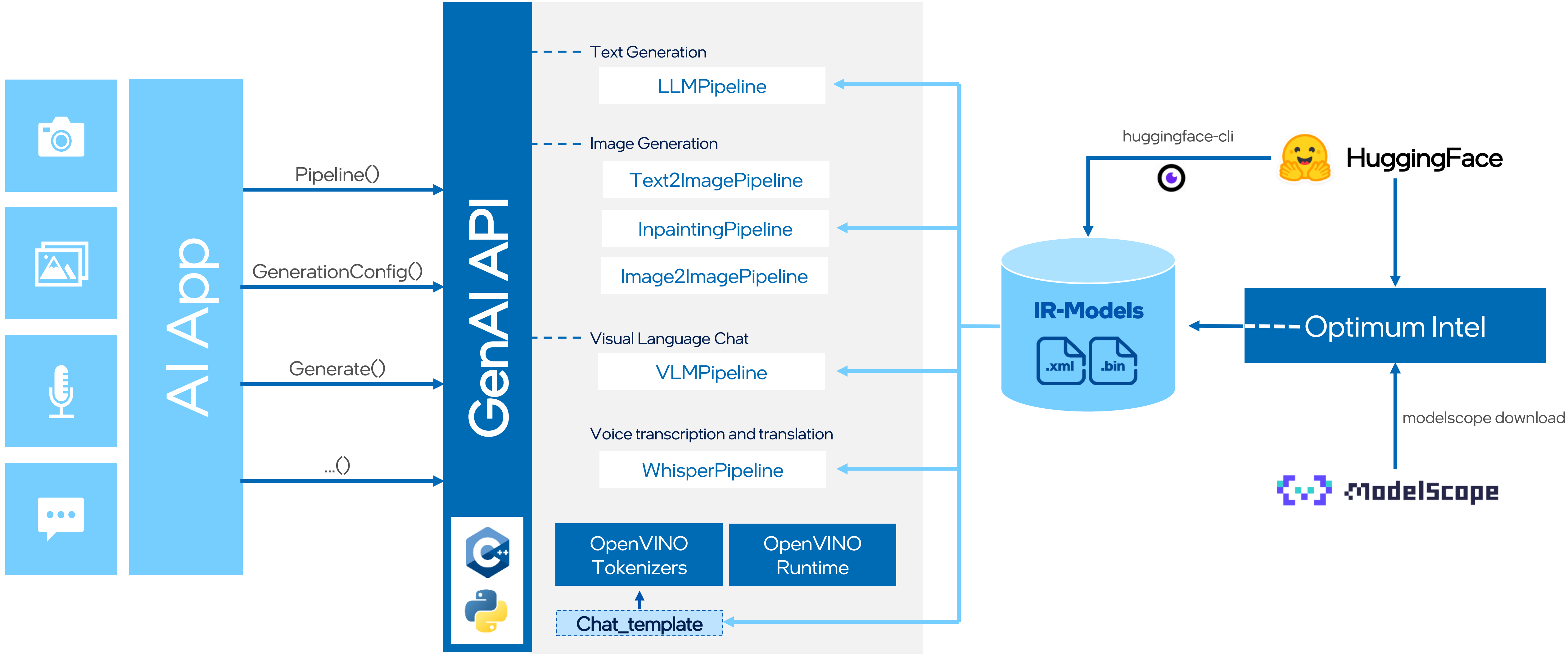
Deploy

OpenVINO GenAI

python

Efficient deployment, minimum dependency and memory footprint

OpenVINO™ GenAI



OpenVINO™ GenAI
GenAI Model Workflow

Export and optimize pipeline | **Deploy pipeline**

Hugging Face | ModelScope

```
huggingface-cli download "OpenVINO/FLUX.1-schnell-int4-ov" --local-dir model_path
```

```
optimum-cli export openvino --model meta-llama/Llama-2-7b-chat-hf --weight-format int4 ov_llama_2
```

```
modelscope download --model 'Qwen/Qwen2-7B-Instruct' --local_dir model_path
```

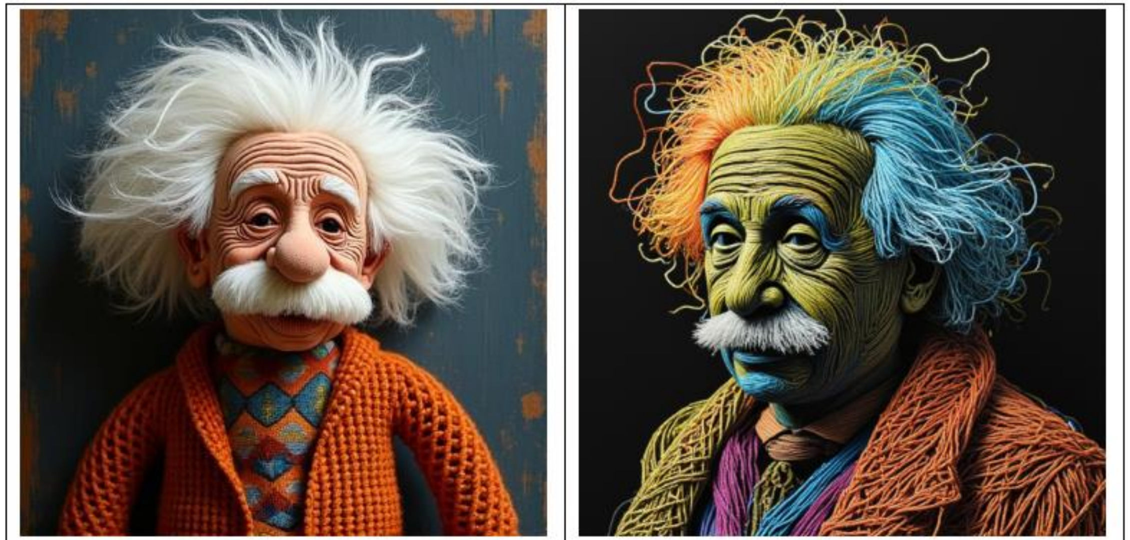
```
optimum-cli export openvino -m model_path --task text-generation-with-past --weight-format int4  ov_qwen_2
```

python™ | C++

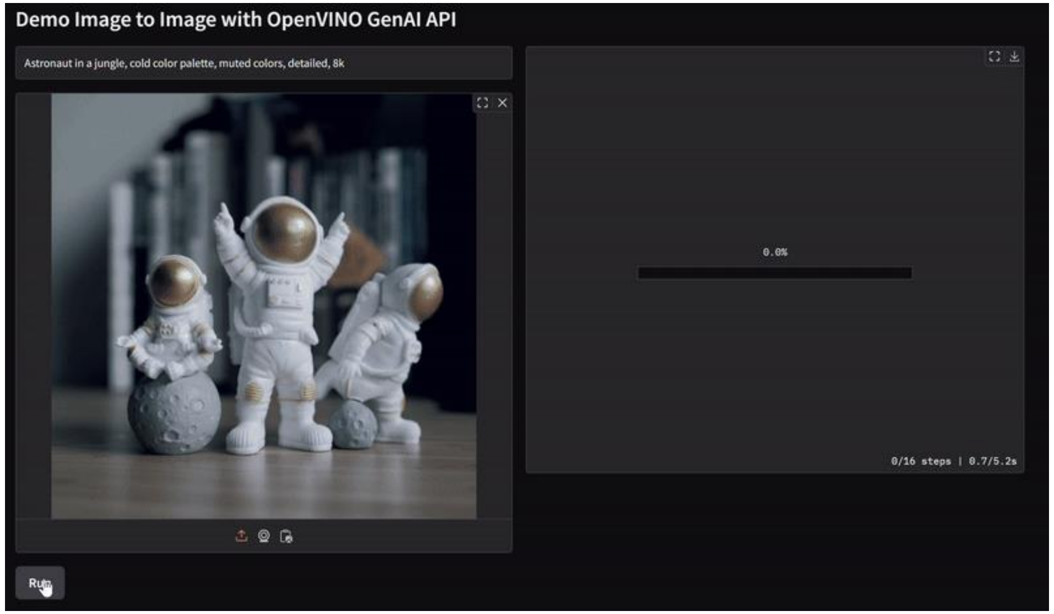
```python
import openvino_genai as ov_genai

pipe = ov_genai.LLMPipeline(model_path, "GPU")
print(pipe.generate("What is OpenVINO?", max_length=200))
```

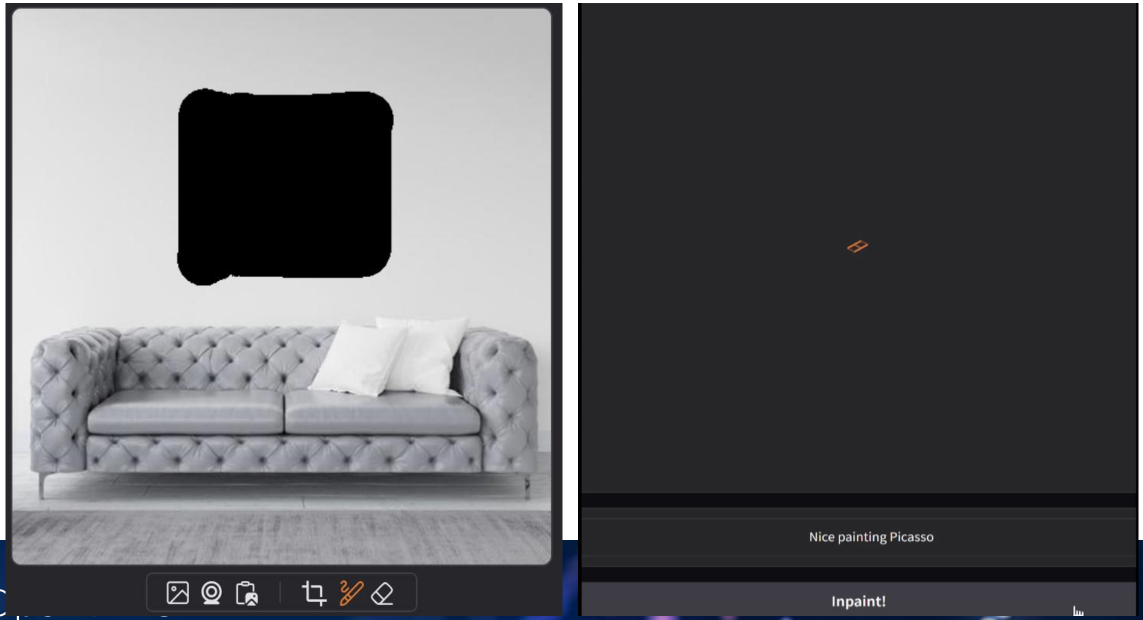
```cpp
#include "openvino/genai/llm_pipeline.hpp"
#include <iostream>

int main(int argc, char* argv[]) {
    ov::genai::LLMPipeline pipe(model_path, "GPU");
    std::cout << pipe.generate("What is Large Language Model?",
                               ov::genai::max_new_tokens(200));
}
```

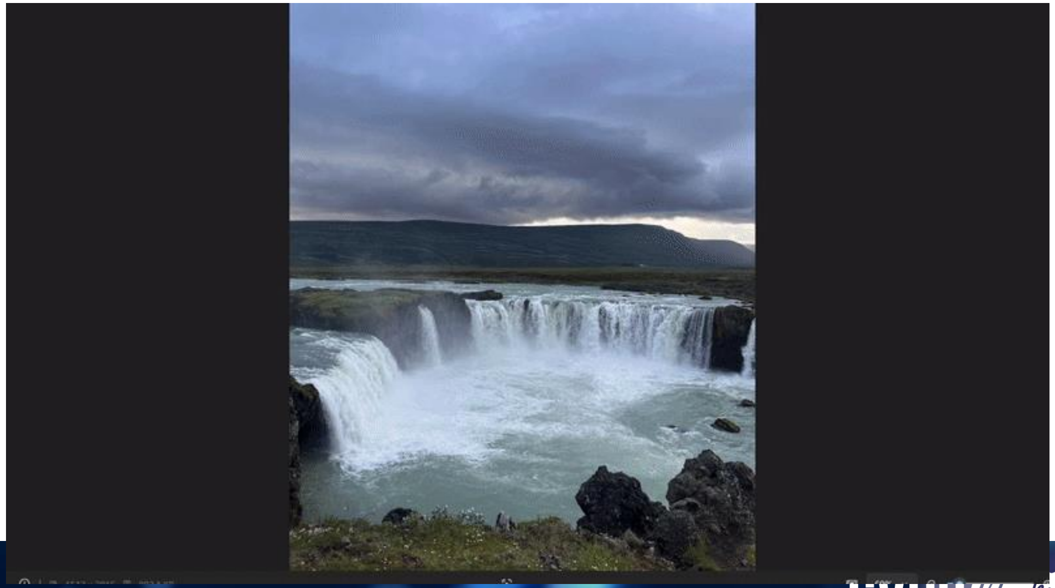
More with OpenVINO™ GenAI



Without LoRA | With LoRA (Alpha=1)







Benefits of Building Applications with OpenVINO™

Build and deploy AI applications in simple steps

Faster inference speed

Maximize AI performance across CPU, GPU, NPU

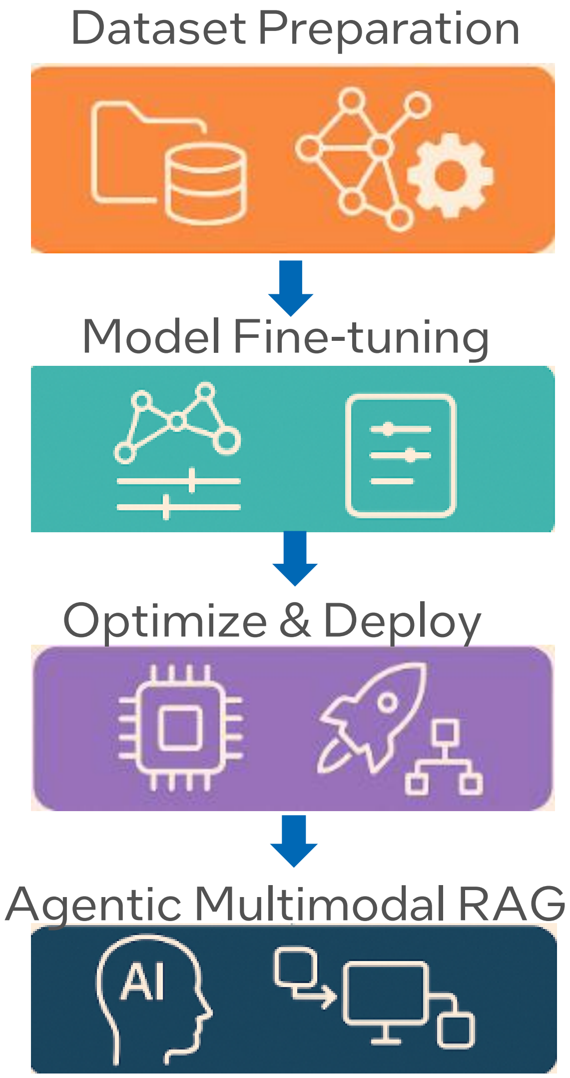
Smaller model and binary size

Reduce memory footprint

Ability to scale to many nodes with serving

What's next?

What You Will Learn



- Module 1: Tool for accelerating your dataset preparation locally
- Module 2: How to fine-tune the multimodal embedding model and VLM
- Module 3: How to optimize and deploy the multimodal RAG pipeline
- Module 4: How to build Agentic multimodal RAG