intel

OpenVINO™

CVPR 2024

Edge-Optimized Deep Learning: Harnessing Generative AI and Computer Vision with Open-Source Libraries

Fundamentals

Paula Ramos
paula.ramos@intel.com
AI Software Evangelists

Adrian Boguszewski
adrian.boguszewski@intel.com
AI Software Evangelists

Raymond Lo
raymond.lo@intel.com

Zhuo Wu
zhuo.wu@intel.com

Ria Cheruvu

USA
(Phoenix, Arizona)

MSc in Data Science

Exp. 5+ years as
AI SW Lead Architect and
Research Engineer

Paula Ramos

USA
(Raleigh, North Carolina)

PhD in
Engineering

Exp. 17+ years
as researcher

Dmitriy Pastushenkov

EMEA
(Karlsruhe, Germany)

MSc in
Computer Science

Exp. 17+ years as Software
Engineer and Architect

Adrian Boguszewski

EMEA
(Warsaw, Poland)

MSc in
Computer Science

Exp. 5+ years as
DL Engineer

Anisha Udayakumar

APJ
(Bengaluru, India)

BTech in
Civil Engineering

Exp. 5+ years as
Innovation Consultant

Zhuo Wu

PRC
(Shanghai, China)

PhD in
Electronics

Exp. 15+ years as
researcher and professor

Raymond Lo

Global Lead
(Santa Clara – HQ, California)

PhD in Computer
Engineering

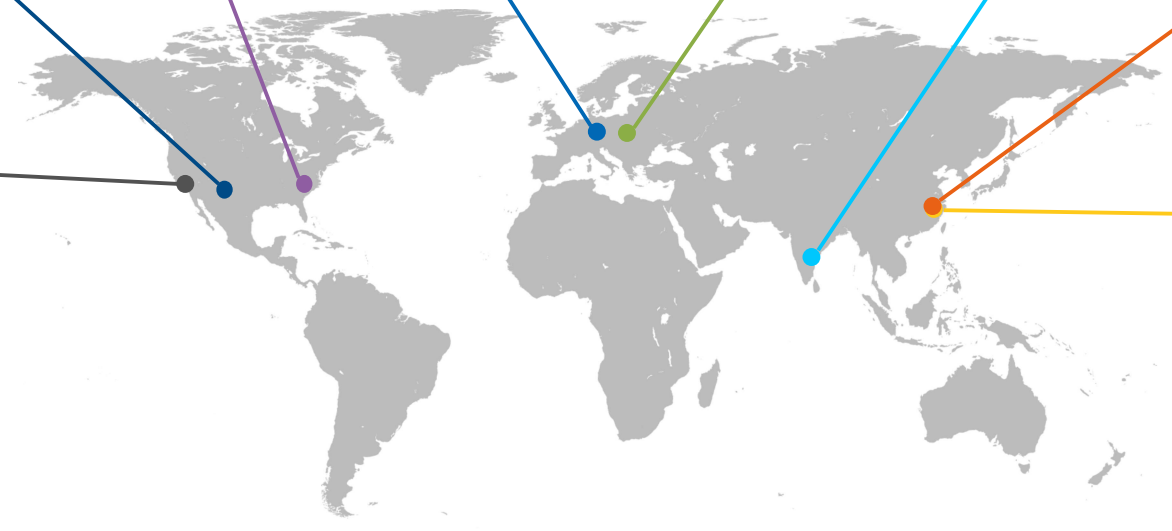
Exp. 9+ as Entrepreneur
Executive and Evangelist

Ethan Yang

PRC
(Shanghai, China)

MSc in Electronics

Exp 5+ years as
AI Solution Engineer



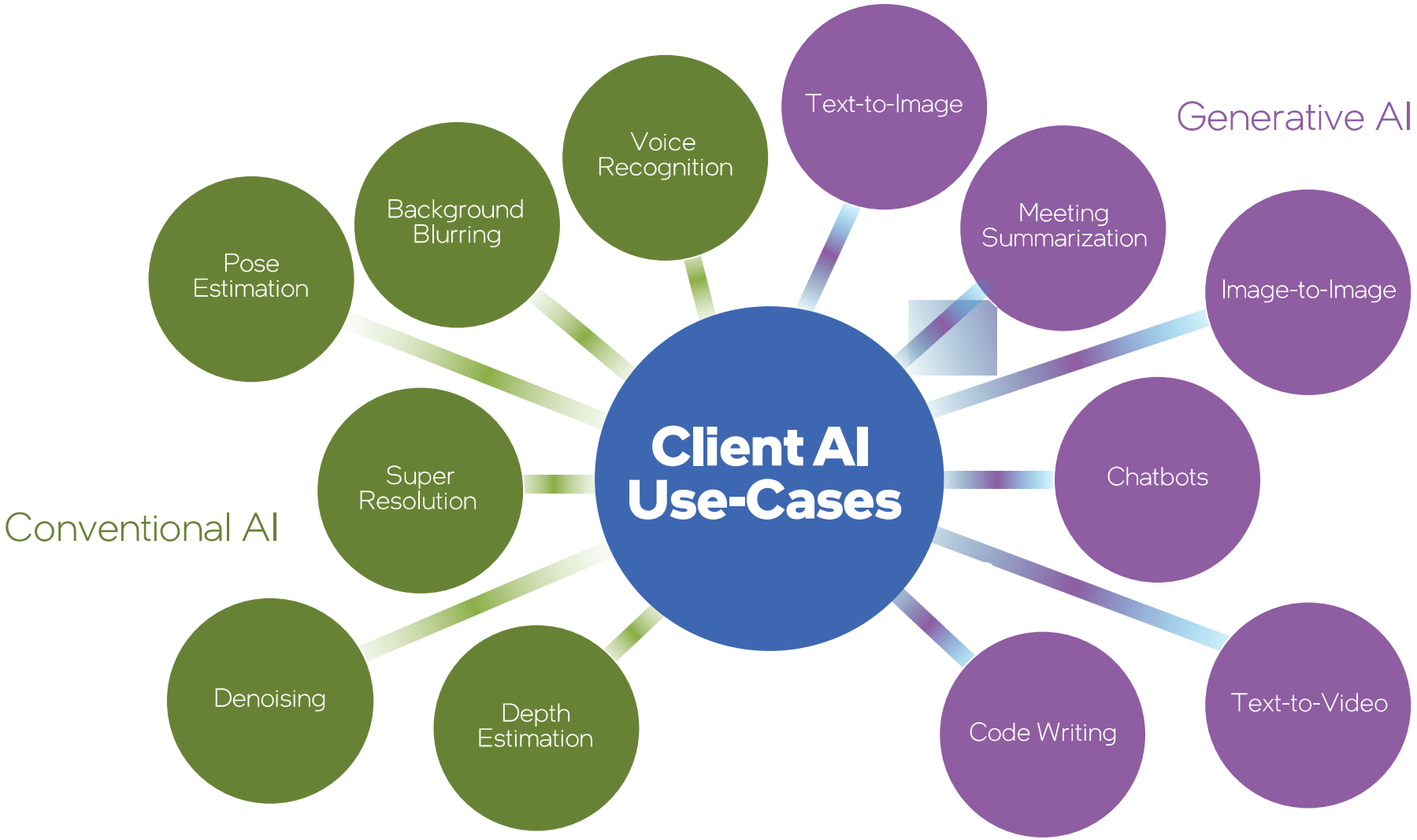
OpenVINO

intel

The Challenge

OpenVINO

intel


Client AI Use-Cases
Conventional AI
Pose Estimation
Background Blurring
Voice Recognition
Super Resolution
Denoising
Depth Estimation
Generative AI
Text-to-Image
Meeting Summarization
Image-to-Image
Chatbots
Code Writing
Text-to-Video

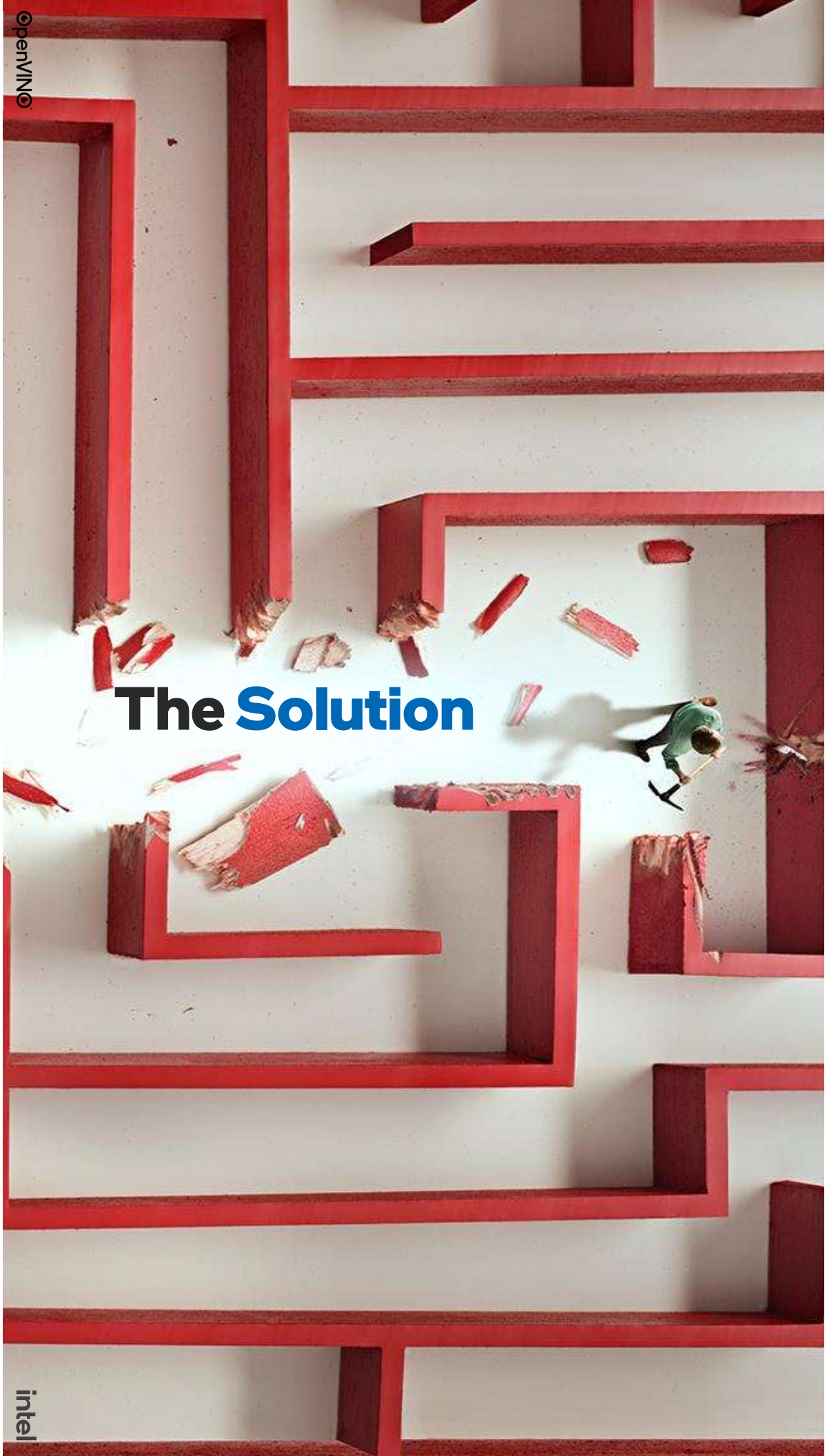
Conventional and Generative AI Comparison

Model Type	Goal	Model Size	User Interactions	Latency
Conventional AI	Output predication based on input and the model	From thousands to hundreds of millions of parameters	Inference runs continuously in the background in many cases	Low latency
Generative AI	Generates unique content: text, code, music	From single digit billions to trillions of parameters	Users interact with the model	Higher latency is accepted

How Do I Optimize and Deploy My AI Solutions?

OpenVINO™

intel.


OpenVINO
The Solution
intel

OpenVINO™

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

Inference

and
Neural
Network

Optimization

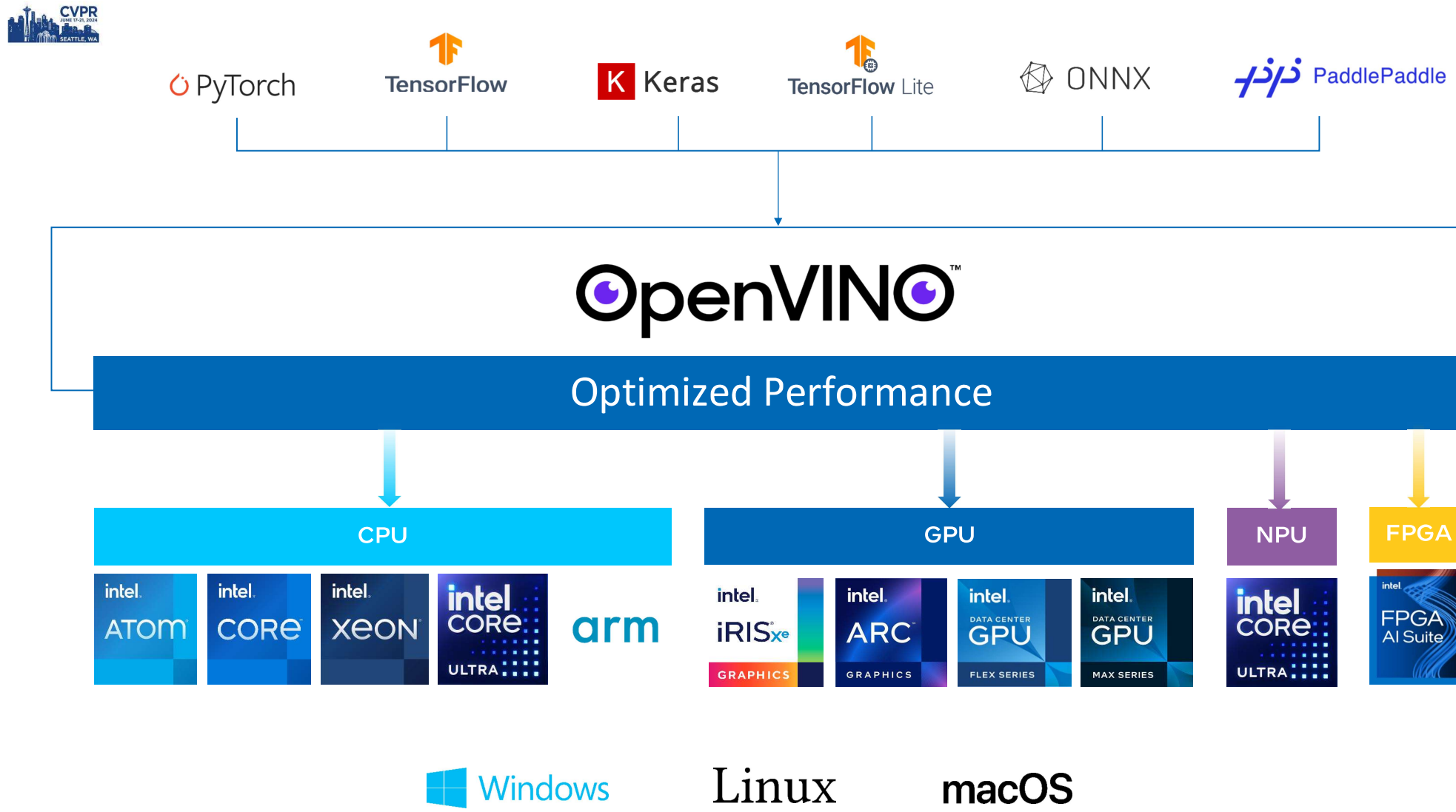
Powered by oneAPI

CVPR JUNE 17-21, 2024 SEATTLE, WA

OpenVINO™

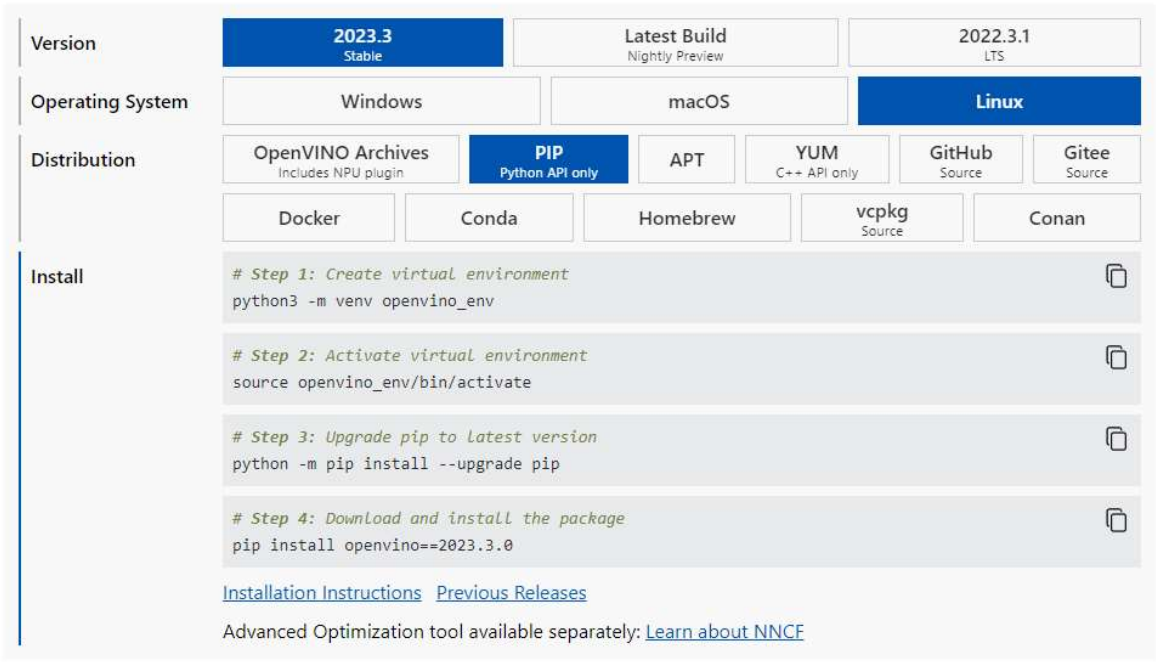
DEVELOPER JOURNEY

1 MODEL 2 OPTIMIZE 3 DEPLOY

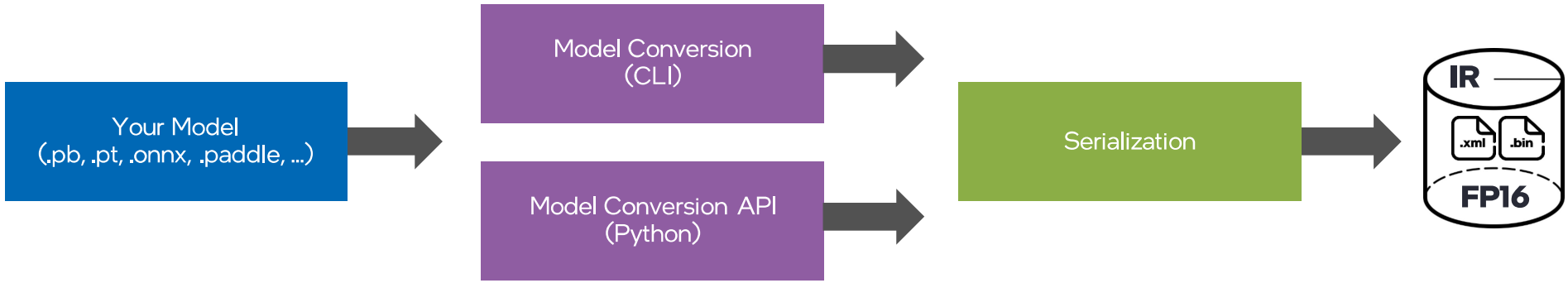

CVPR
PyTorch
TensorFlow
Keras
TensorFlow Lite
ONNX
PaddlePaddle
OpenVINO™
Optimized Performance
CPU
GPU
NPU
FPGA
intel ATOM
intel CORE
intel XEON
intel CORE ULTRA
arm
intel iRISXe GRAPHICS
intel ARC GRAPHICS
intel DATA CENTER GPU FLEX SERIES
intel DATA CENTER GPU MAX SERIES
intel CORE ULTRA
intel FPGA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

Installation

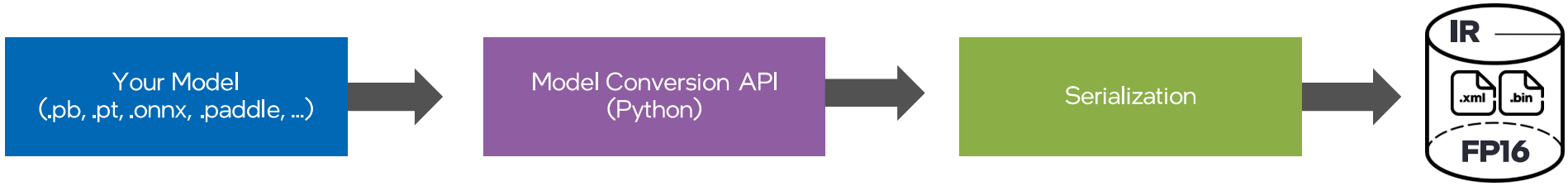
Install OpenVINO™ 2023.3



OpenVINO™ Model Converter

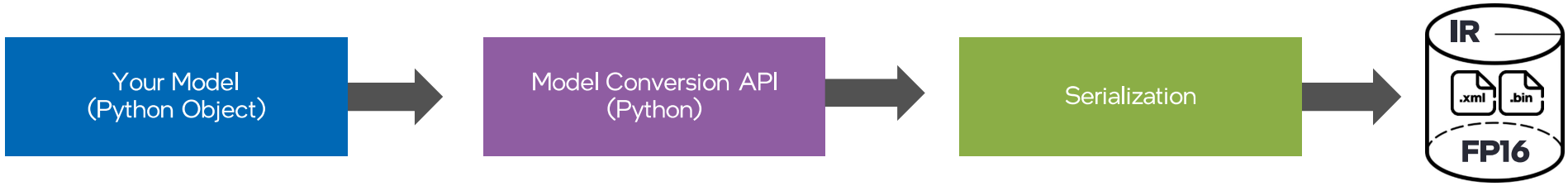


OpenVINO™ Model Converter



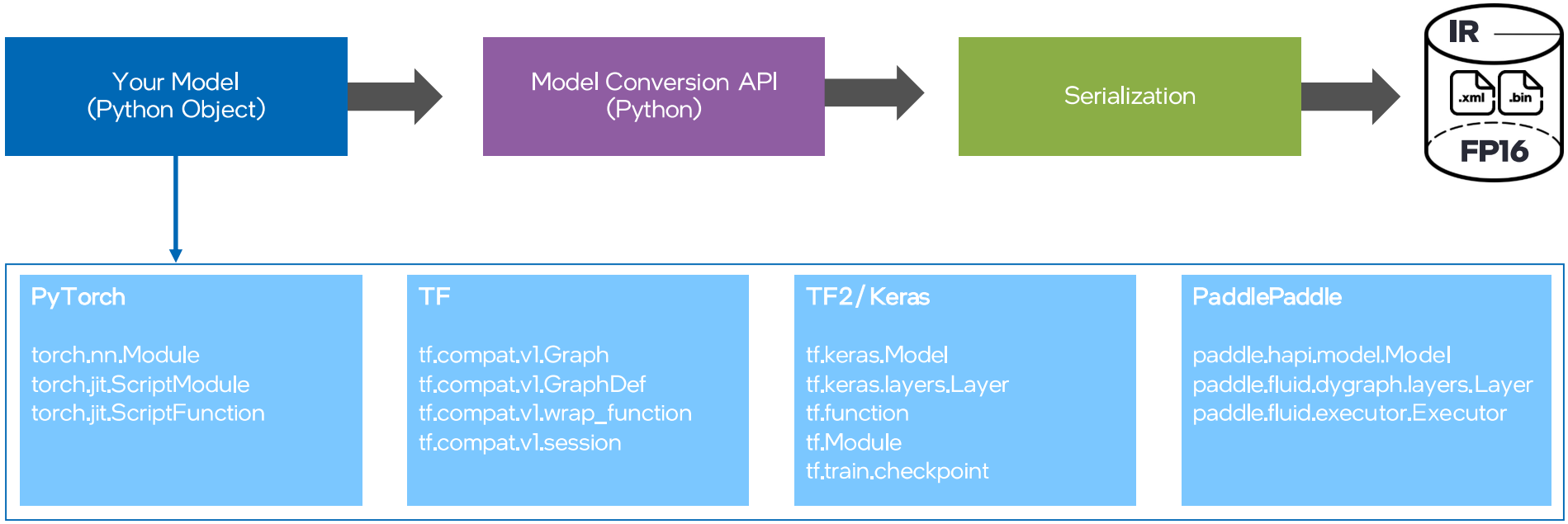

```
ov_model = ov.convert_model("model.onnx",
                            input={"input1": [1,3,224,224]})

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™ Model Converter



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™ Runtime

Automatic Conversion

```
from openvino import runtime as ov

img = load_img()

core = ov.Core()
# PyTorch, Tensorflow, TF Lite, ONNX, PaddlePaddle frontend
model = core.read_model(model="model.pt")
compiled_model = core.compile_model(model=model, device_name="CPU")

output_layer = compiled_model.outputs[0]

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

Supported Devices

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

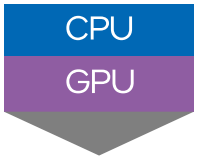
Supported Devices

CPU

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

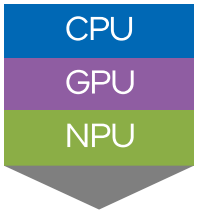
arm

Supported Devices




```
compiled_model = core.compile_model(model=model, device_name="GPU")
```

Supported Devices

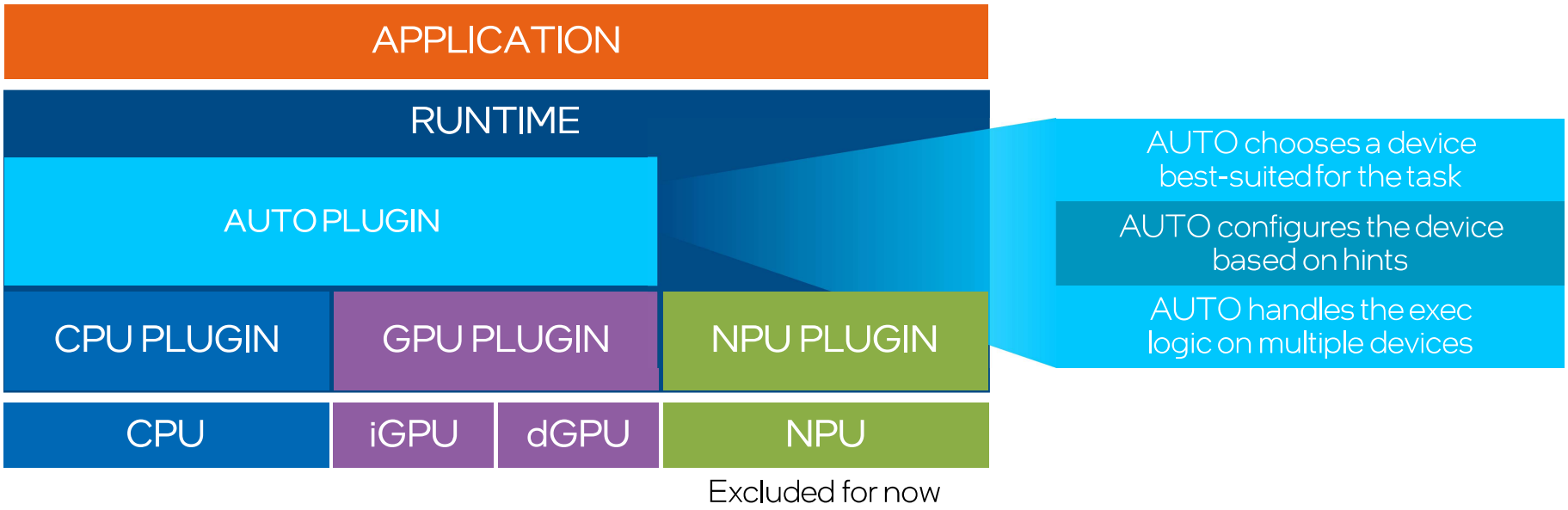



```
compiled_model = core.compile_model(model=model, device_name="NPU")
```

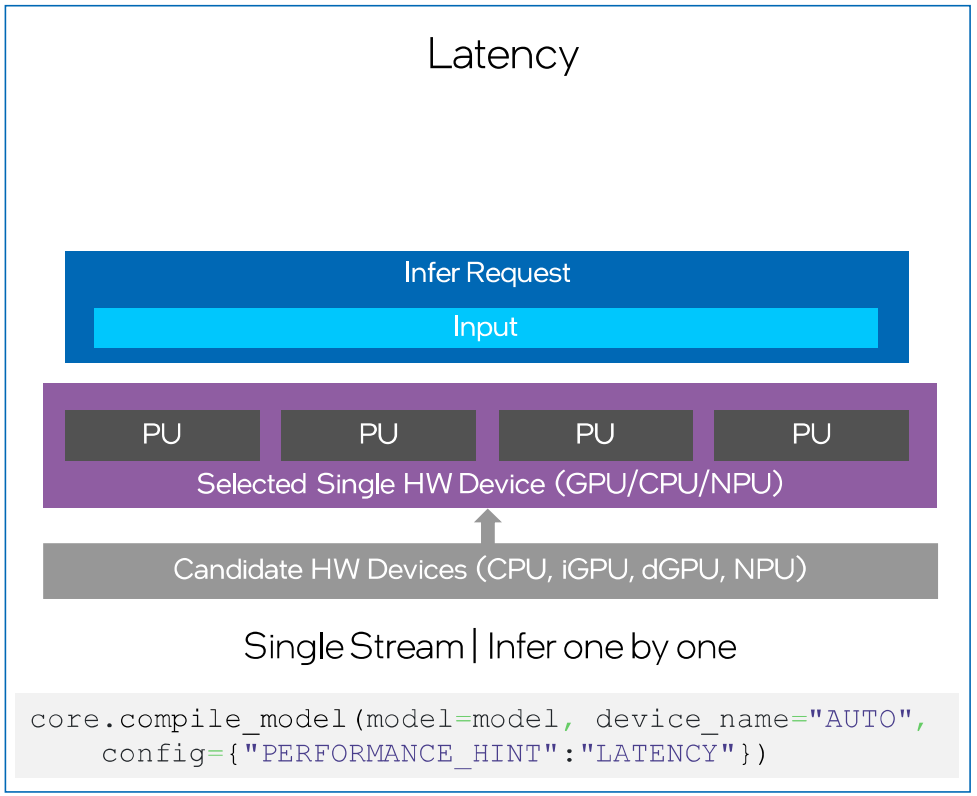
AUTO Device

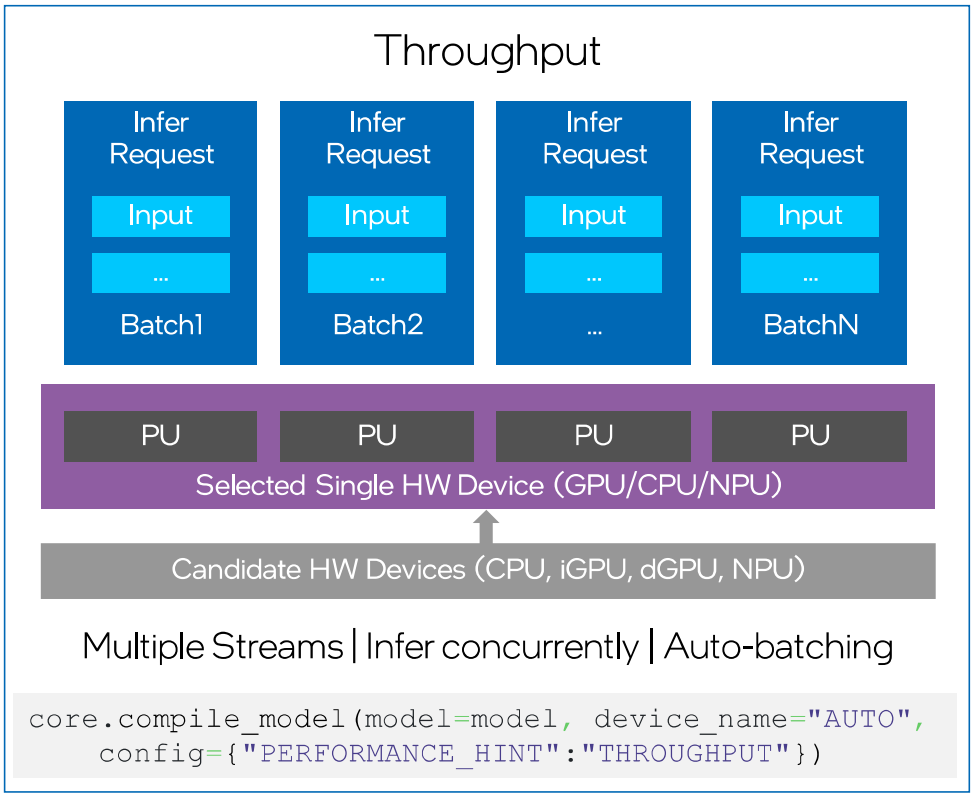
CPU
GPU
NPU
AUTO

```
compiled_model = core.compile_model(model=model, device_name="AUTO")
```



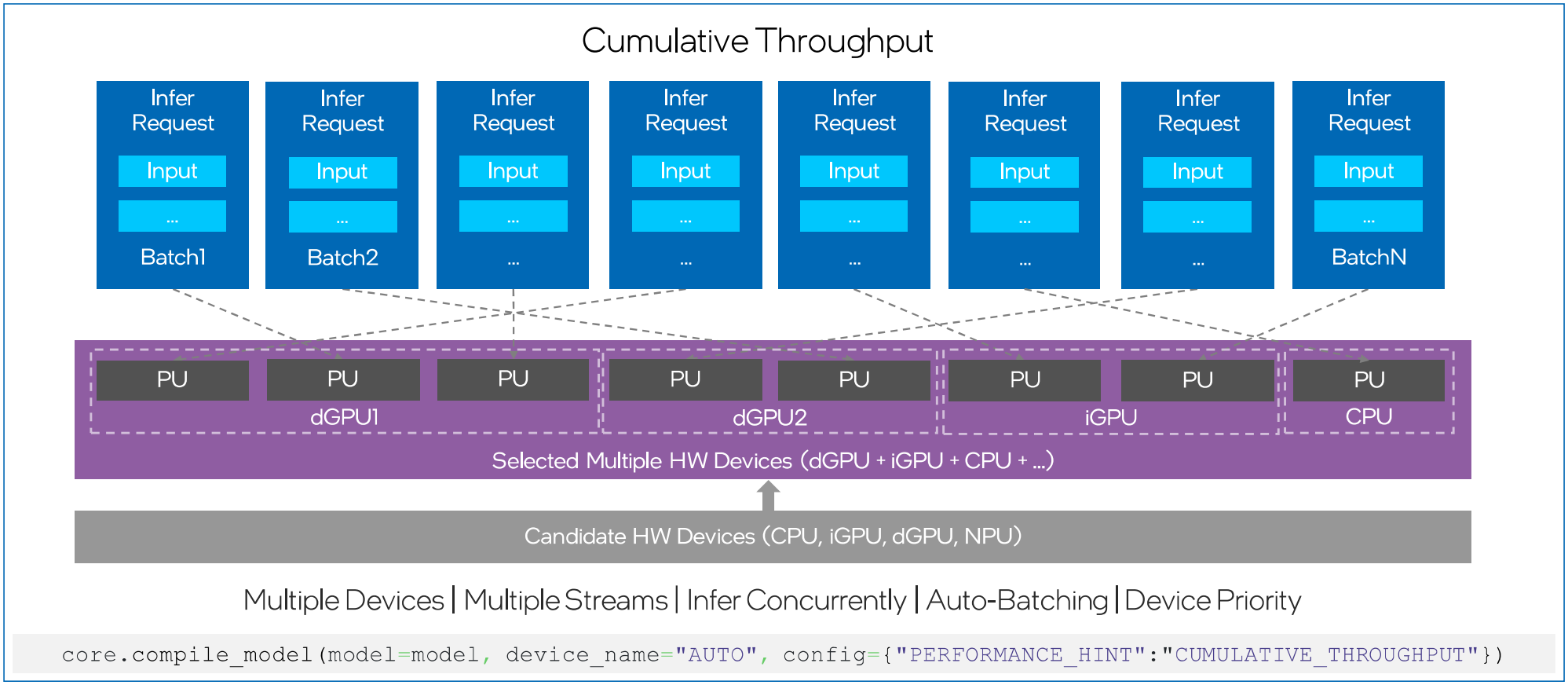
Performance Hints





PU = Processing Unit

Performance Hints



Multiple Devices | Multiple Streams | Infer Concurrently | Auto-Batching | Device Priority

```
core.compile_model(model=model, device_name="AUTO", config={"PERFORMANCE_HINT":"CUMULATIVE_THROUGHPUT"})
```

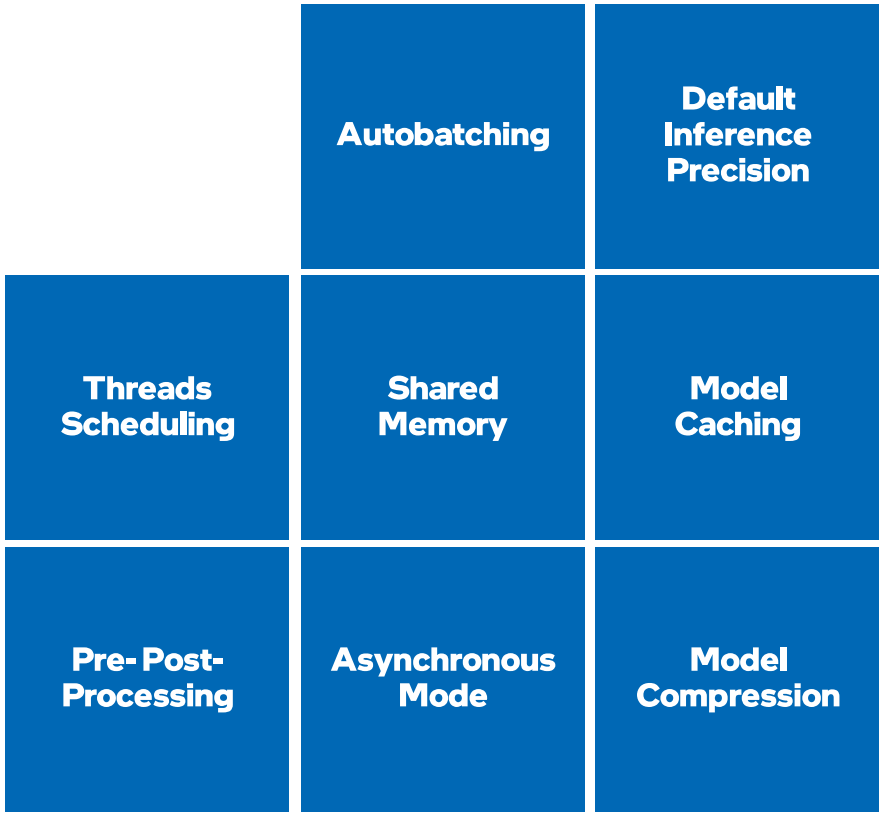
PU = Processing Unit

OpenVINO™ Runtime

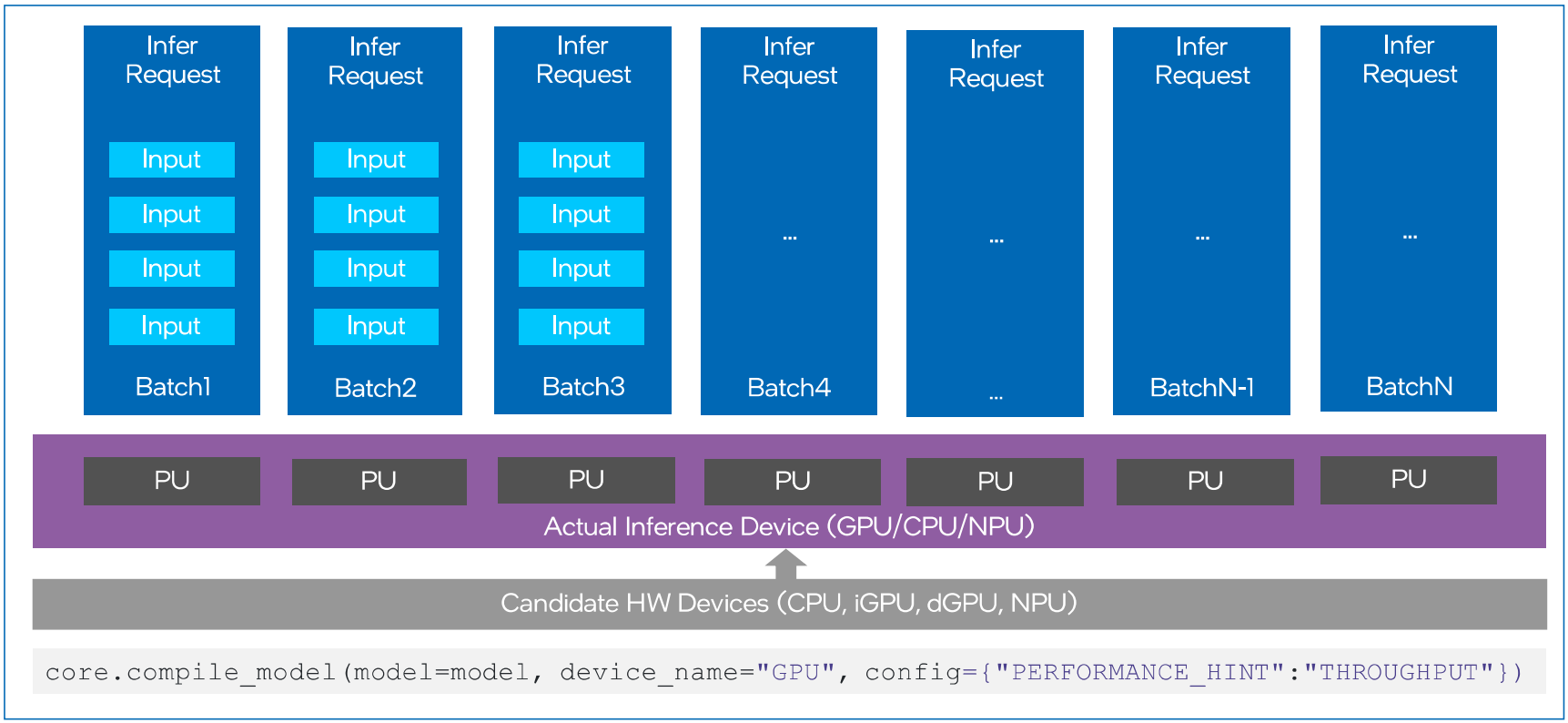
AUTO Plugin – Cumulative Throughput Mode

```
from openvino import runtime as ov

img = load_img()

core = ov.Core()

model = core.read_model(model="model.xml")
# utilize MULTIPLE GPUs
compiled_model = core.compile_model(model,
                 device_name="AUTO:GPU.1,GPU.0,CPU",
                 # device_name="AUTO:-CPU",
                 config={"PERFORMANCE_HINT":"CUMULATIVE_THROUGHPUT"})

output_layer = compiled_model.outputs[0]

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

Software Optimizations



AUTO Batching



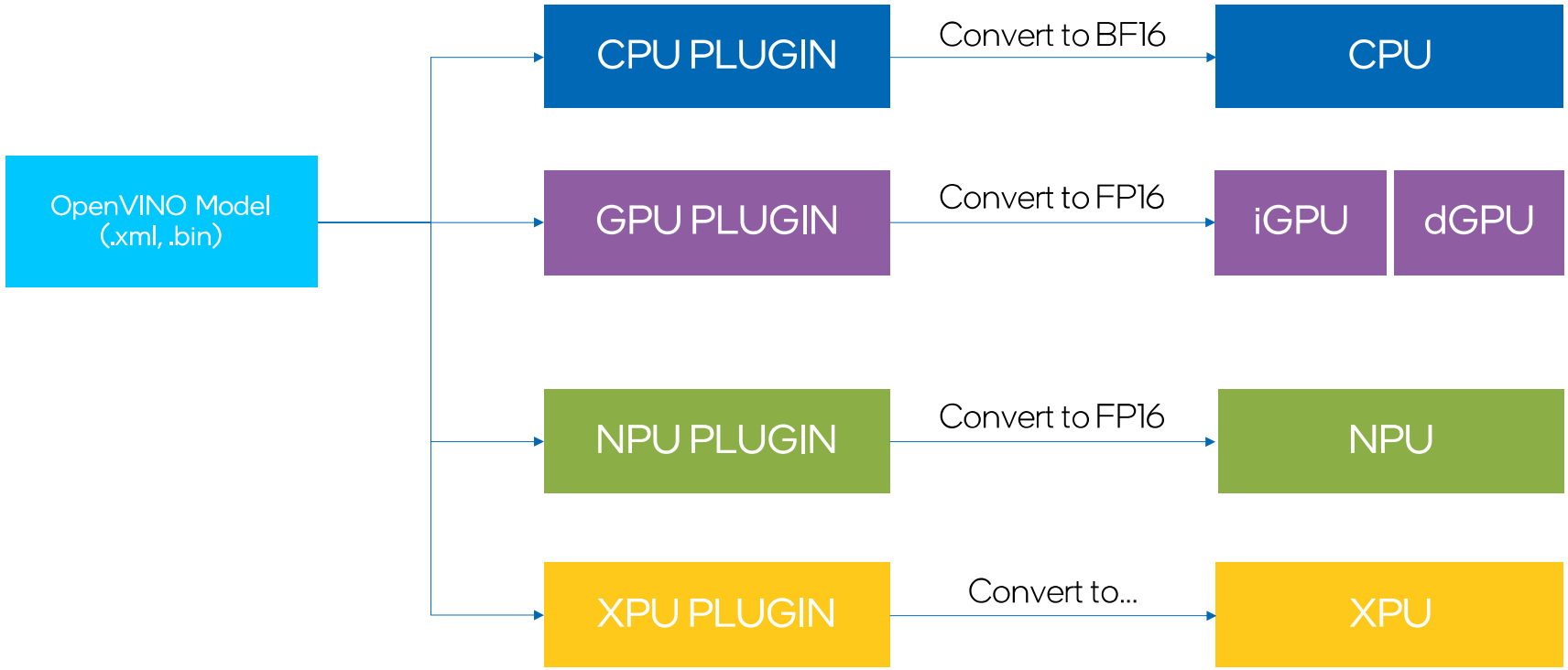

```
core.compile_model(model=model, device_name="GPU", config={"PERFORMANCE_HINT":"THROUGHPUT"})
```

OpenVINO™ Runtime

AUTO Batching

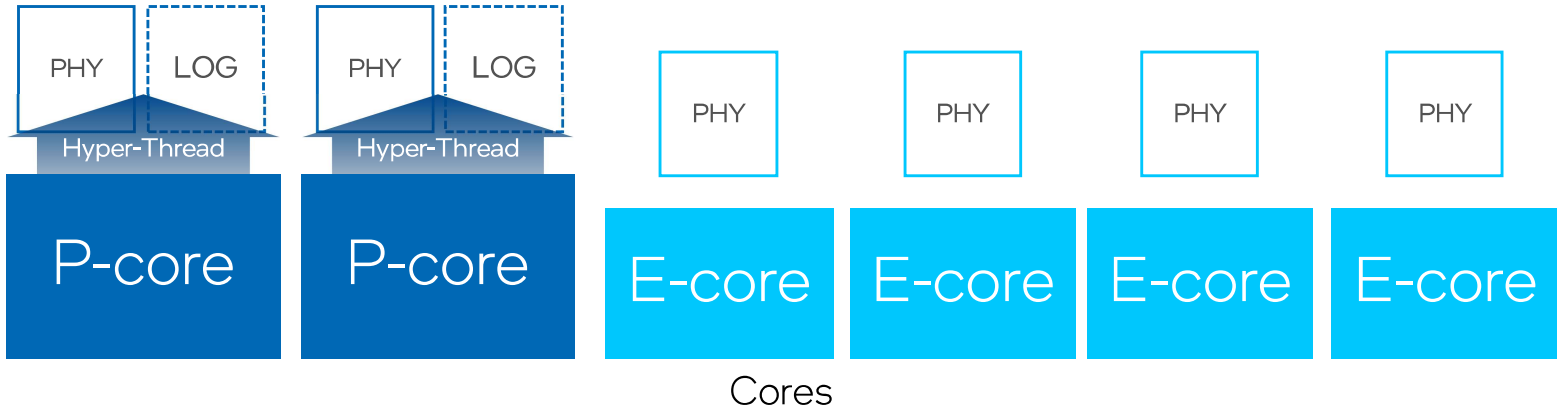
```python
from openvino import runtime as ov

core = ov.Core()

# Implicitly trigger in GPU when in THROUGHPUT mode
compiled_model = core.compile_model(model, "GPU", {"PERFORMANCE_HINT": "THROUGHPUT"})

# Automatic batch size selection
num_req = compiled_model.get_property("OPTIMAL_NUMBER_OF_INFER_REQUESTS")
config = {"PERFORMANCE_HINT": "THROUGHPUT", "PERFORMANCE_HINT_NUM_REQUESTS": num_req}
compiled_model = core.compile_model(model, "GPU", config)

# Limiting batch size according to specific use case
config = {"PERFORMANCE_HINT": "THROUGHPUT", "PERFORMANCE_HINT_NUM_REQUESTS": "4"}
compiled_model = core.compile_model(model, "GPU", config)
```

Default Inference Precision



Thread Scheduling

```
config["SCHEDULING_CORE_TYPE"] = "ANY_CORE" # ANY_CORE, PCORE_ONLY, ECORE_ONLY
config["ENABLE_HYPER_THREADING"] = True
```



Running on E-cores = power saving and P-cores = performance!

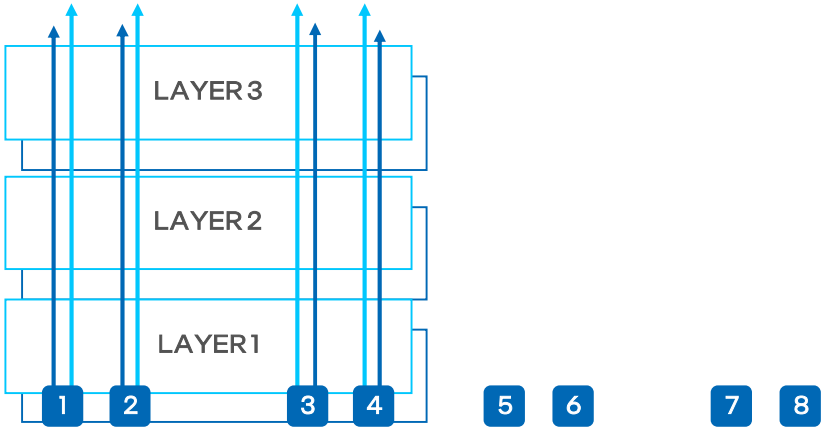
P-core = Performance core, E-core = Efficiency core

Thread Scheduling

Limit resource and pin threads to the CPU processor

```
config["INFERENCE_NUM_THREADS"] = 4
```

(Default configuration)



Threads for running InferRequrest1

Threads for running InferRequrest2

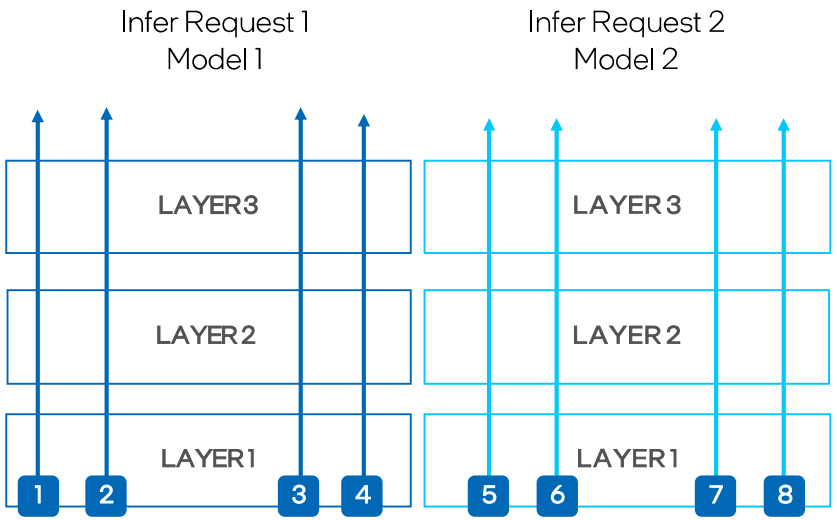
8 CPU processor ID

Thread competing on same resources

Thread Scheduling

OS schedules the threads

```
config["INFERENCE_NUM_THREADS"] = 4
config["ENABLE_CPU_PINNING"] = False #(only on Linux)
```



Threads for running InferRequrest1

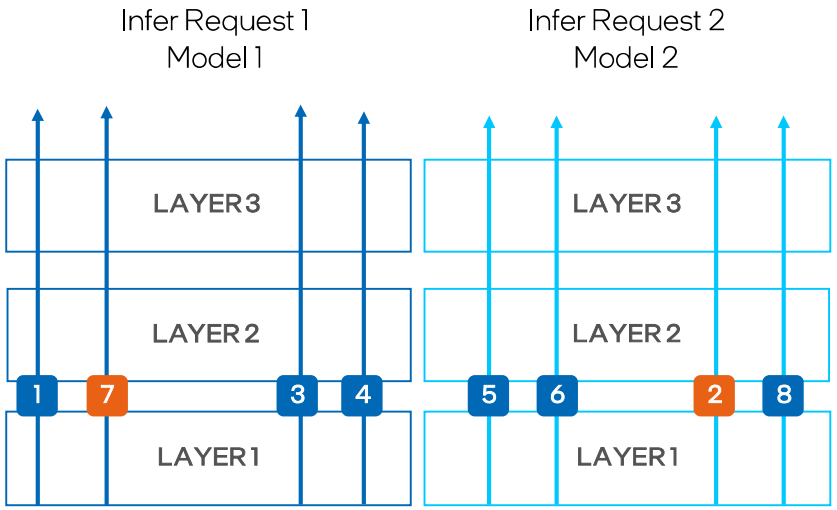
Threads for running InferRequrest2

8 CPU processor ID

Thread Scheduling

OS schedules the threads

```
config["INFERENCE_NUM_THREADS"] = 4
config["ENABLE_CPU_PINNING"] = False #(only on Linux)
```

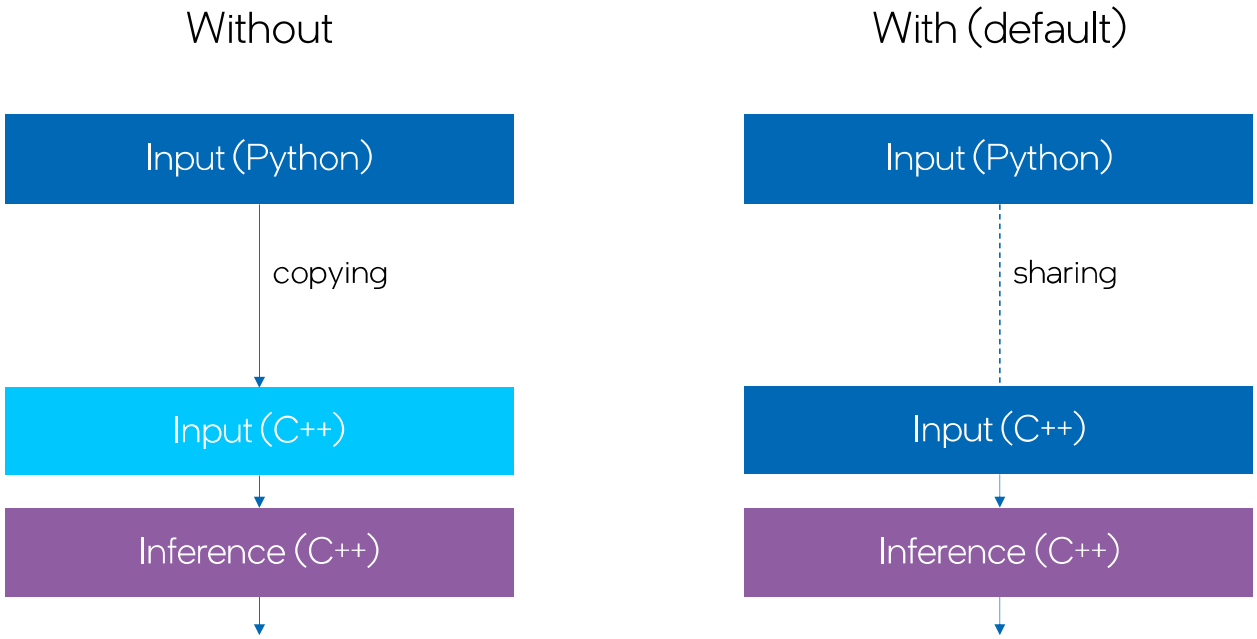


Threads for running InferRequrest1

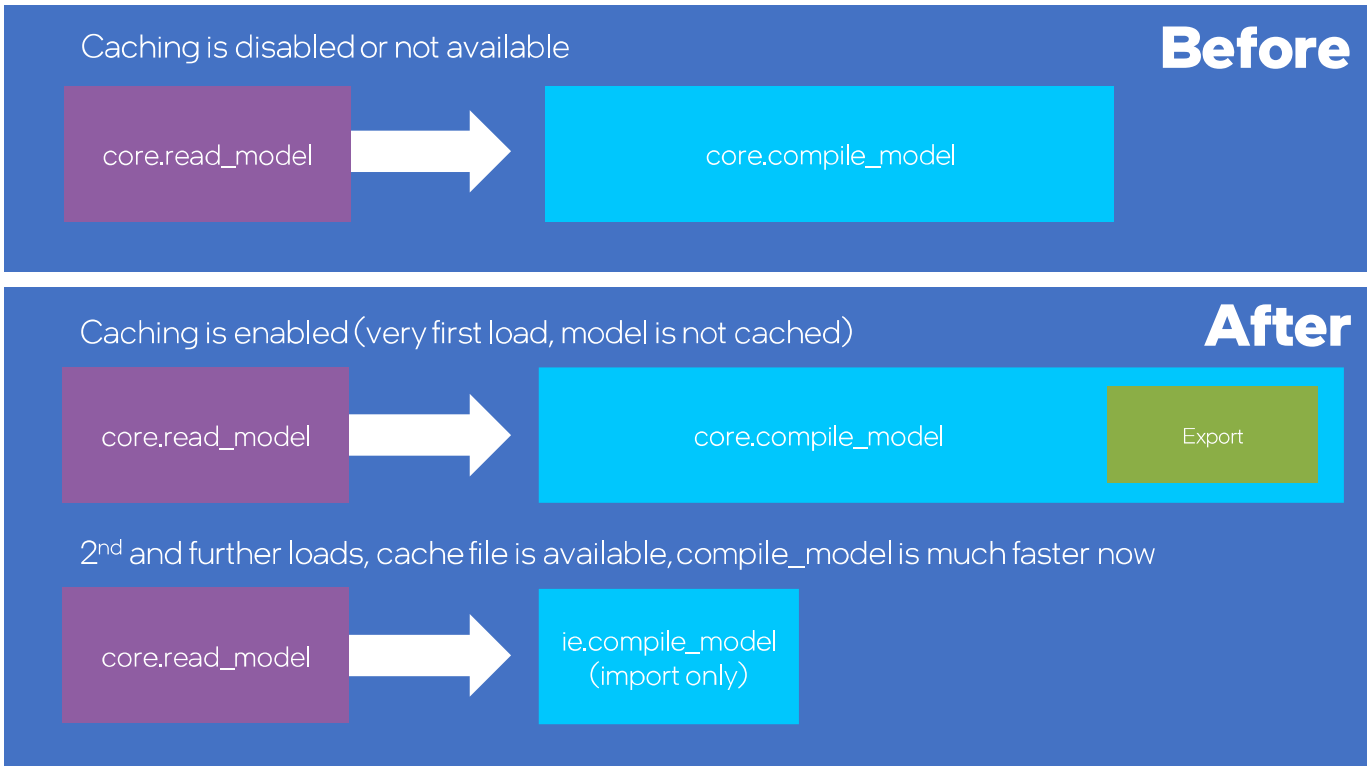
Threads for running InferRequrest2

8 CPU processor ID

Shared Memory



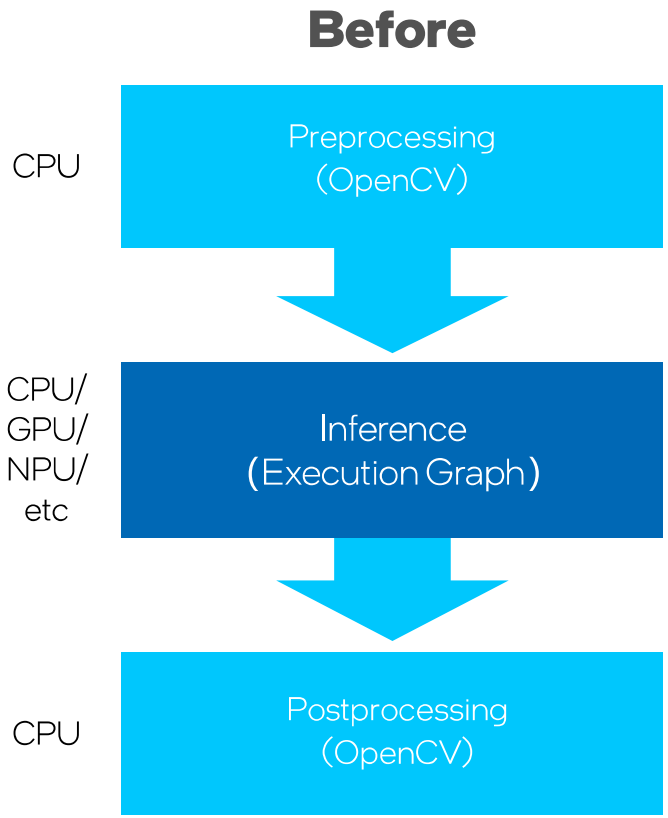
Model Caching



Model Caching

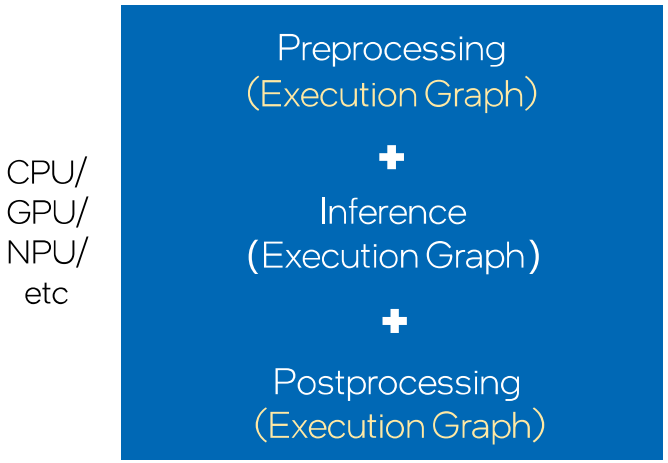
```python
from openvino import runtime as ov

core = ov.Core()
core.set_property({"CACHE_DIR": "/path/to/cache/dir"})

model = core.read_model(model=xml_path)
compiled_model = core.compile_model(model=model, device_name=device_name)
```

Optimize Pre- and Post-Processing



Optimize Pre- and Post-Processing

After



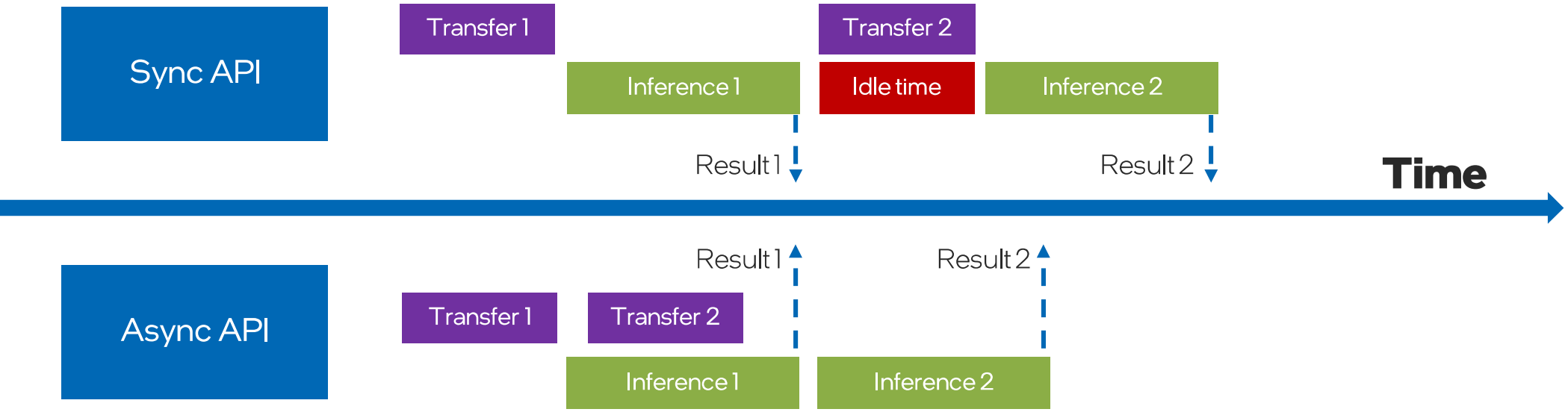
PrePostProcessor (PPP)

```
from openvino import runtime as ov
from openvino.preprocess import PrePostProcessor, ResizeAlgorithm

core = ov.Core()
model = core.read_model("model.xml")
ppp = PrePostProcessor(model)

# setup format of data
ppp.input().tensor().set_element_type(Type.u8)\
                    .set_spatial_dynamic_shape()\
                    .set_layout(Layout('NHWC'))
# add preprocessing
ppp.input().preprocess().convert_element_type(Type.f32)\
                        .resize(ResizeAlgorithm.RESIZE_LINEAR)\
                        .mean([127.5,127.5,127.5])\
                        .scale([127.5,127.5,127.5])
ppp_model = ppp.build()
compiled_model = core.compile_model(model=ppp_model)
```

Pre-Processing Steps


```
from openvino.preprocess import ResizeAlgorithm
ppp.input().preprocess()
            .convert_element_type(Type.f32) \
            .convert_color(ColorFormat.RGB) \
            .resize(ResizeAlgorithm.RESIZE_LINEAR)\
            .mean([127.5,127.5,127.5]) \
            .scale([127.5,127.5,127.5])
```

Asynchronous Mode



Asynchronous Mode

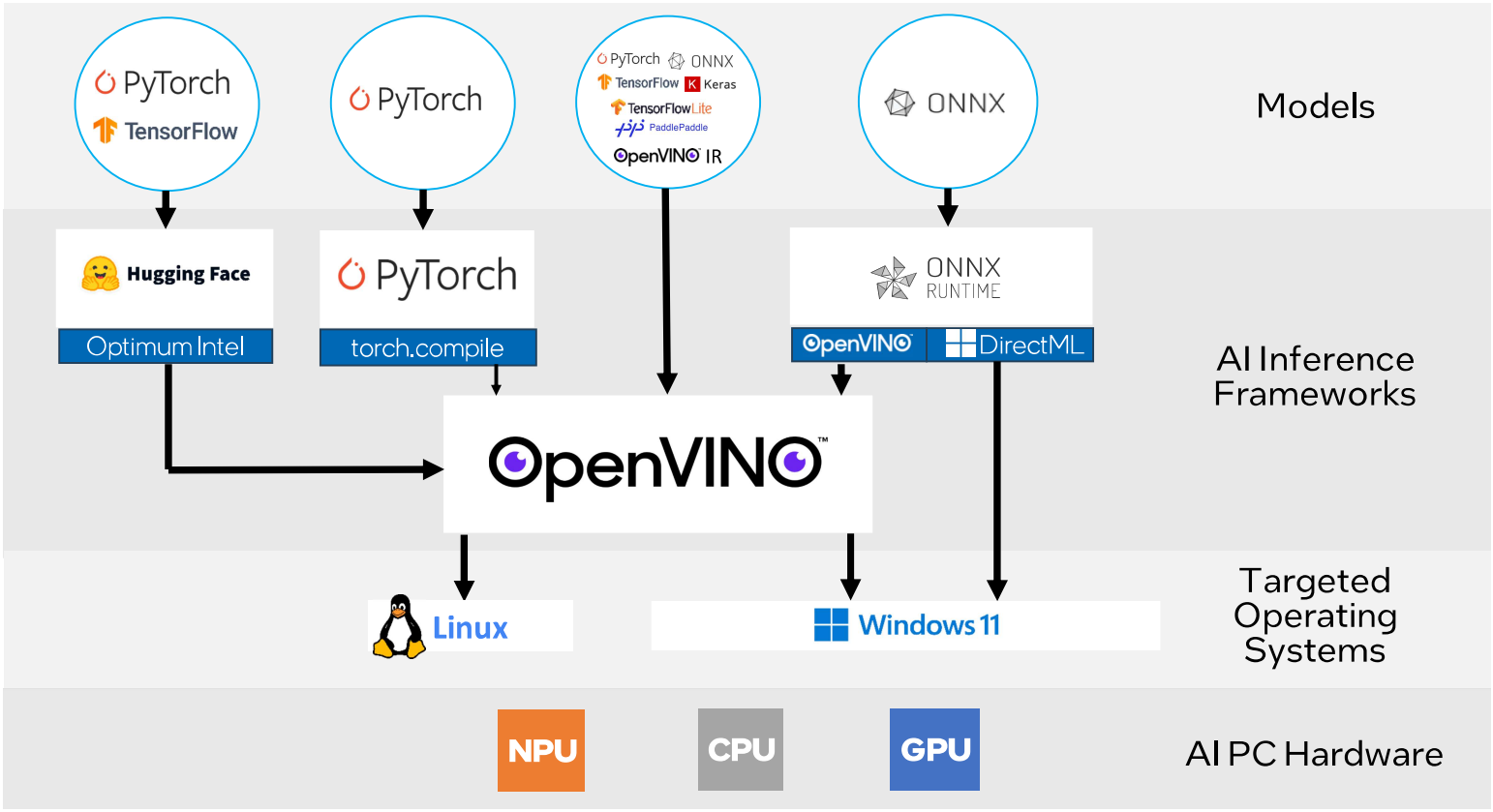
```python
from openvino import runtime as ov

compiled_model = load_and_compile_ov_model()
input_layer = compiled_model.input(0)

def callback(infer_request):
    # do postprocessing
    pass

infer_queue = ov.AsyncInferQueue(compiled_model)
infer_queue.set_callback(callback)

while True:
    frame = cam.read()
    infer_queue.start_async({input_layer.any_name: frame}, (frame, ...))

infer_queue.wait_all()
```

Let's Run the Demo!

OpenVINO

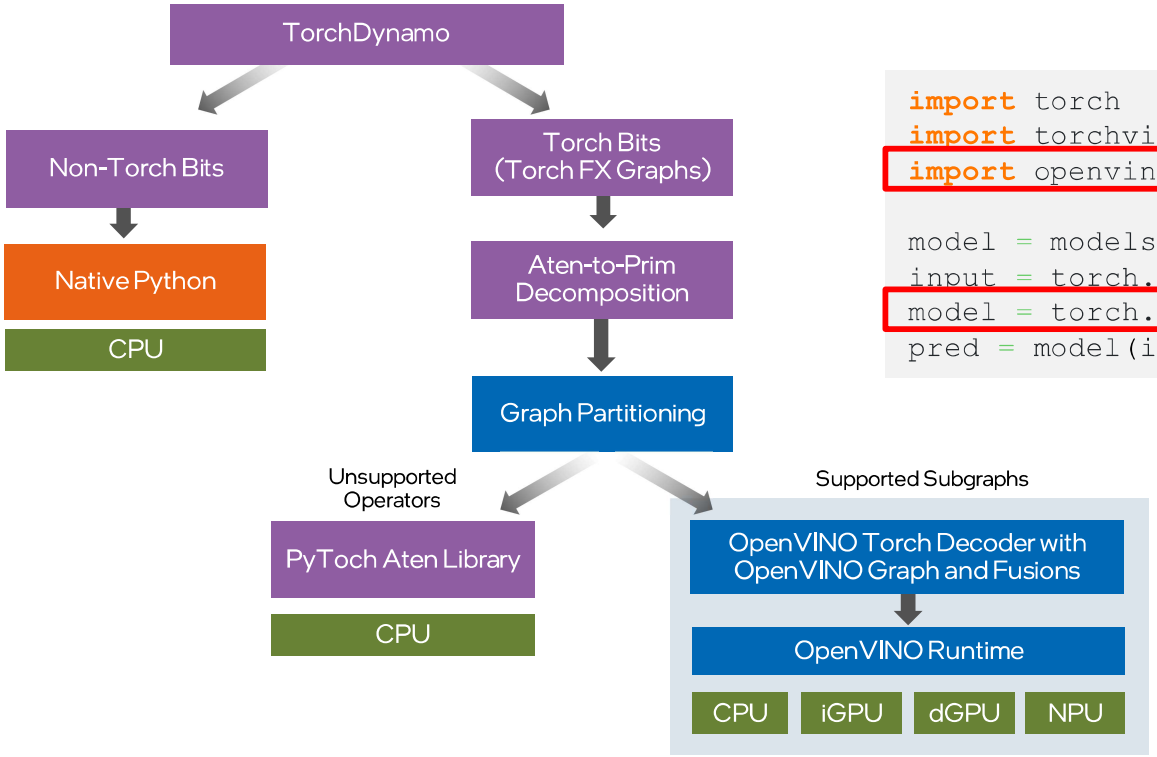
intel

AI Inference SW for AI PC



OpenVINO™ as Backend

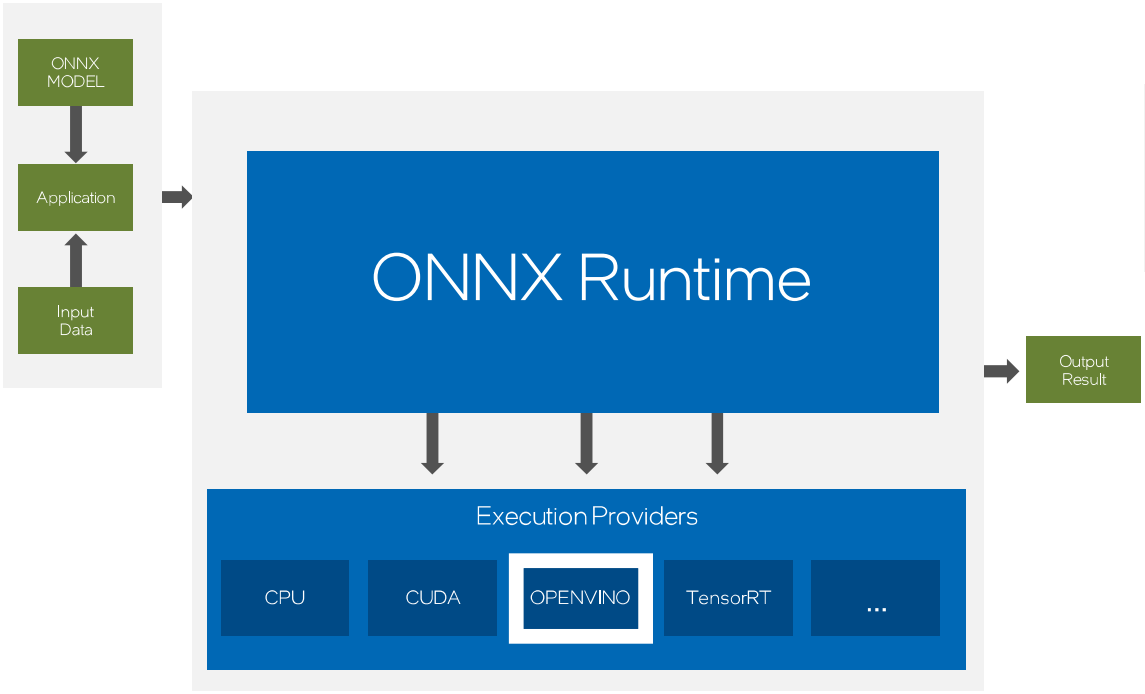
PyTorch 2.0




```python
import torch
import torchvision.models as models
import openvino.torch

model = models.resnet50(pretrained=True)
input = torch.rand((1,3,224,224))
model = torch.compile(model, backend="openvino")
pred = model(input)
```

Only 2 lines of codes

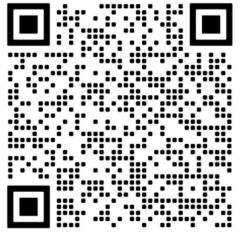
OpenVINO™ Execution Provider for ONNX



Installation

```
Linux
 pip install onnxruntime-openvino
Windows
 pip install onnxruntime-openvino openvino
```

OpenVINO™ Execution Provider for ONNX



```python
import onnxruntime as rt

# Any hardware target of 'CPU_FP32', 'GPU_FP32', 'GPU_FP16', 'MYRIAD_FP16', 'VAD-M_FP16'
device = "CPU_FP32"

# Specify the path to the ONNX model, register the OpenVINO EP and device for inference
sess = rt.InferenceSession("model.onnx", providers=["OpenVINOExecutionProvider"],
provider_options=[{"device_type" : device}])

# Preprocessing the input frame and reshaping it
preprocessed_image = image_preprocess(frame)

# Running the session by passing in the input data of the model
out = sess.run(None, {input_name: preprocessed_image})
```

OpenVINO™ Integration with Optimum

+

OpenVINO™

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

OpenVINO™ Integration with Optimum

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

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

  model_id = "helenai/gpt2-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 = "helenai/gpt2-ov„
model = OVModelForCausalLM.from_pretrained(model_id)
```

```python
from optimum.intel import OVModelForCausalLM

model_id = "gpt2"
model = OVModelForCausalLM.from_pretrained(model_id, export=True)

model.save_pretrained("ov_model")
```

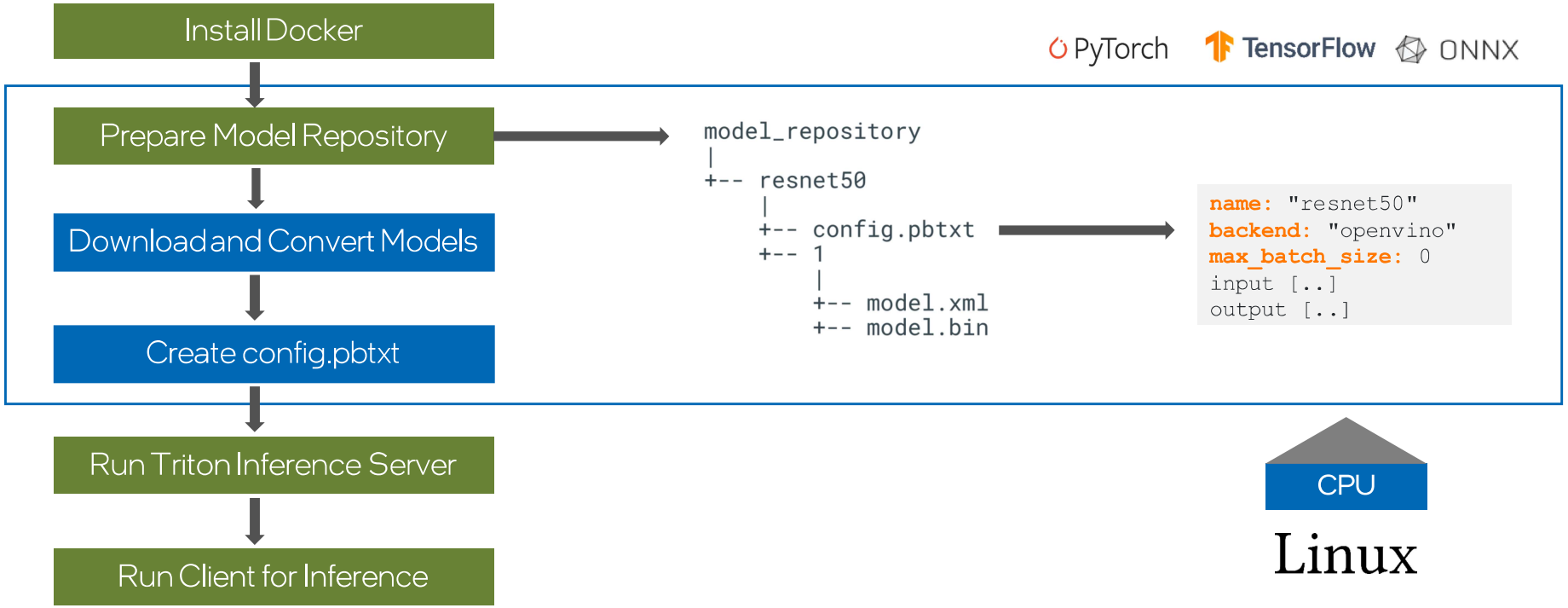
Let's Run the Demo!

OpenVINO

intel

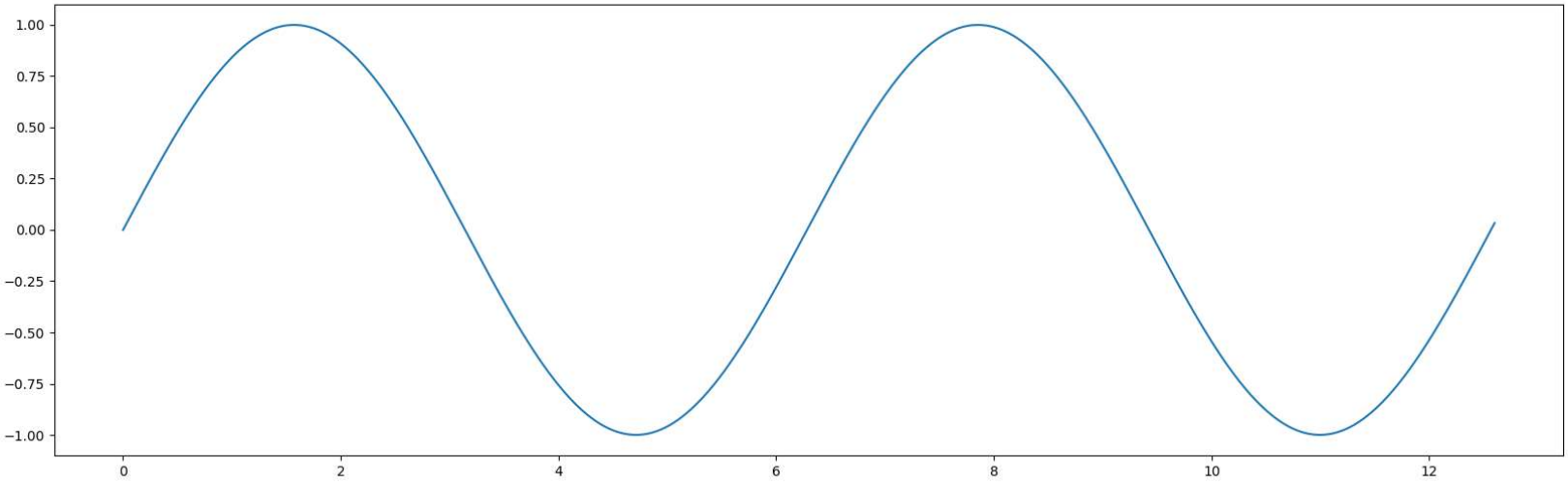
OpenVINO™ as Backend

Triton Inference Server

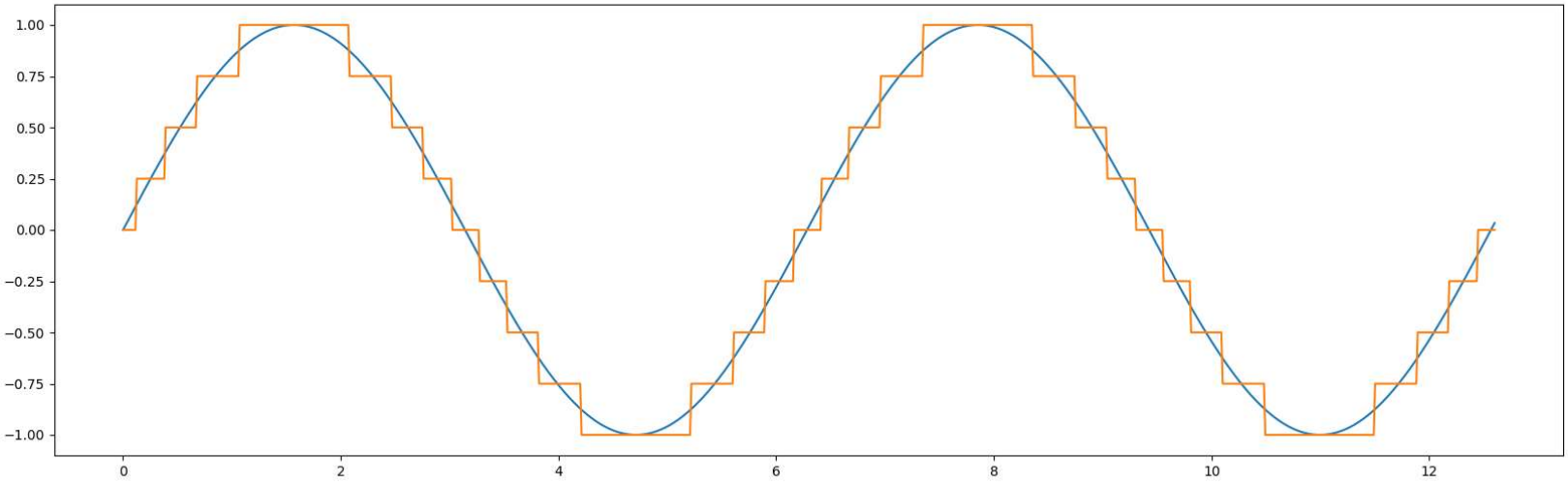



```
root@iceflex:/workspace/ ov # python3 client.py
['11.548589:92' '11.231404:14' '7.527273:95' '6.922710:17' '6.576274:88']
Inference time: 0.0222 seconds
root@iceflex:/workspace/ ov #
```

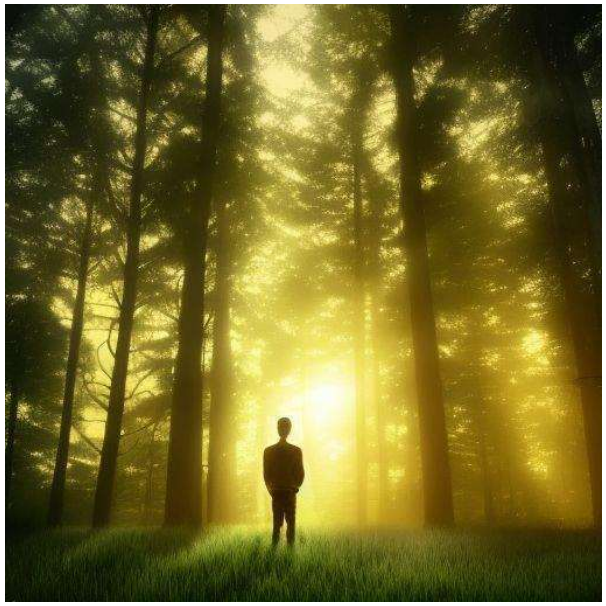
Quantization



Quantization



Color Quantization

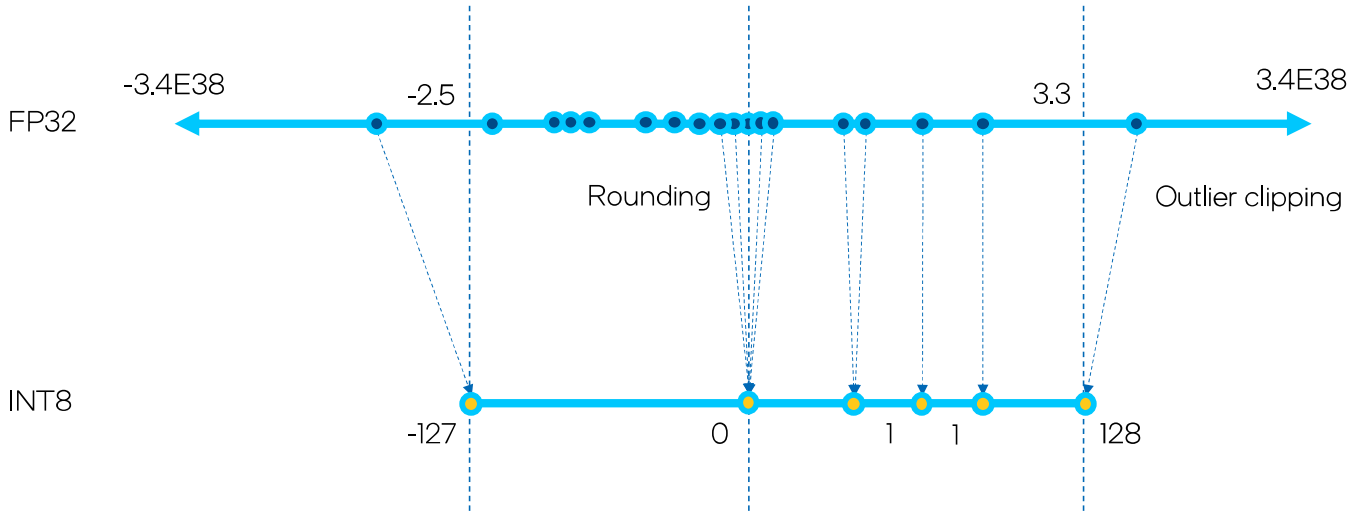


FP32 ⇒ 16,777,216 colors

INT8 ⇒ 256 colors

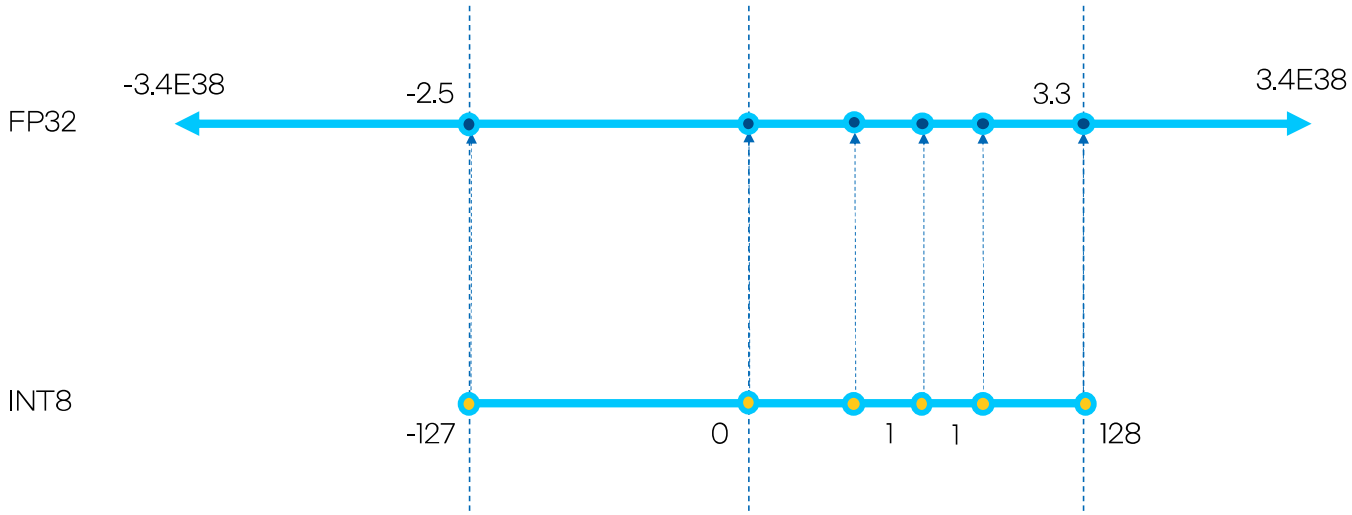
INT4 ⇒ 16 colors

Quantization



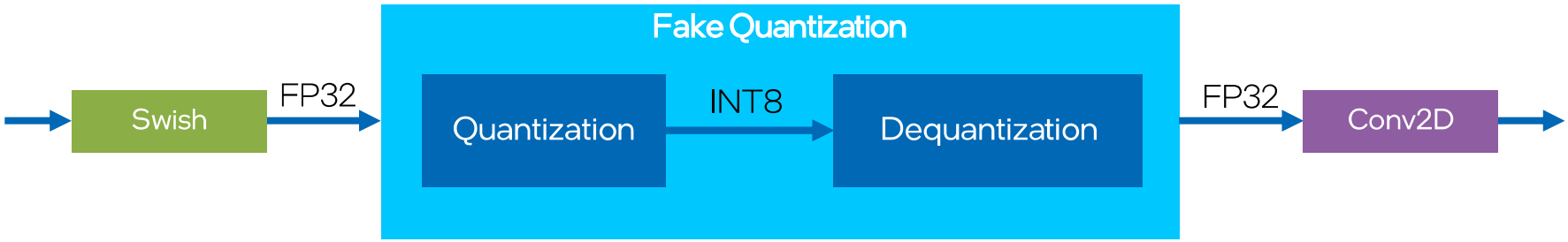

```
int8_value = round(real_value / scale) + zero_point
```

Dequantization

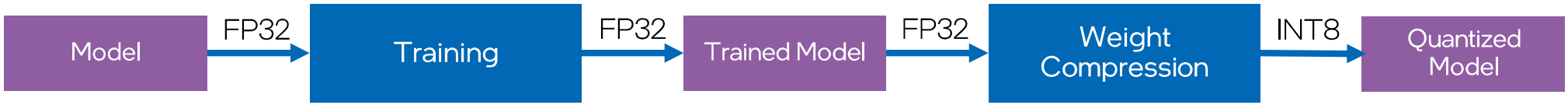



```
real_value = (int8_value - zero_point) * scale
```

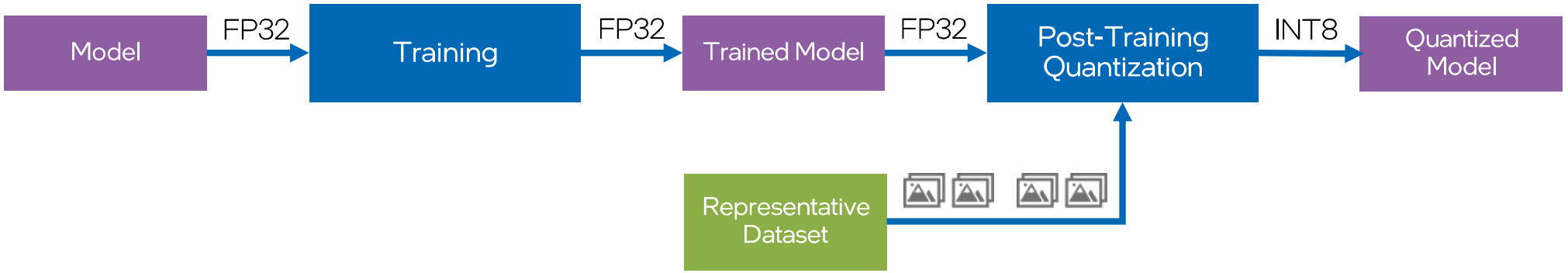
Fake Quantization



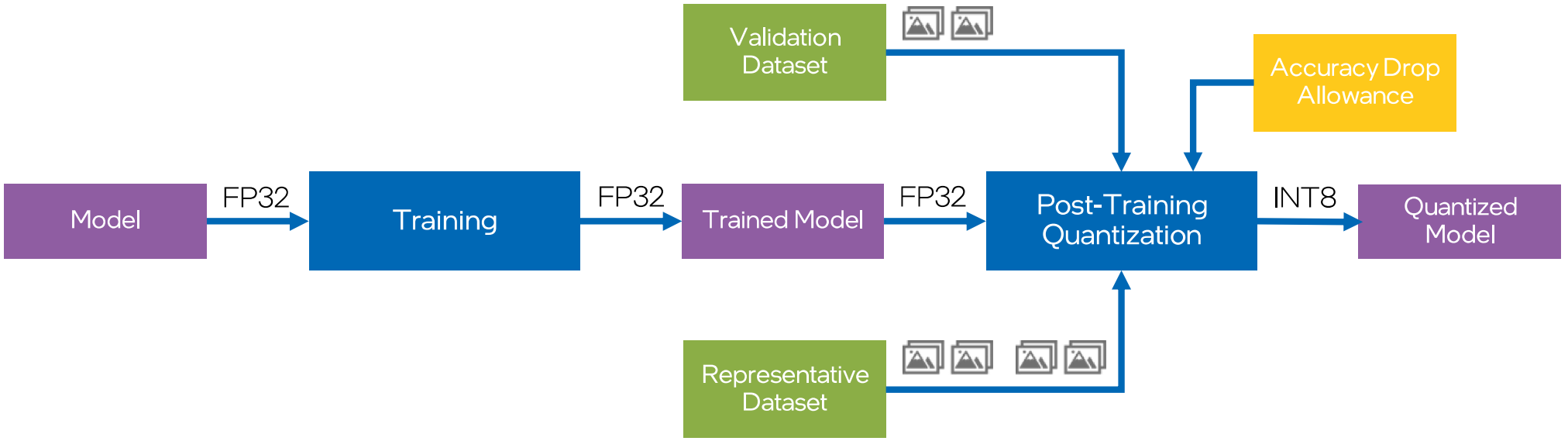
Weight Compression



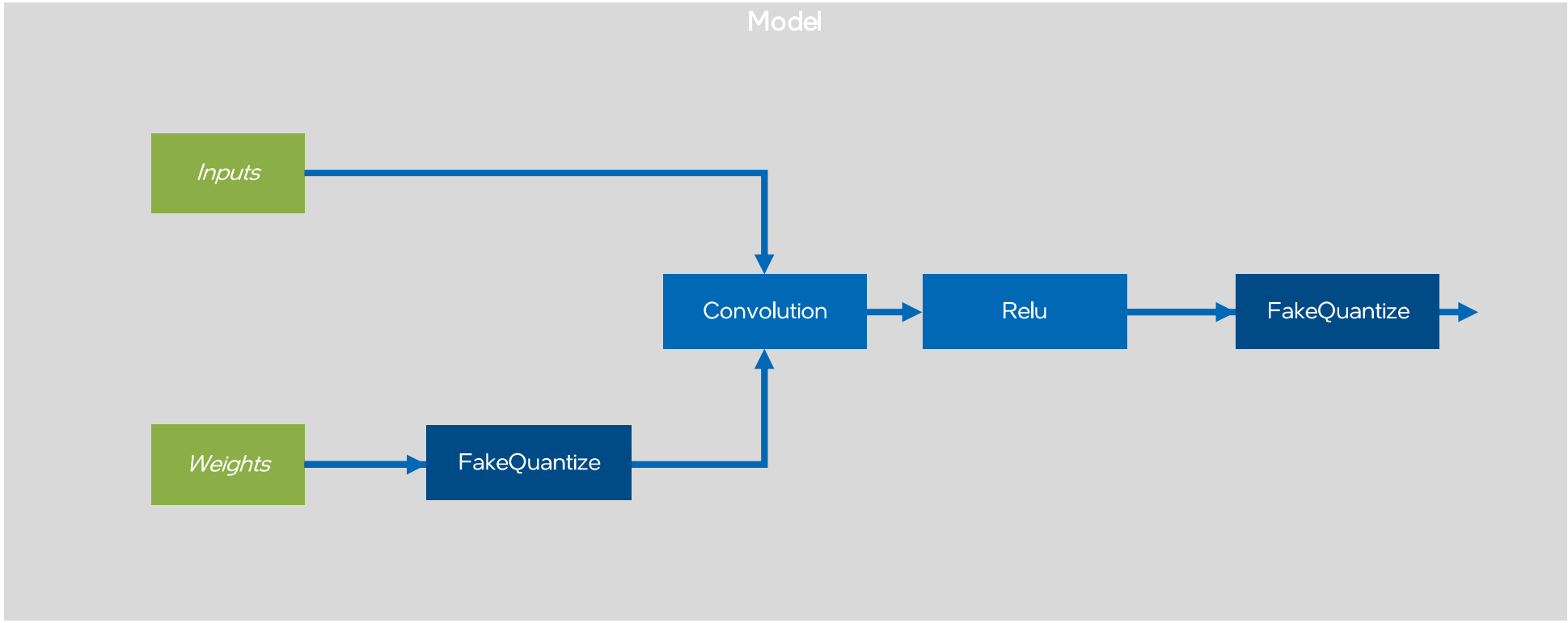
Post-Training Quantization (PTQ)



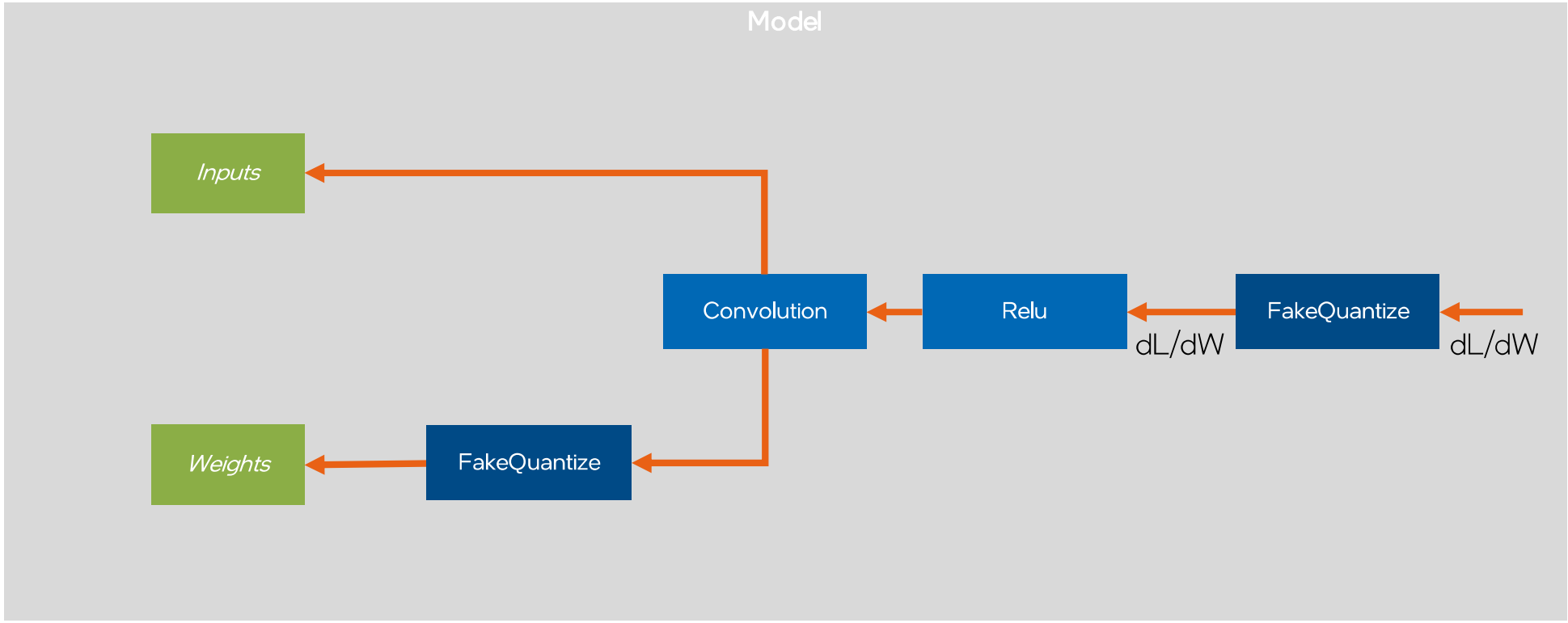
Accuracy-Control Quantization



Quantization-Aware Training



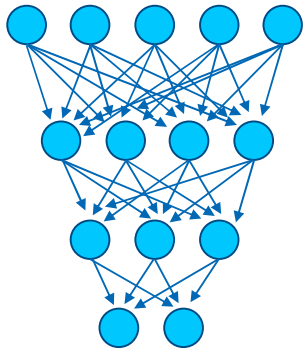
Quantization-Aware Training



Neural Network Compression Framework (NNCF)

TensorFlow PyTorch

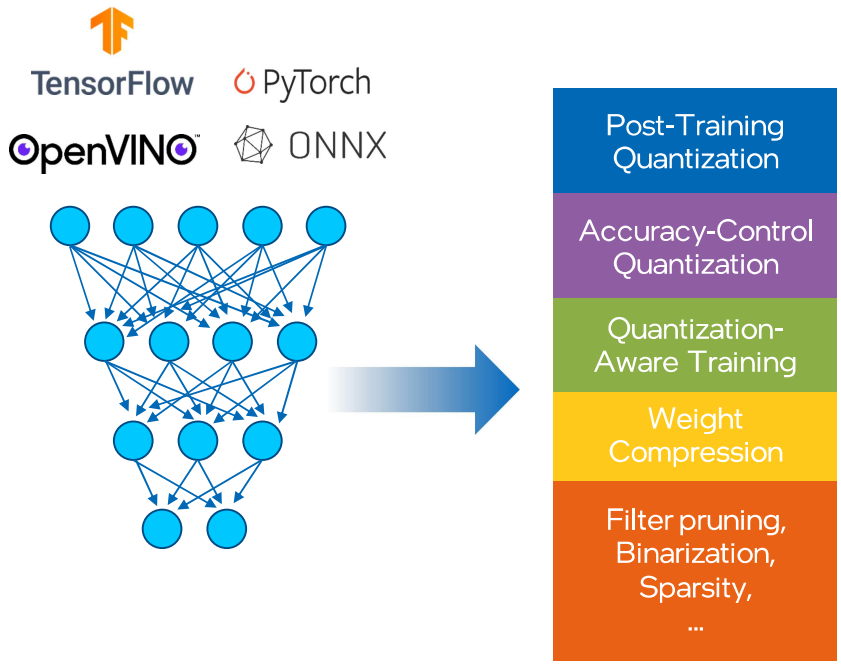
OpenVINO ONNX



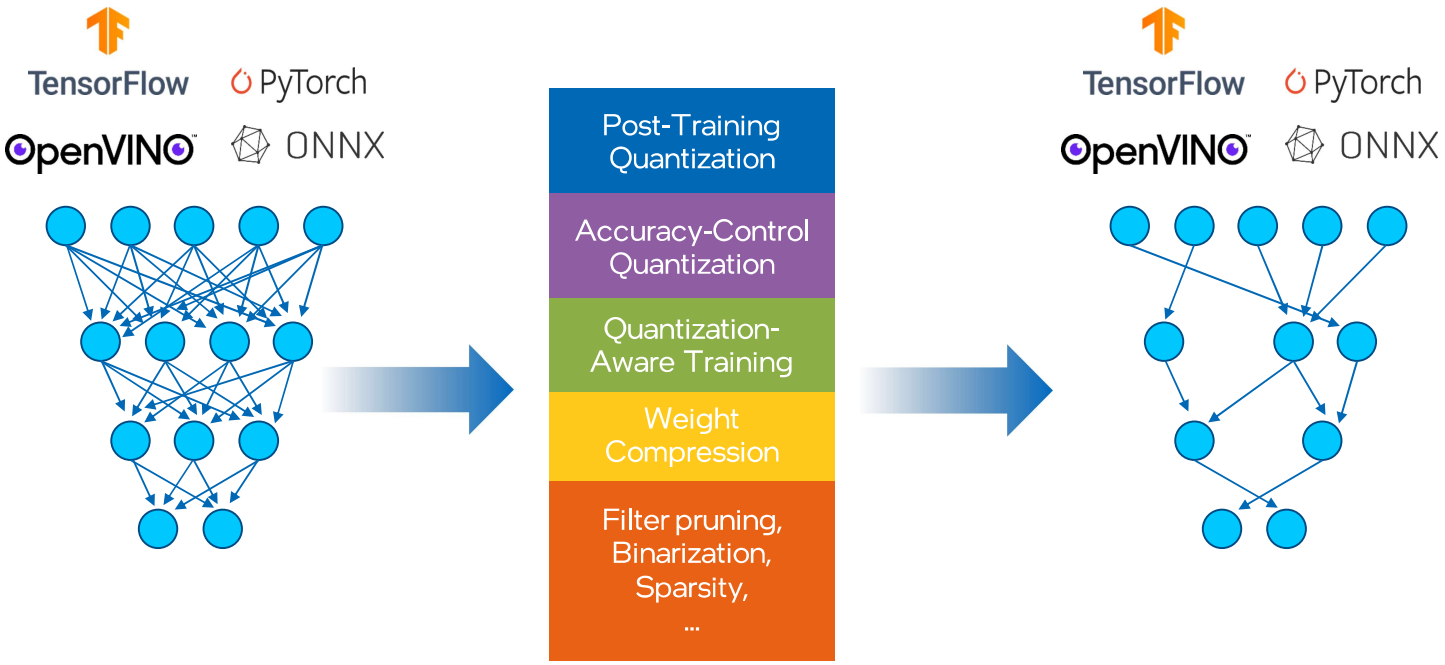
OpenVINO

intel

Neural Network Compression Framework (NNCF)



Neural Network Compression Framework (NNCF)



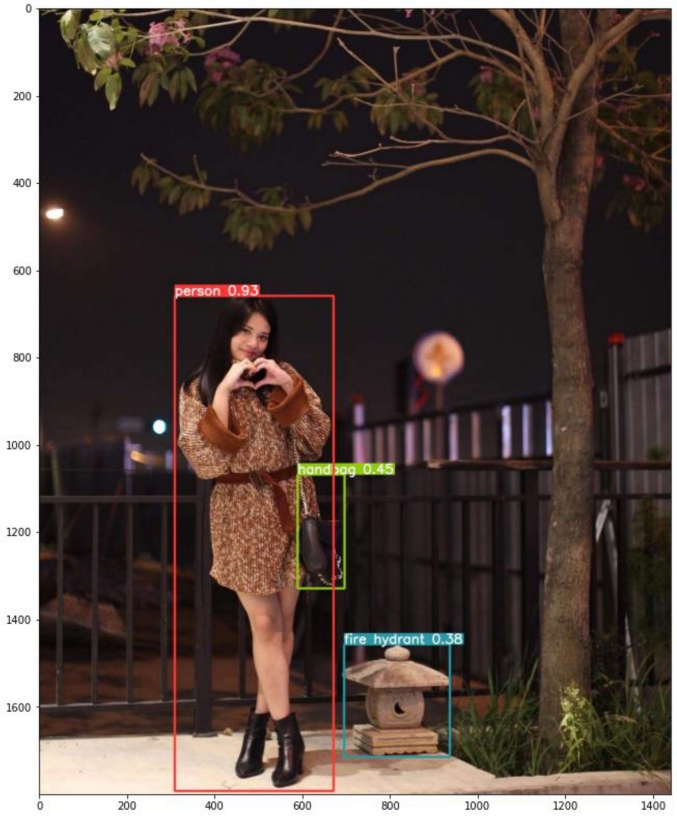

```
pip install nncf
```

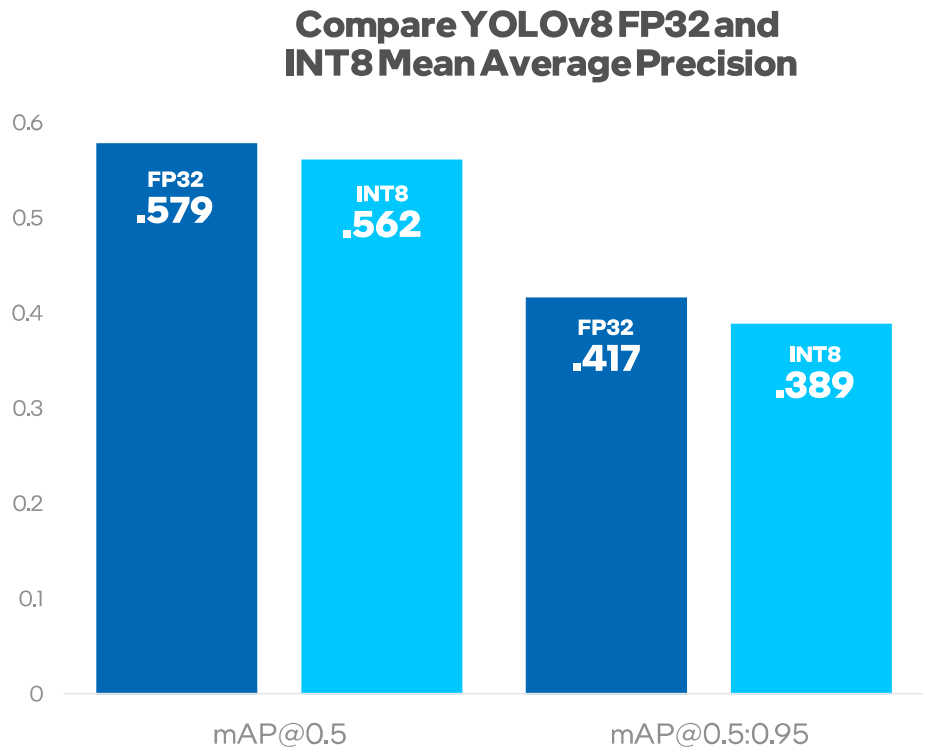
Weight Compression for LLMs (NNCF)

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

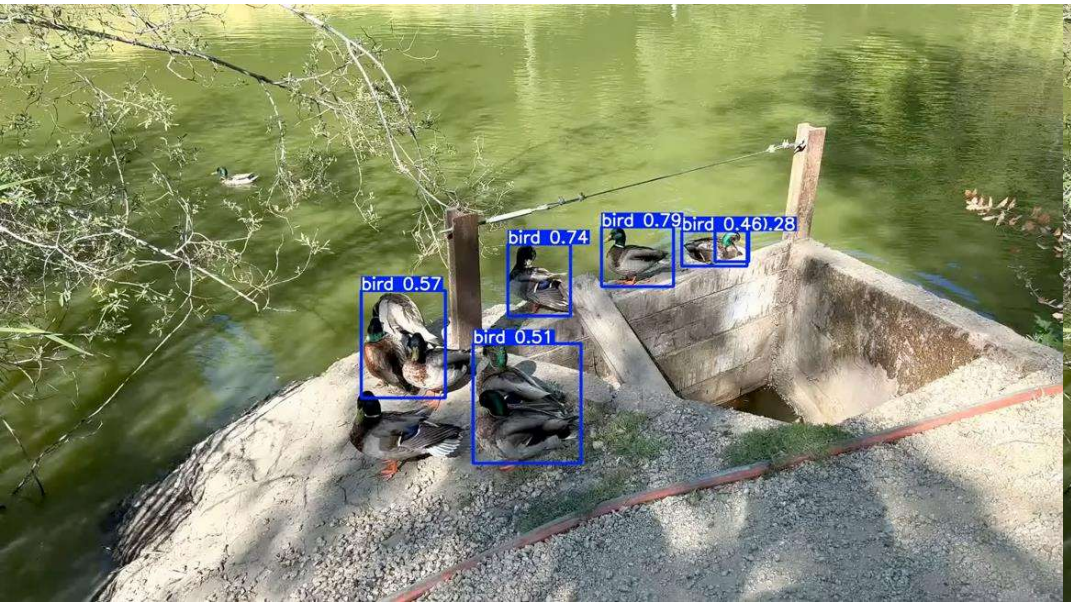
Significant Reduction in RAM usage!

Post-Training Quantization (NNCF)

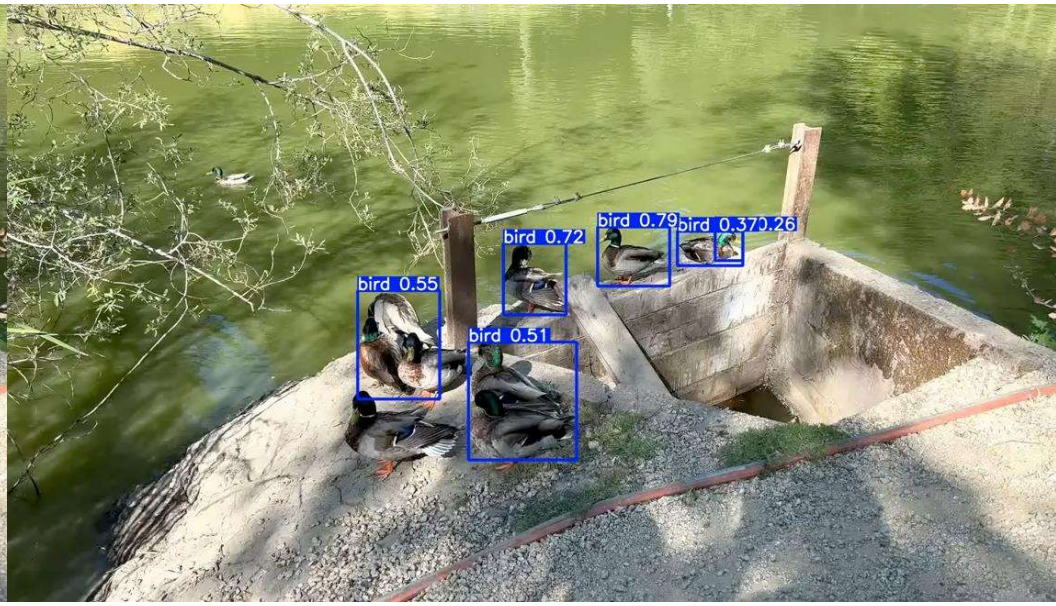




Quantization Results Comparison (YOLOv8)



FP32



INT8

Hybrid Post-Training Quantization (NNCF)

- U-Net: quantization applied on both the weights and activations
- The text encoder, VAE encoder / decoder: quantization applied on the weights

"a photo of an astronaut riding a horse on mars"

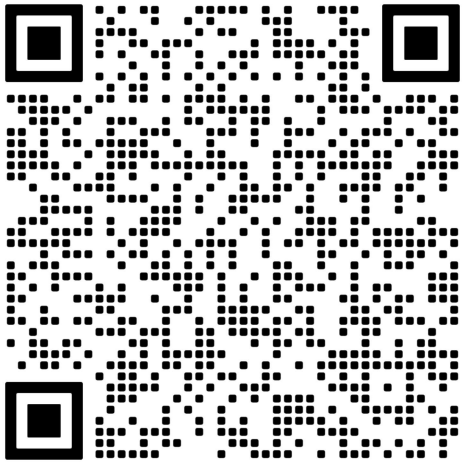
FP32 | INT8 PTQ | INT8 Hybrid PTQ

Quantization Performance Comparison (OpenPose)

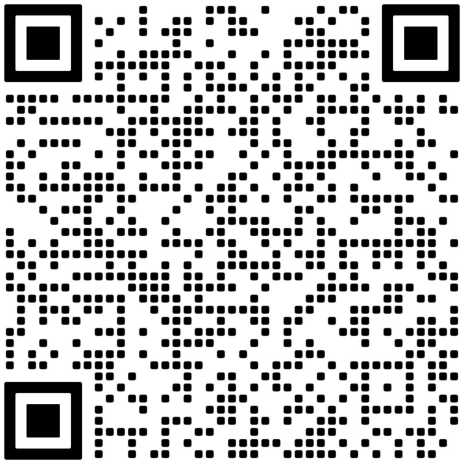
16.6 MB

4.7 MB

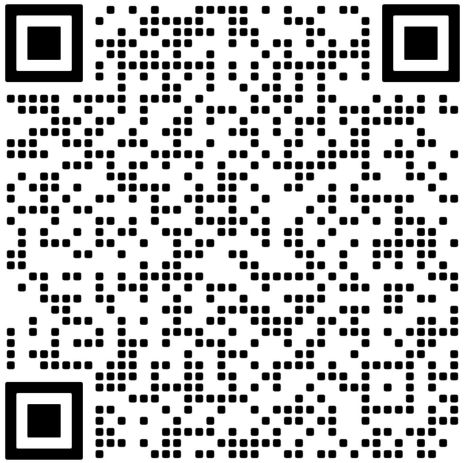
Quantization with NNCF



Post-Training Quantization



Accuracy-Control Quantization



Quantization-Aware Training

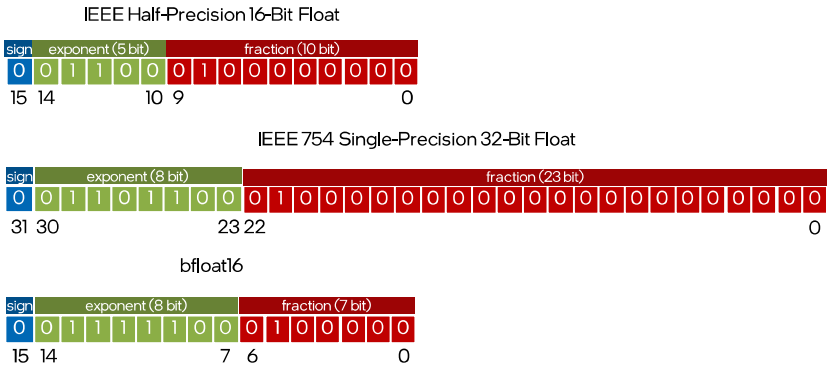
Quantization with NNCF

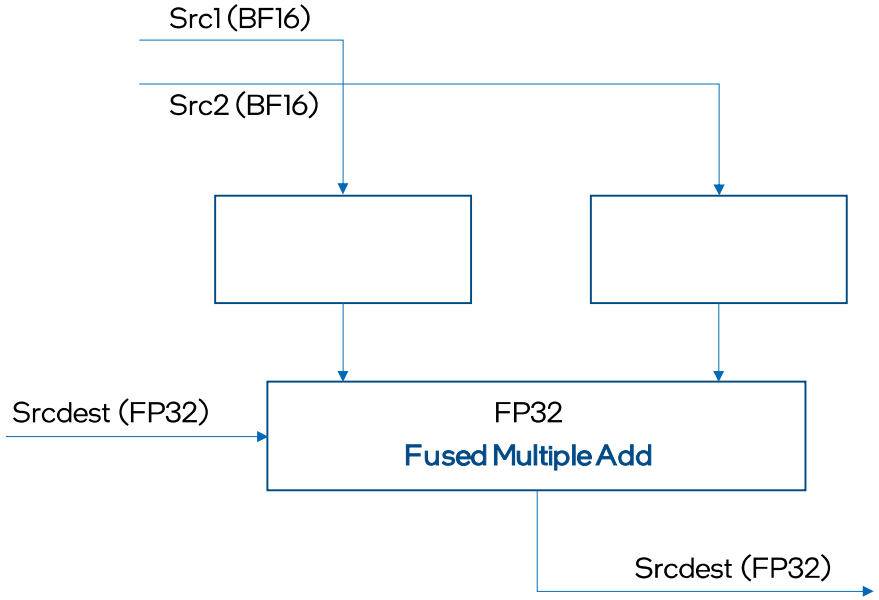
More on this topic in Module 3
(starts 2 pm)


Hardware Optimizations
Bfloat16 (Xeon) | VNNI (Xeon) | AMX (Xeon) | XMX (Arc, Flex)
OpenVINO
intel


How to Get Best
Performance on
Xeon Server?
OpenVINO

BFloat16





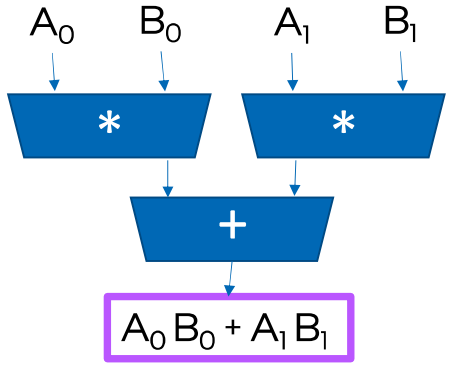
Increase calculation speed

Overview of Instruction Set Used

VPMADDUBSW ❶

Vector multiply into 16-bit + horizontal add of neighbors
Executes on Port 0 in Cycle 0

SRC1 8-bit	A_0	A_1	A_2	A_3	. . . .	A_{63}
SRC2 8-bit	B_0	B_1	B_2	B_3	. . . .	B_{63}
DEST 16-bit	$A_0*B_0 + A_1*B_1$		$A_2*B_2 + A_3*B_3$		. . . .	$A_{62}*B_{62} + A_{63}*B_{63}$



Overview of Instruction Set Used

VPMADDUBSW

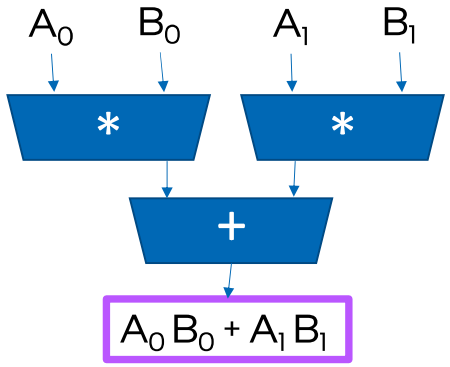
Vector multiply into 16-bit + horizontal add of neighbors
Executes on Port 0 in Cycle 0

1 → 2

VPMADDWD

Effectively upconvert to 32-bit and horizontal add of neighbors
Executes on Port 5 in Cycle 0 (uses the multiplier to multiply by 1)

SRC1 16-bit	$A_0*B_0 + A_1*B_1$	$A_2*B_2 + A_3*B_3$	. . . .	$A_{62}*B_{62} + A_{63}*B_{63}$
SRC2 16-bit	1	1	. . . .	1
DEST 32-bit	$A_0*B_0 + A_1*B_1 + A_2*B_2 + A_3*B_3$	. . . .	$A_{60}*B_{60} + A_{61}*B_{61} + A_{62}*B_{62} + A_{63}*B_{63}$	



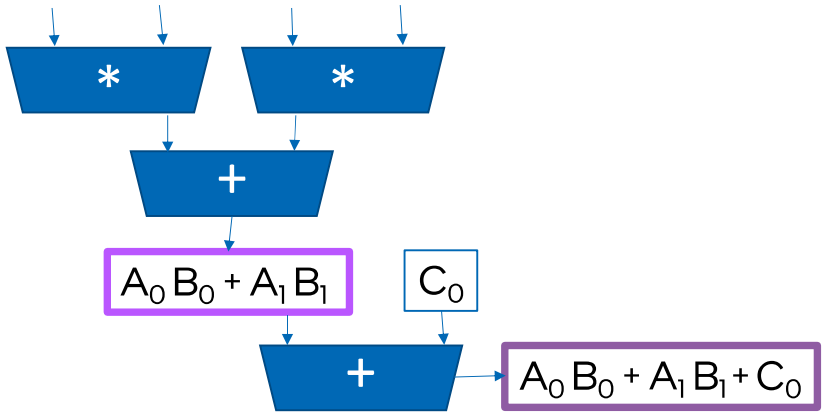
Overview of Instruction Set Used

VPMADDUBSW
Vector multiply into 16-bit + horizontal add of neighbors
Executes on Port 0 in Cycle 0

VPMADDWD
Effectively upconvert to 32-bit and horizontal add of neighbors
Executes on Port 5 in Cycle 0 (uses the multiplier to multiply by 1)

VPADDD
Executes on Port 0 in Cycle 1
(does not use the multiplier)

SRC 1 32-bit	$A_0*B_0 + A_1*B_1 + A_2*B_2 + A_3*B_3$	. . .	$A_{60}*B_{60} + A_{61}*B_{61} + A_{62}*B_{62} + A_{63}*B_{63}$
SRC 2 32-bit	C_0	. . . .	C_{15}
Dest 32-bit	$A_0*B_0 + A_1*B_1 + A_2*B_2 + A_3*B_3 + C_0$	. . . .	$A_{60}*B_{60} + A_{61}*B_{61} + A_{62}*B_{62} + A_{63}*B_{63} + C_{15}$



VNNI - VPDPBUSD

VPMADDUBSW
Vector multiply into 16-bit + horizontal add of neighbors
Executes on Port 0 in Cycle 0

VPMADDWD
Effectively upconvert to 32-bit and horizontal add of neighbors
Executes on Port 5 in Cycle 0 (uses the multiplier to multiply by 1)

VPADDD
Executes on Port 0 in Cycle 1
(does not use the multiplier)

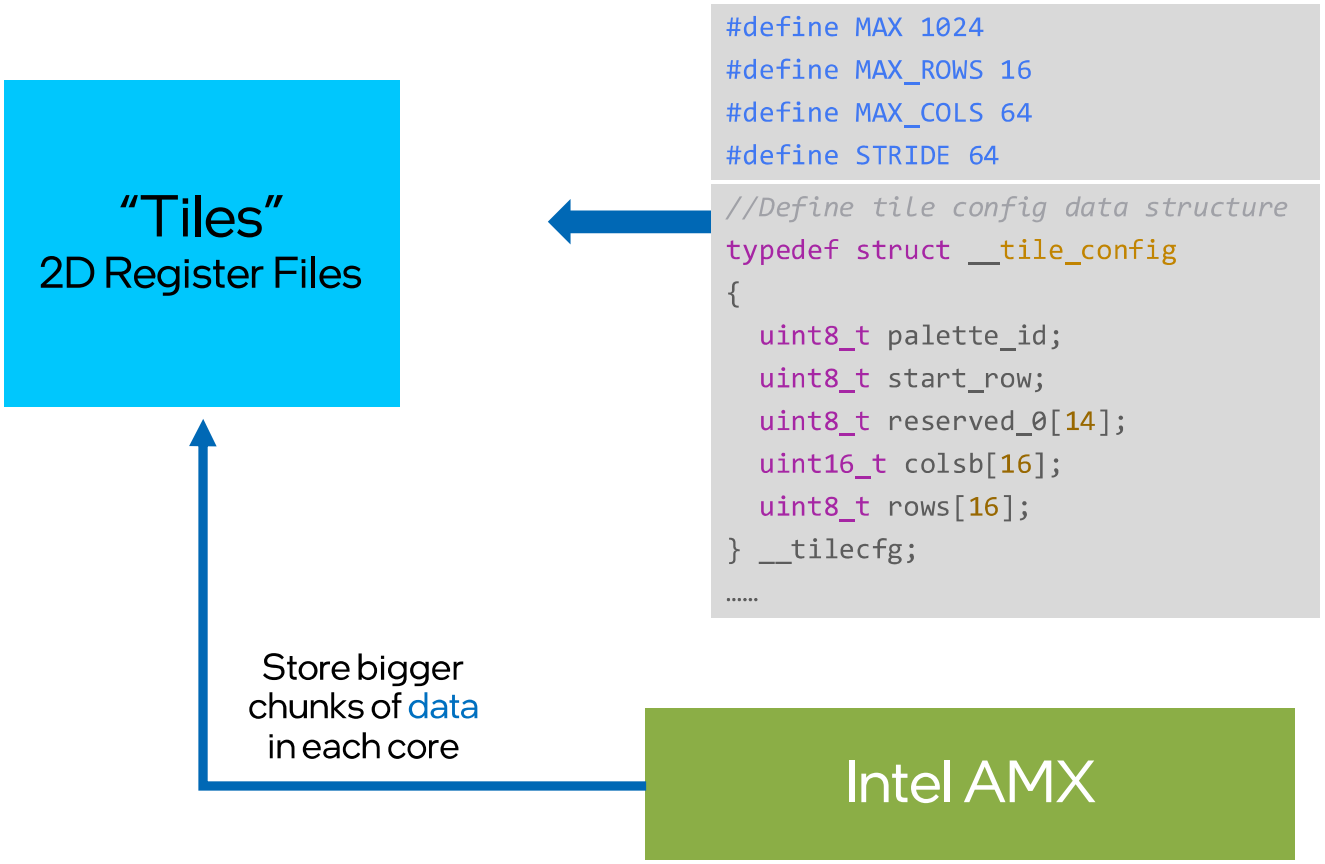
VPDPBUSD
executes on both Port 0 and Port 5 in 1 cycle

(3X PEAK OPS)

SRC 1 8-bit	A_0	A_1	A_2	A_3	....	A_{63}
SRC 2 8-bit	B_0	B_1	B_2	B_3	....	B_{63}
SRC3 / DEST 32-bit	C_0				....	C_{15}
	$A_0*B_0+A_1*B_1+A_2*B_2+A_3*B_3+C_0$				....	$A_{60}*B_{60}+A_{61}*B_{61}+A_{62}*B_{62}+A_{63}*B_{63}+C_{15}$

VPMADDUBSW + VPMADDWD + VPADDD fused into VPDPBUSD

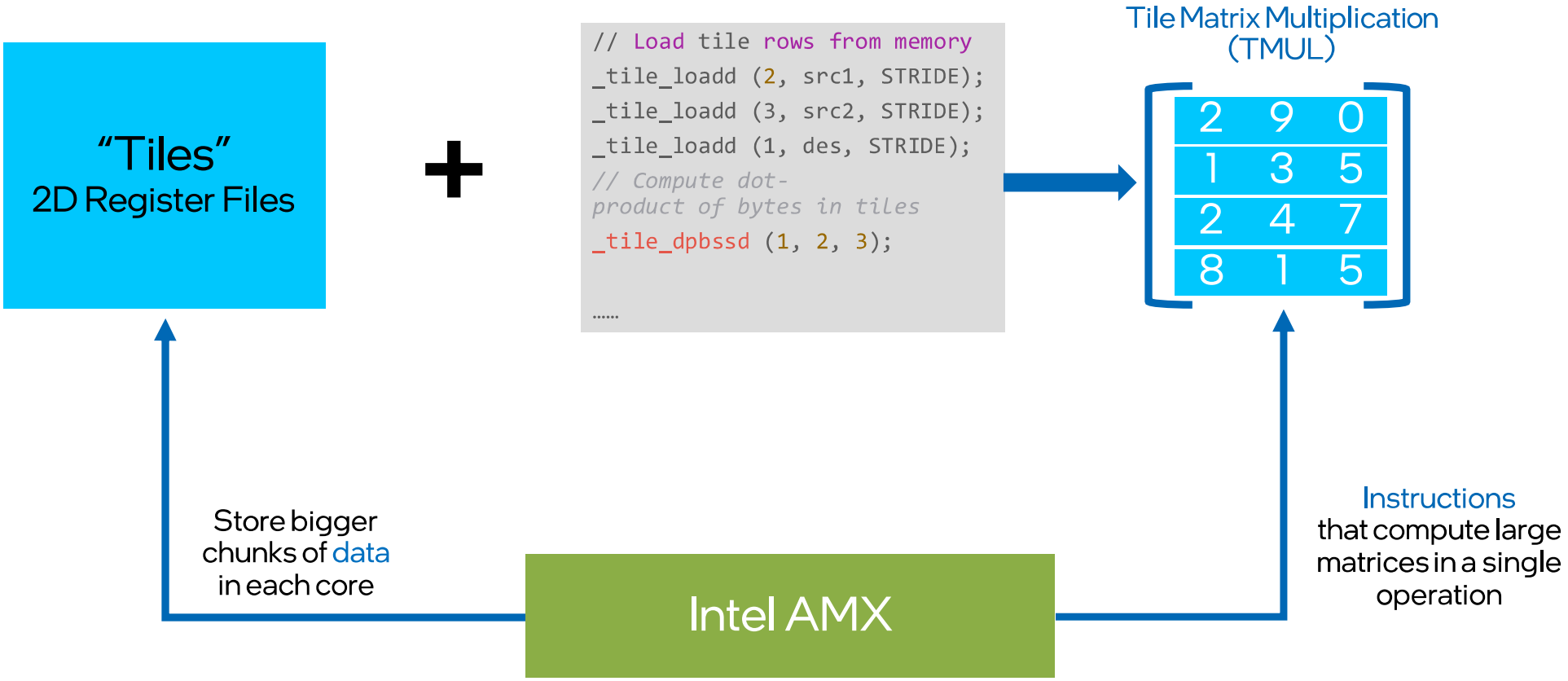
Intel® AMX (Advanced Matrix Extensions)

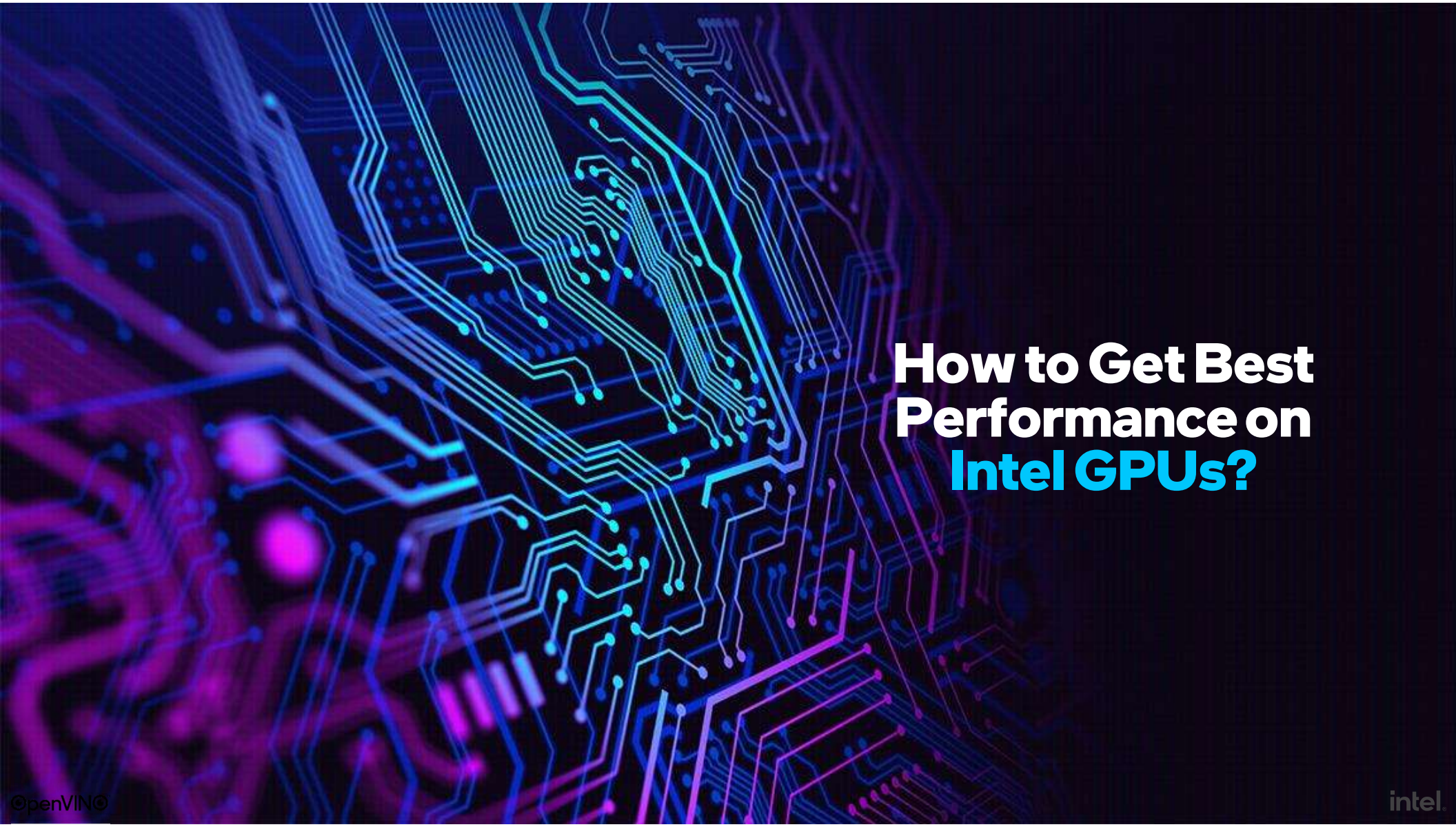



```
#define MAX 1024
#define MAX_ROWS 16
#define MAX_COLS 64
#define STRIDE 64
```

```
//Define tile config data structure
typedef struct __tile_config
{
  uint8_t palette_id;
  uint8_t start_row;
  uint8_t reserved_0[14];
  uint16_t colsb[16];
  uint8_t rows[16];
} __tilecfg;
……
```

Intel® AMX (Advanced Matrix Extensions)

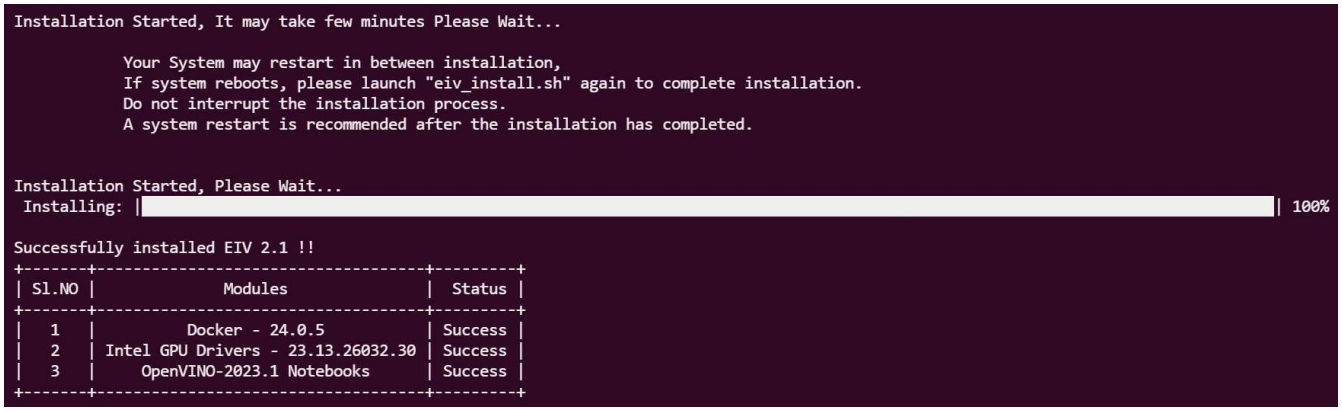



How to Get Best
Performance on
Intel GPUs?
OpenVINO
intel.

Easy Installation of GPU Driver

Edge Insights Vision (EIV)

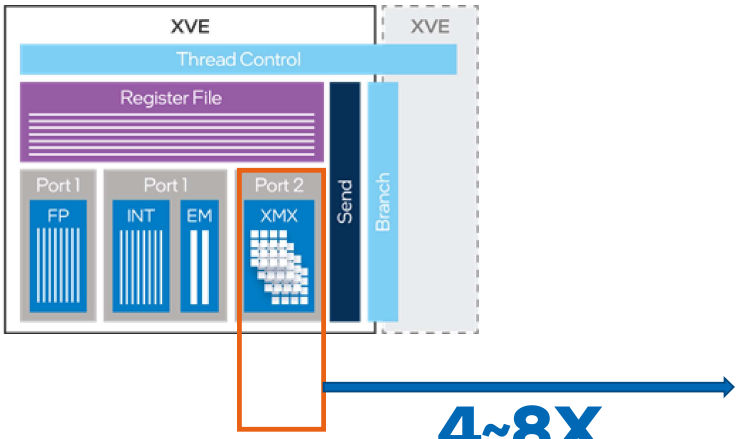
```
# Step 1: Clone the Repository
$ git clone https://github.com/intel/edge-insights-vision.git
$ cd edge-insights-vision
$ ./eiv_install.sh

# Step 2: Restart your system after installation is complete.
```



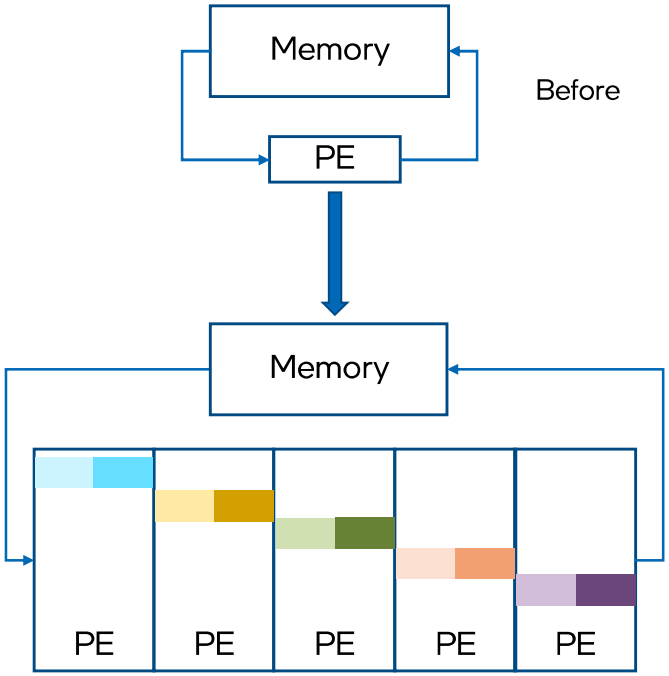

```
# Step 3: Verify driver version
$ clinfo | grep 'Driver Version'
```

Intel® XMX (Xe Matrix Extension)

Xe-HPG Graphics Architecture,
Next Gen Architecture of dGPU



4~8X
Boost



Systolic array (pipeline)

PE: Processing Element

How to Check Whether You Have XMX?

```
$ ./hello_query_device
[ INFO ] GPU.0
[ INFO ]        SUPPORTED_PROPERTIES:
[ INFO ]                Immutable: OPTIMIZATION_CAPABILITIES : FP32 BIN FP16 INT8
# XMX is not supported
[ INFO ] GPU.1
[ INFO ]        SUPPORTED_PROPERTIES:
[ INFO ]                Immutable: OPTIMIZATION_CAPABILITIES : FP32 BIN FP16 INT8 GPU_HW_MATMUL

# XMX is supported
```

Software + Hardware

Software + Hardware
Optimization Makes Up The Other Half!

AI Deployment

Edge	AI PC	Cloud
Industry-specific AI use cases, Real-time data processing, Wider reach	Wide range of consumer specific AI use-cases	Centralization, Edge device or PC connects to the cloud to perform computation and get results back
Pros Data sovereignty Cost efficiency Increased control Autonomous execution	**Pros** Data sovereignty Cost efficiency Special computing capabilities for optimal performance and energy consumption Increased control	**Pros** Large amount of data, limitless compute on demand
Cons Compute is limited by local resources	**Cons** Compute is limited by local resources	**Cons** Risk of data privacy High latency Dependency on the connection to the cloud


What is AI PC?
OpenVINO
intel.

Three AI Engines in Intel® Core™ Ultra

The right balance of power and performance for building and deploying AI models **with OpenVINO™**

NPU

Power Efficiency

Ideal for sustained AI workloads and AI offload for battery life

CPU

Fast Response

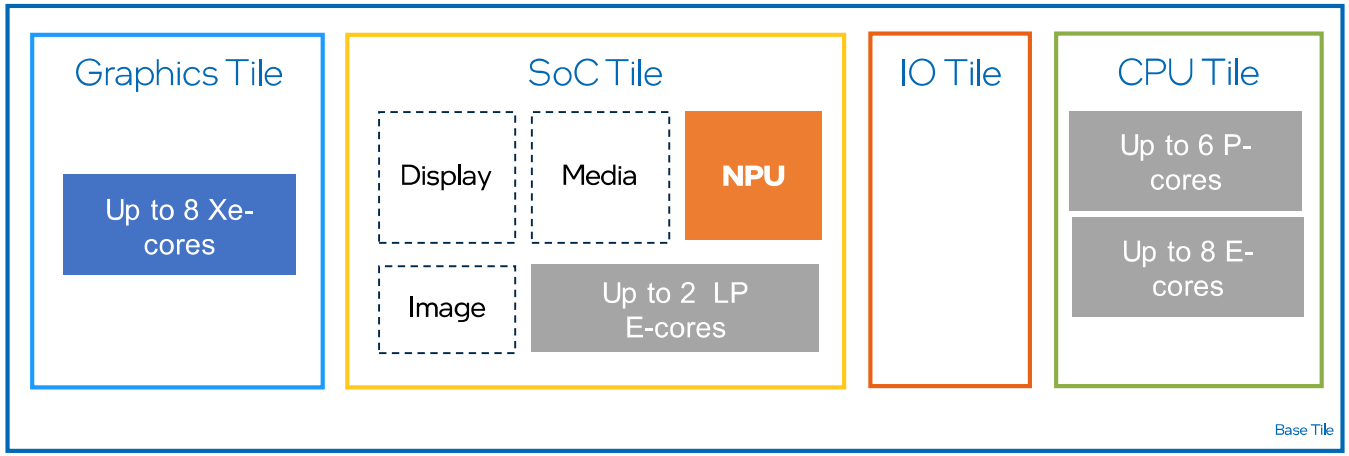
Ideal for low-latency AI workloads

GPU

High Throughput

Ideal for AI-accelerated digital content creation and gaming

Intel® Core™ Ultra Architecture

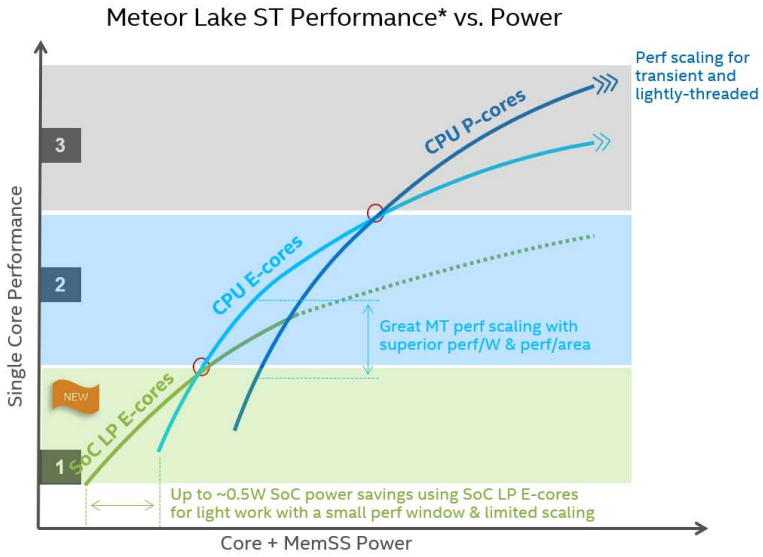


CPU

Hybrid CPU Architecture

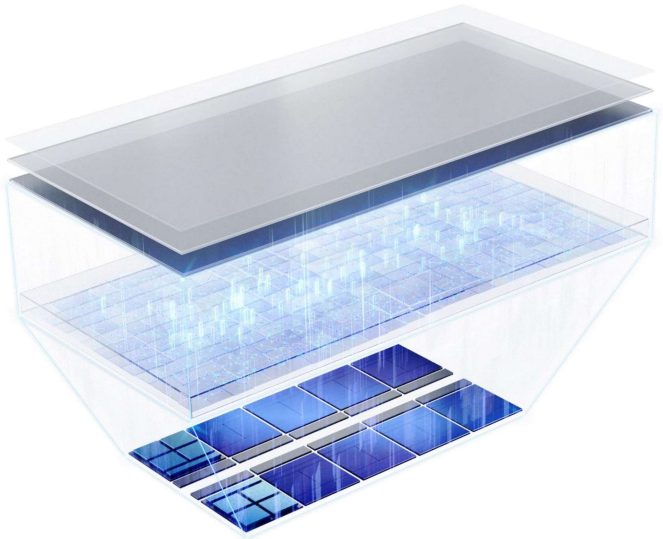
Intel 4

New 3-Level Hybrid : P-cores, E-cores, LP E-cores (up to 14 cores/20 threads)



Right cores for the right work

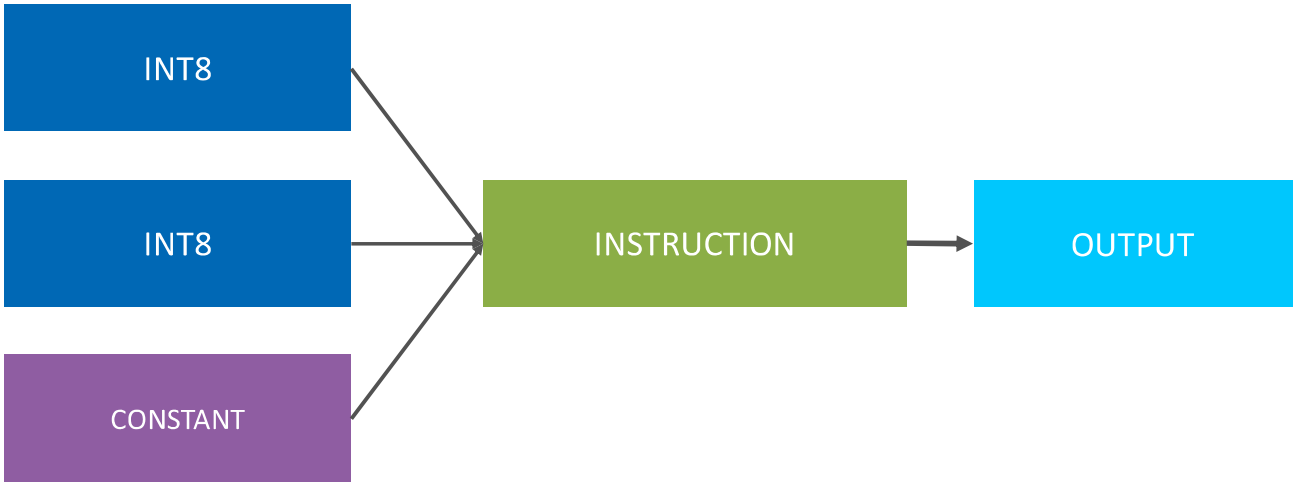
Intel® Thread Director



+ at the right time

CPU

Acceleration Capabilities
Intel® DL Boost: VNNI



8-bit INT acceleration on CPU cores, as part of AVX instructions

Combines 3 instructions into 1 to maximize compute resources

GPU

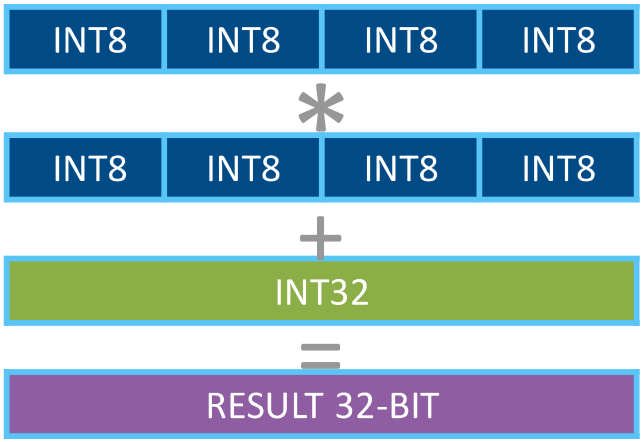
Acceleration Capabilities

4 (U-series) or 8 (H-series)

Xe LPG

Precision
FP32, FP16, INT8

Intel® DL Boost: DP4a



8-bit INT acceleration
on Intel® Xe graphics

NPU

Neural Processing Unit

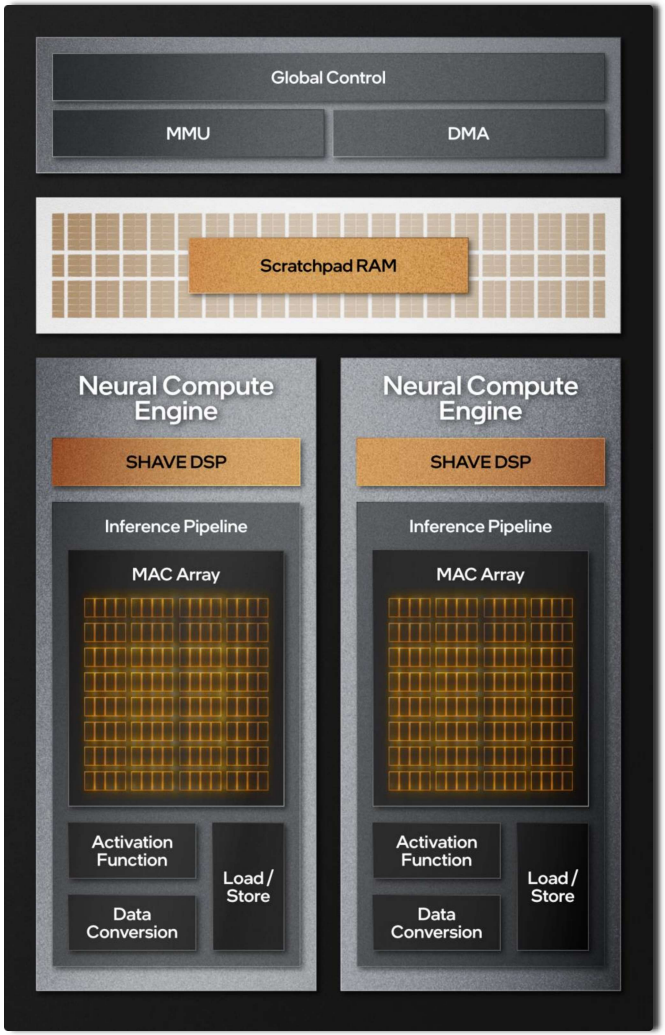
Performance, Low Power AI Compute

Mix of Fixed Function & Programmable Compute

Mixed Precision & Multi precision

Efficient & Flexible Matrix Multiplication and Convolution

Supported precisions: INT8, FP16, FP32 (emulated)



OpenVINO

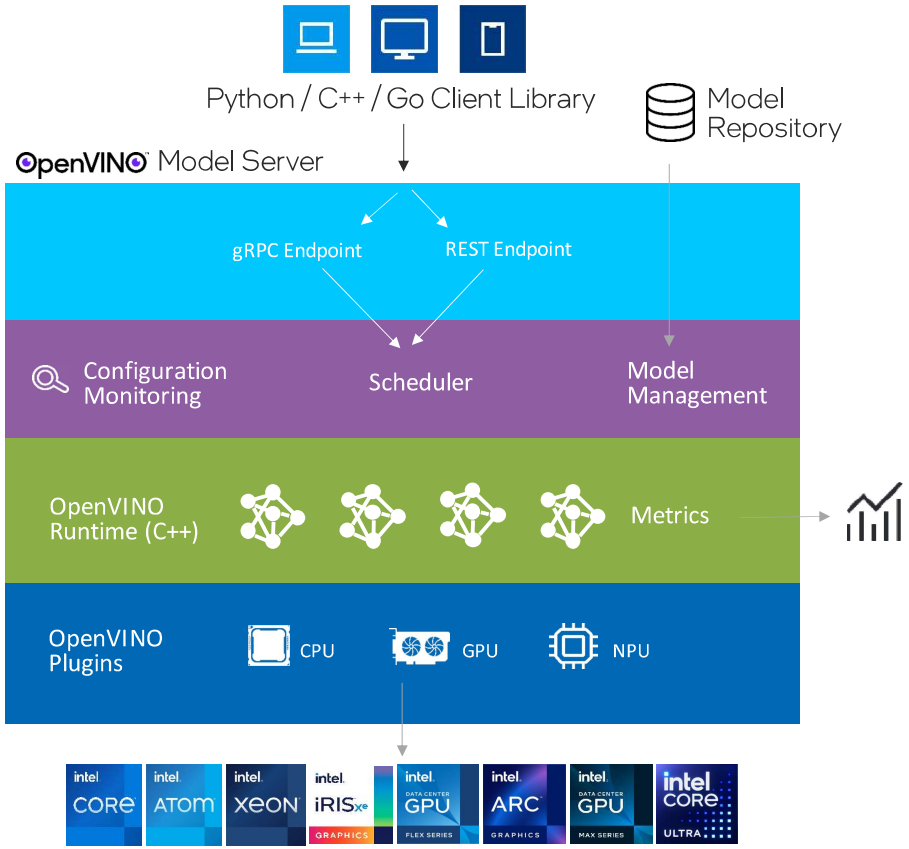
intel.

Let's Run the Demo!

OpenVINO

intel.

OpenVINO™ Model Server

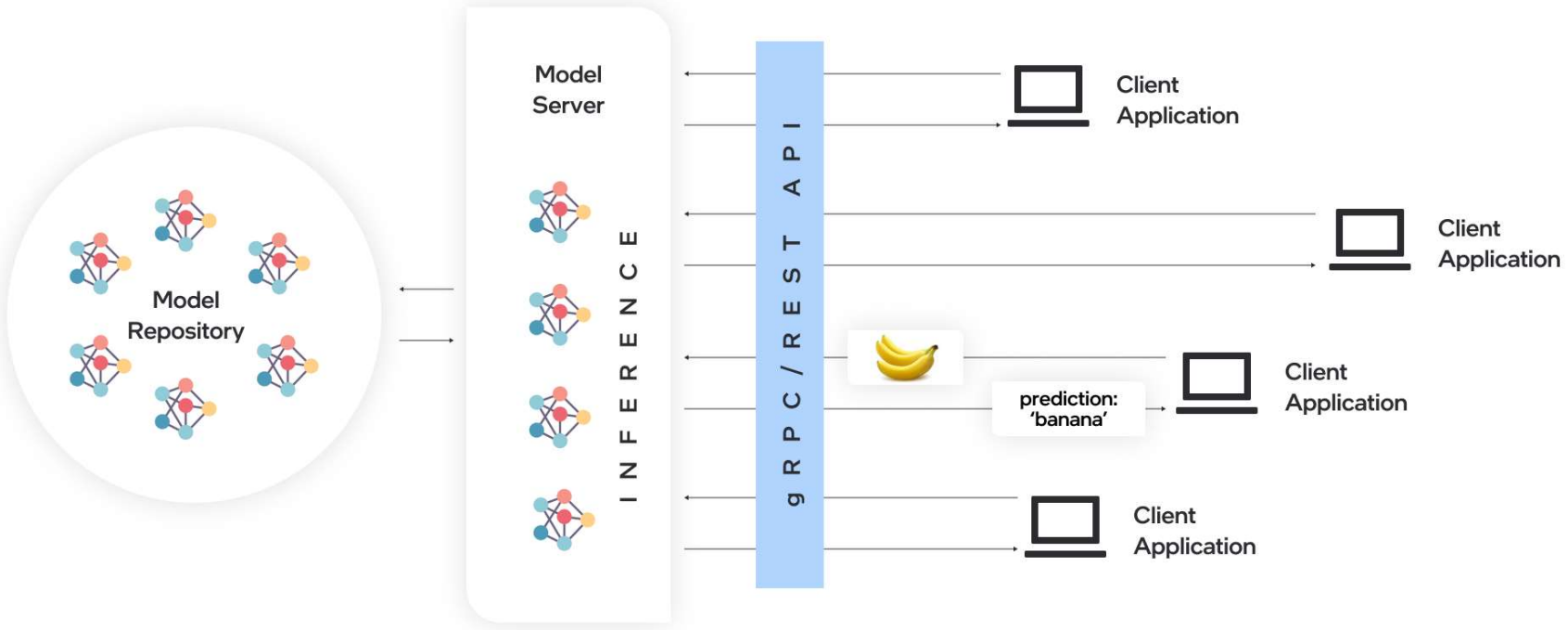


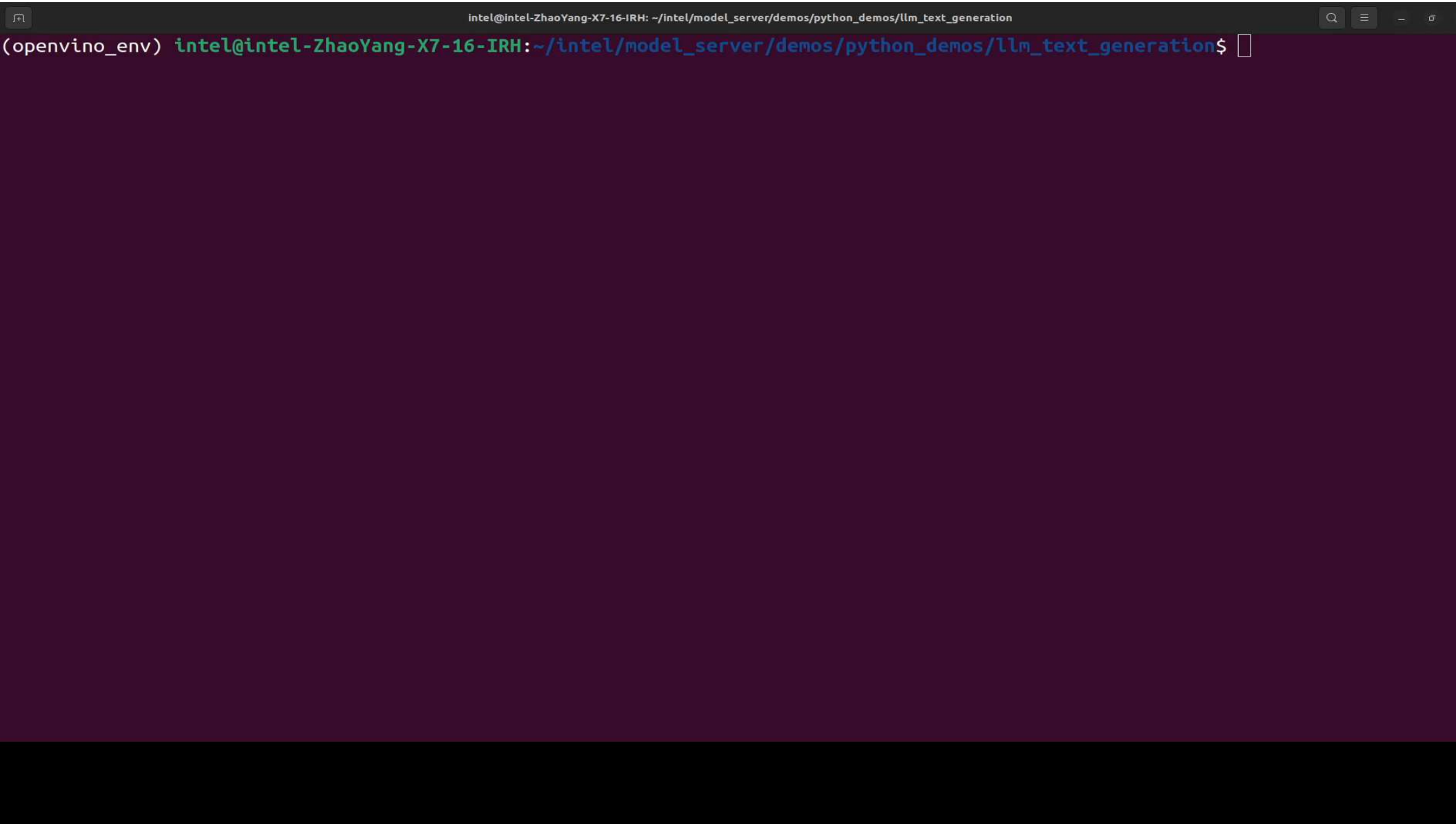
OpenVINO

intel

OpenVINO™ Model Server

Powered by OpenVINO™ Runtime




intel@intel-ZhaoYang-X7-16-IRH: ~/intel/model_server/demos/python_demos/llm_text_generation
(openvino_env) intel@intel-ZhaoYang-X7-16-IRH:~/intel/model_server/demos/python_demos/llm_text_generation$

OpenVINO™ Model Server

Run Server

```
docker run -d --rm -p 9000:9000 -v $(pwd)/onnx:/model:ro openvino/model_server \
      --port 9000 \
      --model_name gpt-j-6b \
      --model_path /model \
      --plugin_config '{"PERFORMANCE_HINT":"LATENCY","NUM_STREAMS":1}'
```

OpenVINO™ Model Server

Run Client

```
curl -X POST http://localhost:8000/v1/models/usem:predict \
-H 'Content-Type:application/json' \
-d '{"instances": ["dog", "Puppies are nice.", "I enjoy taking long
walks along the beach with my dog."]}'
```

```
from ovmsclient import make_grpc_client

client = make_grpc_client("localhost:9000")
data = ["dog", "Puppies are nice.", "I enjoy taking long walks along the
                                    beach with my dog."]
inputs = {"inputs": data}
results = client.predict(inputs=inputs, model_name="usem")
```

OpenVINO™ Training eXtensions (OTX)

01	02	03	04	05	06
Dataset	Models	BKM	Improve	Optimize	Deploy
How does my data look like	Which AI model can I use?	What are the BKM to run this model?	Is this the highest performance I can achieve?	How to maximize the model's run-time efficiency?	Any working inference example?

OpenVINO™ Training eXtensions (OTX)

- One-stop shop of verified algorithms for many vision tasks and learning methods
- Provides simple CLI and API for quick start without hassles
- Full OpenVINO integration for model optimization, inference and deployment

OpenVINO™ Training eXtensions (OTX)

CLI

```
# <train, test, export, explain, deploy>
# $ otx <entrypoint> --arg value
$ otx train \
   --task classification  \
   --data_root /path/to/data …
```

API

```
>>> from otx.engine import Engine
>>> engine = Engine(data_root="data/wgisd")
>>> engine.train()
```

One command is all you need!

OpenVINO™ Training eXtensions (OTX)

Features

Task Types

- Classification
- Detection, Rotated Detection
- Semantic and Instance Segmentation
- Anomaly Detection
- Action Recognition
- Visual Prompting

Learning Methods

- Fully-supervised
- Semi-supervised
- Self-supervised
- Class Incremental
- Imbalanced

API / CLI Functionality

- Auto-installation
- Auto-learning method
- Integrated Image Tiling
- Hyper-parameter Optimization
- OpenVINO Optimization
- Integrated Explainable AI (XAI)

OpenVINO™ Training eXtensions (OTX)

Installation

CLI

```
# <train, test, export, explain, deploy>
# Install via PyPI
$ pip install otx

# Multiple installation options
otx install --help
```

Lightweight, hardware-agnostic installation

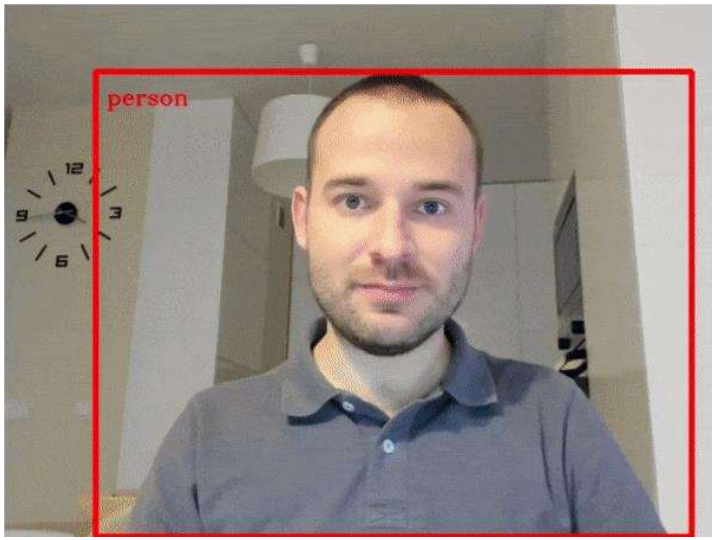
OpenVINO™ Training eXtensions (OTX)

More on this topic in Module 1
(starts 12 pm)


OpenVINO Notebooks
Run OpenVINO Applications on
CPU, GPU, and Selectively NPU
NPU
CPU
GPU
OpenVINO
intel

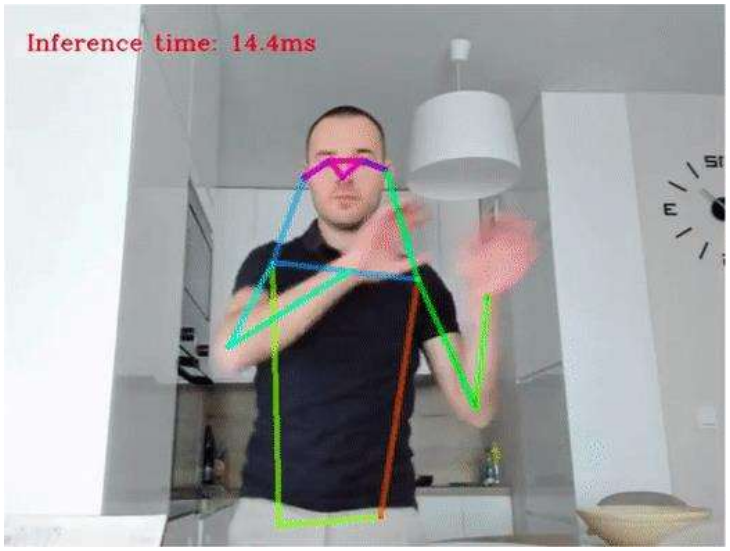
OpenVINO Notebooks

Object Detection



OpenVINO Notebooks

Pose Estimation



CVPR
JUNE 17-21, 2024
SEATTLE, WA

OpenVINO Notebooks

Segment Anything Model

OpenVINO

intel

CVPR JUNE 17-21, 2024 SEATTLE, WA

OpenVINO Notebooks

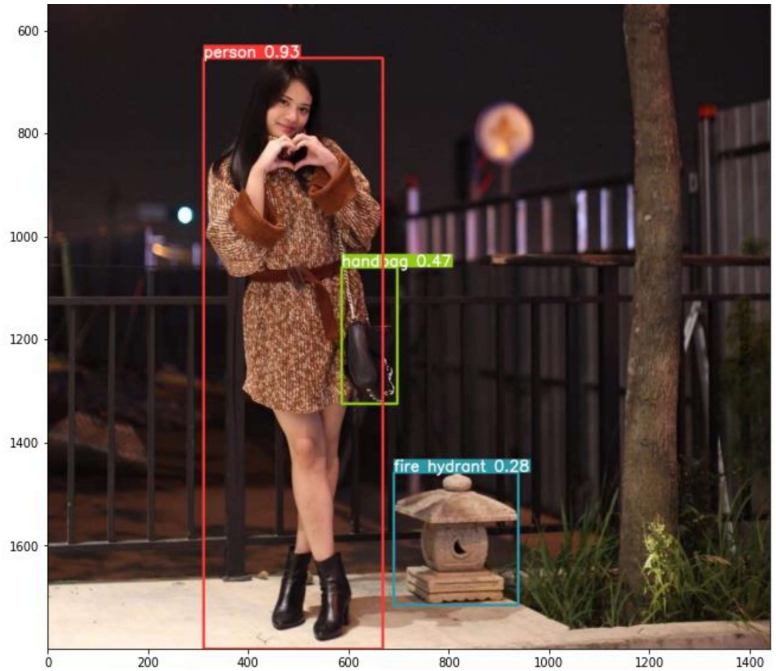
Depth Estimation

OpenVINO

intel

OpenVINO Notebooks

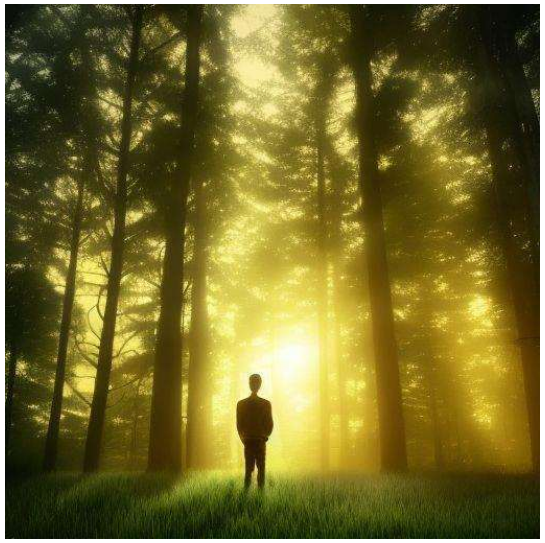
YOLOv8 YOLO



OpenVINO Notebooks

Stable Diffusion





CVPR JUNE 17-21, 2024 SEATTLE, WA

OpenVINO Notebooks

Latent Consistency Models

OpenVINO

intel

Latent Consistency Models (LCM)

More on this topic (inc. LoRA) in Module 2
(starts 1 pm)

OpenVINO Notebooks

Control Net

OpenVINO

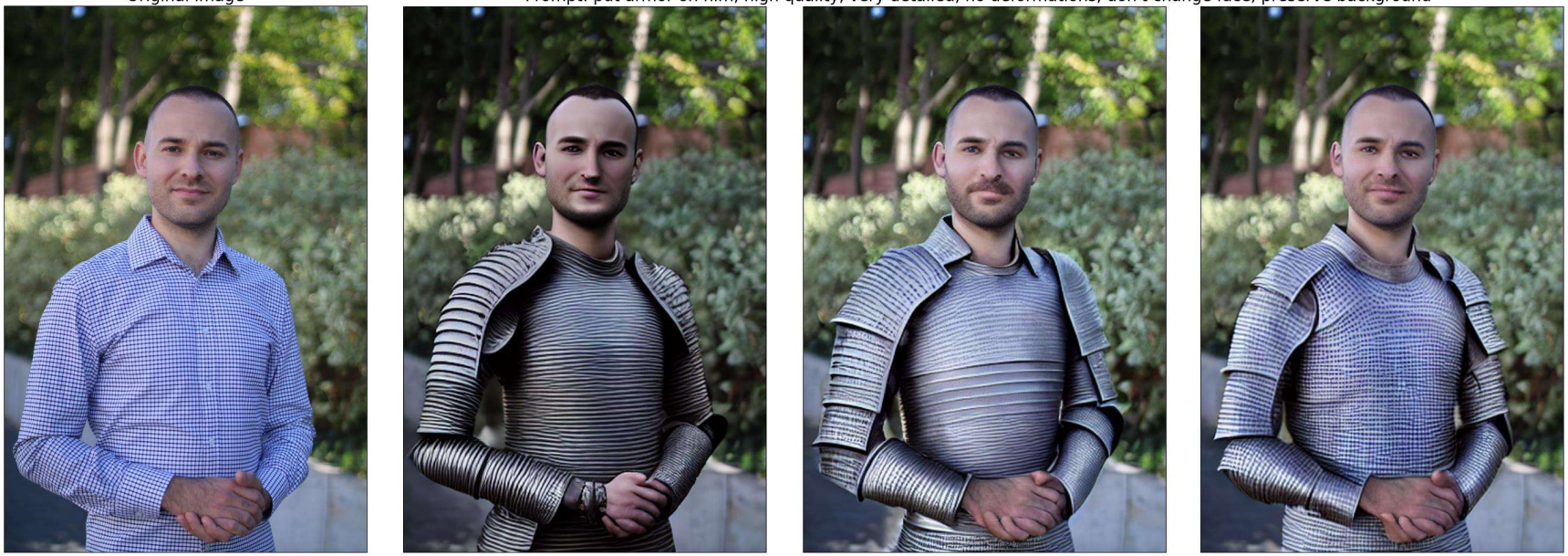
intel

OpenVINO Notebooks

Pix2Pix

Original image

Prompt: put armor on him, high quality, very detailed, no deformations, don't change face, preserve background



OpenVINO Notebooks

QR Code Monster

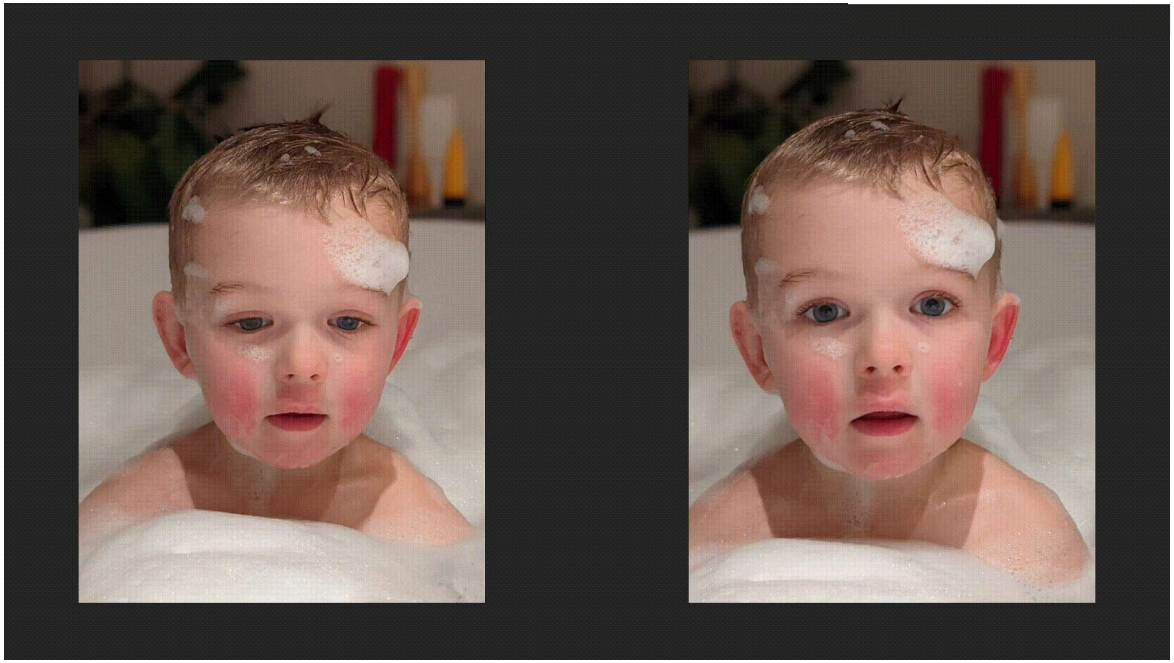
OpenVINO Notebooks

Text to Video (ZeroScope)

A panda eating bamboo on a rock

OpenVINO Notebooks

Film-Slowmo



OpenVINO

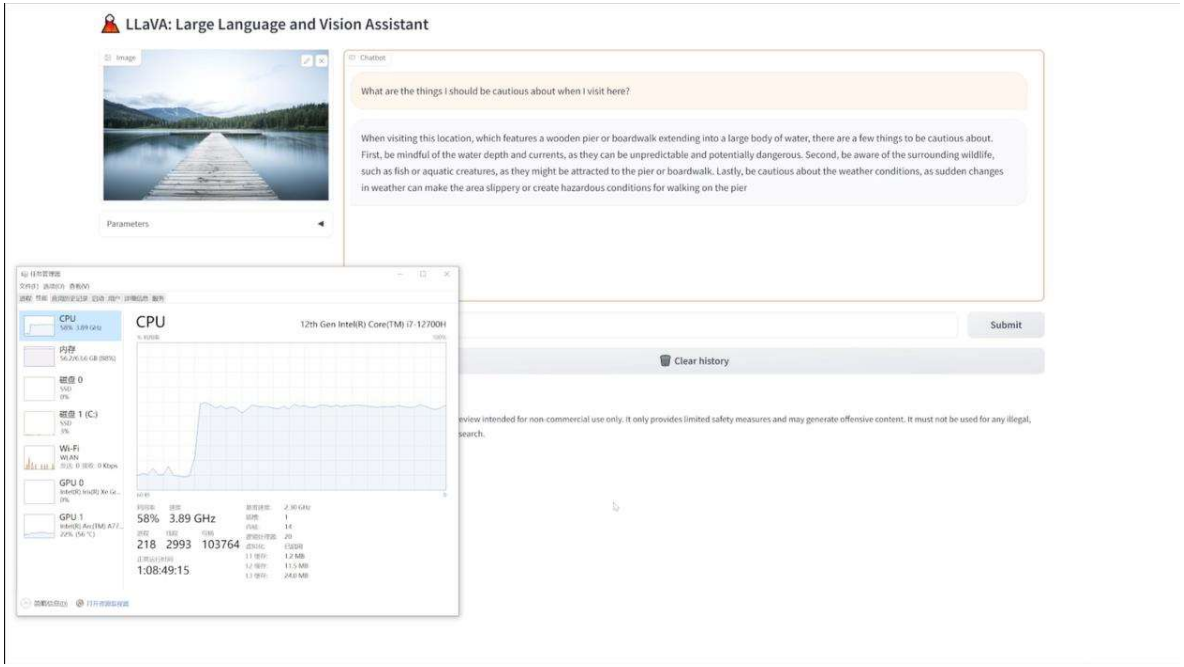
intel

CVPR JUNE 17-21, 2024 SEATTLE, WA

OpenVINO Notebooks

LLaVA

(Large Language and Vision Assistant)



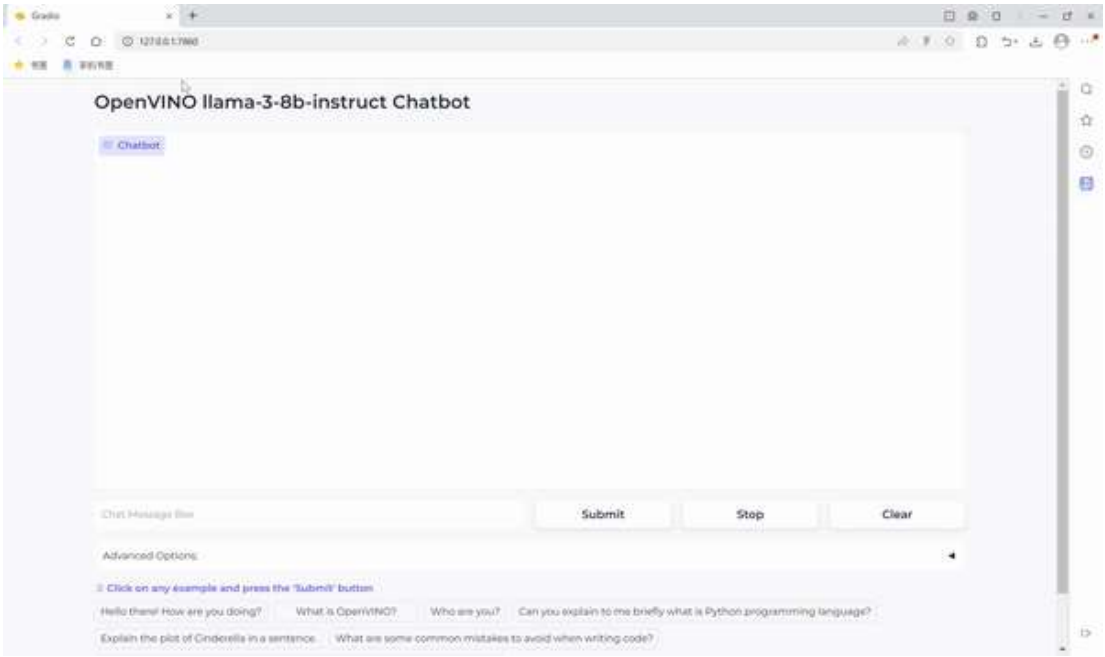
OpenVINO

intel

CVPR JUNE 17-21, 2024 SEATTLE, WA

OpenVINO Notebooks

LLama3 Chatbot



OpenVINO

intel

CVPR
JUNE 17-21, 2024
SEATTLE, WA

OpenVINO Notebooks 📚

Music Generation, Text to Speech (Bark), Speech to Text (Whisper), Diarization ...

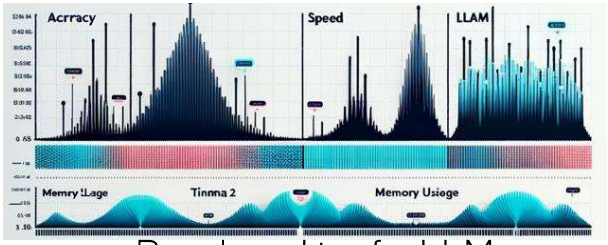
🎶 🎤 💬 📖

OpenVINO™

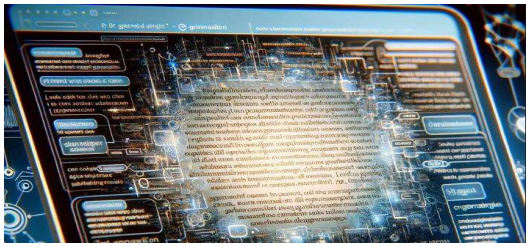
intel.

OpenVINO™ GenAI Pipeline Repo

OpenVINO Native C&C++ Pipeline for Gen AI and LLM



Benchmarking for LLMs



Text generation C++ samples

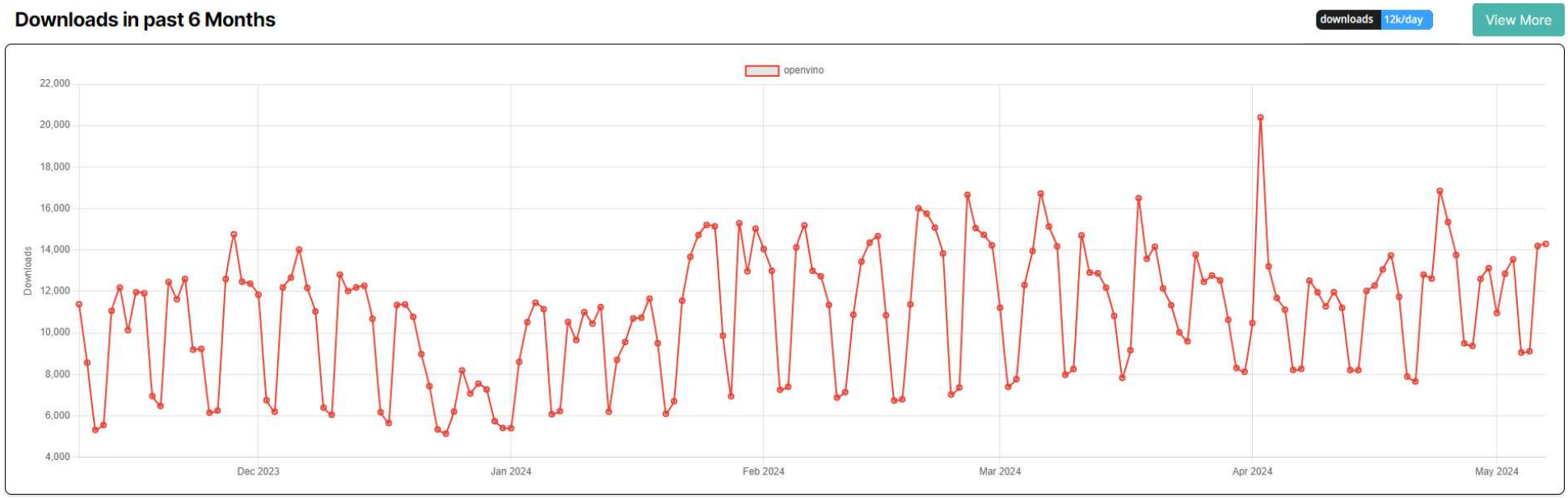
Stable Diffusion (with LoRA) C++ image Generation Pipeline

LCM (with LoRA) C++ image Generation Pipeline

OpenVINO

intel.

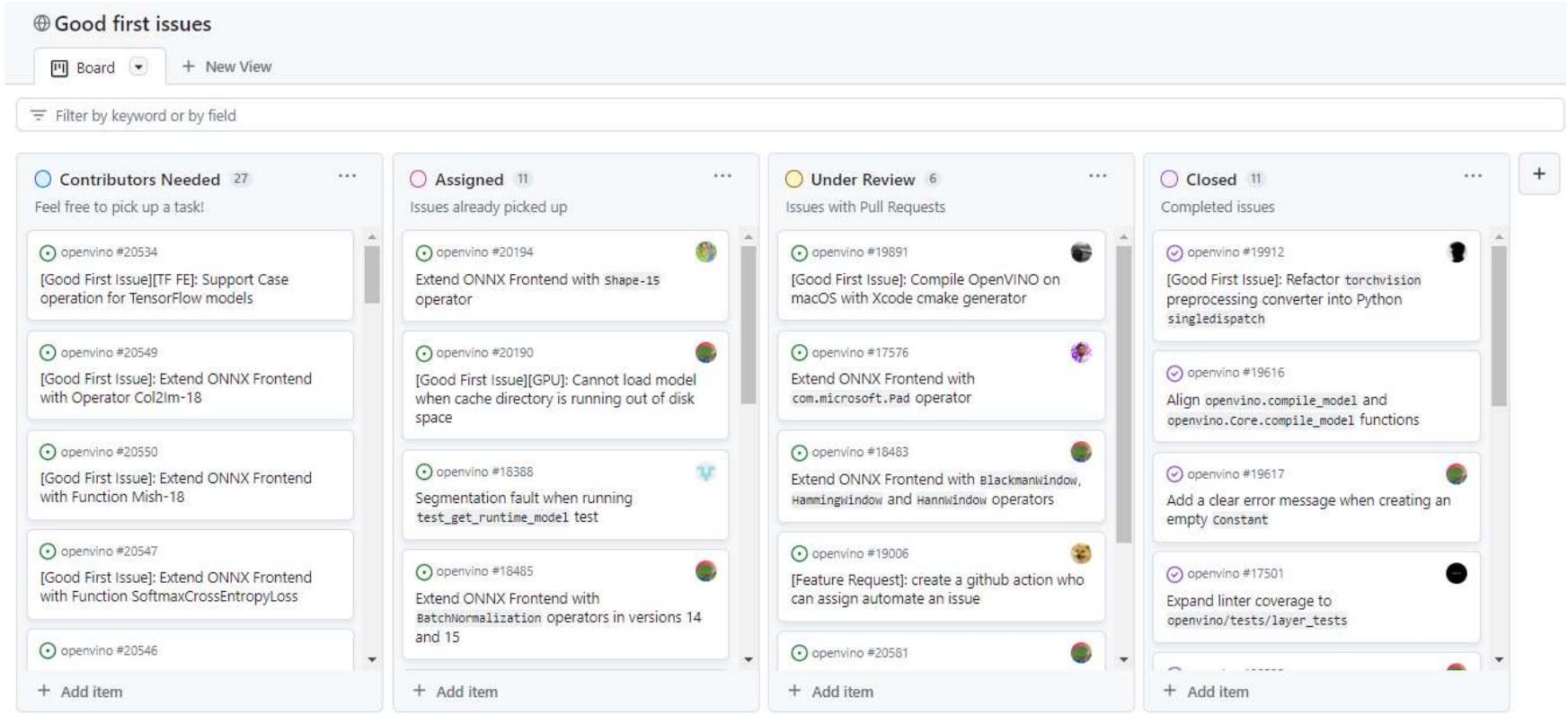
OpenVINO™ Ecosystem Adoption



Over 3,500,000 downloads!

Source: https://piptrends.com/package/openvino

Contribute to OpenVINO™ Toolkit



How to start: https://medium.com/openvino-toolkit/how-to-contribute-to-an-ai-open-source-project-c741f48e009e

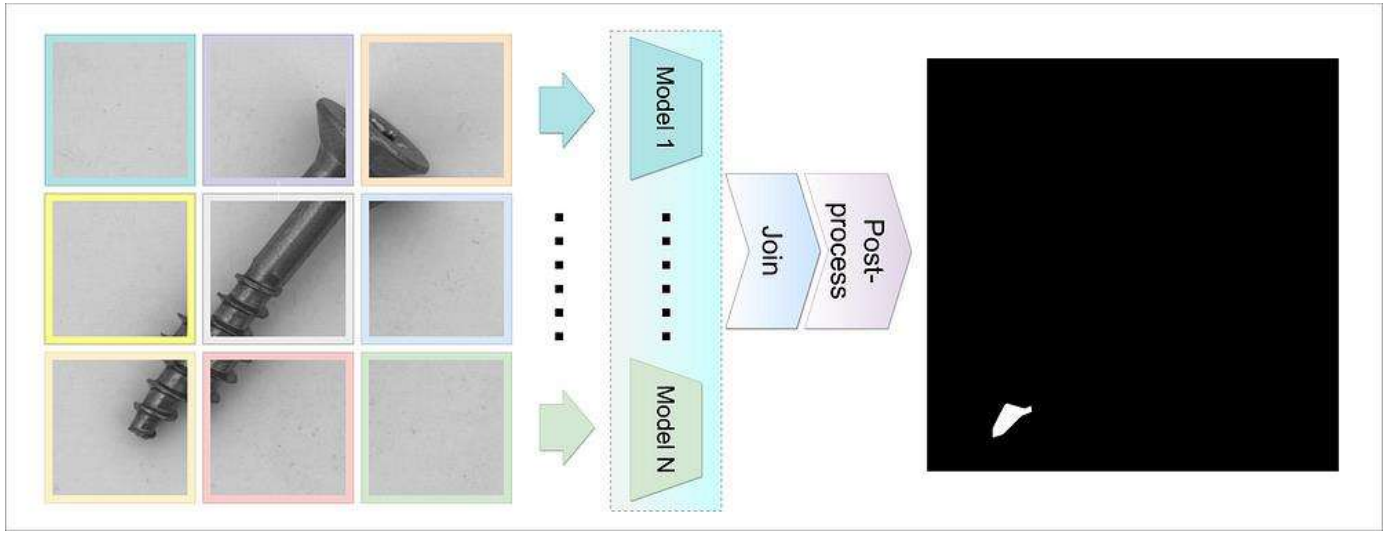
Google Summer of Code

+ OpenVINO™

https://github.com/openvinotoolkit/openvino/wiki/Google-Summer-Of-Code

Google Summer of Code 2024

Divide and Conquer: High-Resolution Industrial Anomaly Detection via Memory Efficient Tiled Ensemble



at VAND 2.0: Visual Anomaly and Novelty Detection - 2nd Edition


Connect
With Us
OpenVINO
intel


lu.ma
6:05 AM EDT
Explore Events
Sign In
Networking
Meetup
@CVPR
intel ai
Paula Ramos invites you to join
CVPR Networking Meetup:
sponsored by Intel
JUN
20
Thursday, June 20
7:00 PM - 10:00 PM PDT
Register to See Address
Seattle, Washington
Registration
Approval Required
Your registration is subject to approval by the host.
Welcome! To join the event, please register below.
Request to Join
About Event
Uzair Ahmed

intel innovation

AI: The New Age
Solving the World's Toughest Challenges, **Together**.

Calling All Developers & Technologists!

From front-end, web, app devs to back-end, full-stack, database & DevOps to data scientists & researchers, and more: Learn, collaborate, and solve at Intel Innovation – an event for developers by developers.

- Hear from leading industry luminaries, technologists & start-up entrepreneurs in the field of AI.
- Get the latest AI development tools, hands-on experience & join on-site Hackathons to optimize your AI code & workflows.
- Learn the breadth of future technology advancements in AI through keynotes, sessions, birds of a feathers, and hands-on labs.
- Share unique ideas and perspectives and collaborate with your peers.

Save the Date:
September 24-25, 2024
San Jose Convention Center, CA

Opt-In for Early Access When Registration Opens!

www.intel.com/innovation

intel

CTA

Sign up for receiving the latest updates and news from the AI PC Developer program

AI PC Developer Program

- Link: https://www.intel.com/content/www/us/en/developer/topic-technology/ai-pc/overview.html#gs.85rzmn

Related Products

- NPU Documentation
- Built-in GPU
- Intel® Core™ Ultra processor


intel
Thank You
<speaker>
<speaker>@intel.com


intel

Notices and Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.

Intel technologies may require enabled hardware, software or service activation.

Your costs and results may vary.

Intel is committed to respecting human rights and avoiding complicity in human rights abuses. See Intel's Global Human Rights Principles. Intel's products and software are intended only to be used in applications that do not cause or contribute to a violation of an internationally recognized human right.