



Edge AI Handbook: A Practical Approach

Handbook

January 2026



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

No product or component can be absolutely secure.

The code names presented in this document are only for use by Intel to identify products, technologies, or services in development, that have not been made commercially available to the public, i.e., announced, launched or shipped. They are not "commercial" names for products or services and are not intended to function as trademarks.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Contents

1.0	Introduction	7
1.1	Terminology	7
2.0	Processor SKU Selection	10
2.1	OpenVINO™ 2025 Performance Benchmarks	11
3.0	System Setup.....	20
3.1	OpenVINO™ Toolkit Bare Metal Setup with Virtual Environment	20
3.1.1	On Linux* OS	20
3.1.2	Installation Steps for Windows* OS.....	25
3.2	OpenVINO™ Toolkit Setup with Containerization.....	28
3.2.1	Install Operating System, Drivers, and Required Packages	28
3.2.2	Install Docker*	28
3.2.3	Build OpenVINO™ xPU Docker* Image	29
3.2.4	Build OpenVINO™ Notebooks Docker* Image	32
3.3	Geti™ Application Setup on Premise	33
3.3.1	Install Operating System and Upgrade Linux* Kernel	33
3.3.2	Install Intel® GPU Packages	34
3.3.3	Install Geti™ Application.....	36
4.0	System Profiling	37
4.1	top.....	37
4.2	Intel® GPU Top.....	38
4.3	Intel® Performance Counter Monitor	38
4.4	Intel® XPU-SMI	40
5.0	Accuracy Optimization.....	42
5.1	Geti™ Application Accuracy Model Selection.....	42
5.2	Geti™ Application Model Input Selection.....	42
6.0	Performance Optimization.....	44
6.1	OpenVINO™ Toolkit Model Conversion	44
6.2	OpenVINO™ Toolkit Model Compression	53
6.3	Geti™ Application Model Quantization.....	54
6.4	Channels-First and Channels-Last	54
6.5	Static and Dynamic Input Shapes.....	55
6.6	Input Batching	55
6.7	Input and Output Sizes.....	55
6.8	Heterogeneous Processing	56
6.9	Asynchronous Inferencing using Geti™ Application SDK.....	59
6.10	Core Pinning	60
6.11	Model Caching.....	60

7.0	Deployment	61
7.1	Intel® DL Streamer	61
7.1.1	Modify Pre- and Post-Processing	61
7.1.2	Build Intel® DL Streamer Docker* Image.....	63
7.1.3	Run Intel® DL Streamer Docker* Image	63
7.2	OpenVINO™ Toolkit Model Server	65
7.2.1	Spin Up the OpenVINO™ Toolkit Model Server	66
7.2.2	Perform Client Inference	66
7.3	Geti™ Application SDK	67
7.3.1	Build Geti™ Application SDK with Jupyter* Notebook Docker* Image	67
7.3.2	Run Geti™ Application SDK with Jupyter* Notebook Docker* Image	68
8.0	Applications.....	69
8.1	Text QnA.....	69
8.1.1	Architecture.....	70
8.1.2	Implementation	72
8.2	Image QnA	77
8.2.1	Architecture.....	78
8.2.2	Implementation	80
8.3	Video QnA.....	90
9.1.1	Architecture.....	91
9.1.2	Implementation	92

Figures

Figure 1.	Python* Download Page on Microsoft Store*	26
Figure 2.	Intel® PCM Sample Output	39
Figure 3.	Geti™ Application Model Selection	42
Figure 4.	Geti™ Application Model Input Tiling.....	43
Figure 5.	Geti™ Application Model Input Size	43
Figure 6.	Geti™ Application Model Quantization.....	54
Figure 7.	Intel® DL Streamer OpenVINO™ Toolkit Input Image Loading	62
Figure 8.	Intel® DL Streamer Pre-Processing	62
Figure 9.	Intel® DL Streamer Geti™ Application Post-Processing.....	62
Figure 10.	Text QnA Hardware Components Diagram	70
Figure 11.	Text QnA Software Deployment Diagram.....	71
Figure 12.	Text QnA Software Sequence Diagram	72
Figure 13.	Image QnA Hardware Components Diagram.....	78
Figure 14.	Image QnA Software Deployment Diagram	79
Figure 15.	Image QnA Software Sequence Diagram	79
Figure 16.	Object Detection Model Selection.....	88
Figure 17.	Object Detection Compute Unit Selection.....	88
Figure 18.	Object Detection and VLM Input Image	88
Figure 19.	Object Detection Output Image.....	89

Figure 20.	Image QnA Compute Unit Selection.....	89
Figure 21.	Image QnA Model Quantization Selection.....	89
Figure 22.	Image QnA Output Text.....	90
Figure 23.	Video QnA Hardware Components Diagram	91
Figure 24.	Video QnA Software Deployment Diagram	92
Figure 25.	Video QnA Software Sequence Diagram.....	92
Figure 26.	Video QnA Output Text.....	99

Tables

Table 1.	Terminology	7
Table 2.	Model Information.....	10
Table 3.	ResNet50 FP16 Latency in Milliseconds.....	12
Table 4.	ResNet50 FP16 Throughput in Frames per Second	12
Table 5.	ResNet50 INT8 Latency in Milliseconds	13
Table 6.	ResNet50 INT8 Throughput in Frames per Second	13
Table 7.	Ultralytics* YOLO11 INT8 Latency in Milliseconds	14
Table 8.	Ultralytics* YOLO11 INT8 Throughput in Frames per Second.....	14
Table 9.	Qwen3-8B INT4 First Token Latency in Milliseconds.....	15
Table 10.	Qwen3-8B INT4 Throughput in Tokens per Second.....	15
Table 11.	Gemma3-4B INT4 First Token Latency in Milliseconds	16
Table 12.	Gemma3-4B INT4 Throughput in Tokens per Second	16
Table 13.	StableDiffusion-V1-5 INT8 Latency in Seconds	17
Table 14.	OpenVINO™ Toolkit 2025 Performance Benchmarks Hardware Platforms.....	17
Table 15.	OpenVINO™ Toolkit 2025 Performance Benchmarks Software Platforms.....	19
Table 16.	YOLOv7 Heterogeneous Processing Software BOMs.....	56
Table 17.	Text QnA Hardware and Software BOMs	69
Table 18.	Image QnA Hardware and Software BOMs.....	77
Table 19.	Video QnA Hardware and Software BOMs.....	90

Revision History

Date	Revision	Description
January 2026	1.0	Initial release.

§

1.0 Introduction

This handbook will cover a practical approach to edge AI application development. It spans system selection, system setup, system profiling, accuracy optimization, performance optimization, and deployment. It also includes sample applications for implementation references.

There are many benchmark results for AI tasks performed at the edge. They are important for initial system selection. The system setup section covers both bare-metal and containerization methods. Now that the system can be used for model benchmarking, system profiling becomes critical to measure resource allocation. Whether the model accuracy or the model performance is not up to satisfaction, accuracy optimization and performance optimization sections share some tips to tune them. The system setup section also includes steps for installing training tools to improve model accuracy. Having acceptable model accuracy and performance, the deployment section provides both streaming and serving inference pipeline methods.

The applications are commonly referred to as edge-native, meaning they are born at the edge rather than in the cloud (cloud-native). One characteristic of edge native applications is streaming around the clock. It will be very expensive to stream videos from multiple cameras to the cloud all year round. Another characteristic is real-time processing. These applications require an instantaneous response, with latency as the Key Performance Indicator (KPI) rather than throughput. Let alone processing the data in the cloud, processing the data on a discrete GPU is already considered slow as compared to an integrated GPU. The final characteristic is security. The user wants to control where the data is stored and how it is being used. So, edge native is synonymous with secure real-time streaming.

This handbook is far from sufficient. For more information, please consult the processor, system, and tool setup documentation and requirements.

1.1 Terminology

Table 1. Terminology

Term	Description
AI	Artificial Intelligence
ASR	Automatic Speech Recognition
AWQ	Activation-Aware Weight Quantization
BIOS	Basic Input/Output System

Term	Description
CLI	Command Line Interface
Cmd	Command
CPU	Central Processing Unit
dGPU	Discrete Graphics Processing Unit
DL	Deep Learning
DP	DisplayPort
E-core	Efficient Core
FP16	Floating Point 16-bit
FP32	Floating Point 32-bit
FPS	Frames per Second
GFLOps	Giga Floating Point Operations per Second
GPU	Graphics Processing Unit
GRUB	GRand Unified Bootloader
HDMI	High-Definition Multimedia Interface
ID	Identifier
iGPU	Integrated Graphics Processing Unit
INT4	Integer 4-bit
INT8	Integer 8-bit
I/O	Input/Output
IP	Internet Protocol
IR	Intermediate Representation
KPI	Key Performance Indicator
LLM	Large Language Model
LPE-core	Low Power Efficient-cores
LTS	Long Term Support
N/A	Not Applicable
NLP	Natural Language Processing

Term	Description
NNCF	Neural Network Compression Framework
NPU	Neural Processing Unit
NUC	Next Unit of Computing
OEM	Original Equipment Manufacturer
ONNX	Open Neural Network Exchange
OS	Operating System
OVMS	OpenVINO™ Model Server
P-core	Performance Core
PoE	Power over Ethernet
POS	Points of Sale
px	Pixel
QnA	Question and Answer
RAM	Random Access Memory
SDK	Software Development Kit
SKU	Stock Keeping Unit
TTS	Text to Speech
URL	Uniform Resource Locator
USB	Universal Serial Bus
VLM	Visual Language Model
VQA	Visual Question Answering
WSL	Windows* Subsystem for Linux*

2.0 Processor SKU Selection

Edge AI applications span many industries, including but not limited to smart cities, infrastructure, aerospace, retail, education, healthcare, federal and governmental, manufacturing, and robotics. It also reflects the variety of systems across the entry, light, medium, and heavy AI categories. The systems for entry AI are normally sub 15 watts that can only process one camera up to 30 frames per second (FPS). For light AI systems, the power range is 15 to 65 watts without dGPU. The light AI with additional 50, 100, or 200 watts dGPU falls into medium AI category. Finally, the heavy AI systems have server processors with over 65 watts CPU and multiple 100s of watts of dGPUs.

This section will help you get a rough idea of which system is suitable for your application using [OpenVINO™ 2025 Performance Benchmarks Results](#). If you are planning to use OpenVINO™ toolkit 2024, please refer to the [OpenVINO™ 2024 Performance Benchmarks](#) results.

For the edge AI system categories, this section focuses on light AI using [i2700H](#), [i55H](#), [i288V](#), and [i265H](#) CPUs as well as medium AI with the addition of [A750E](#), [A770M](#), and [B580](#) dGPUs. If the light AI systems are too powerful for your application, you can use entry AI systems ([N50](#) to [N355](#)). In contrast, if you need more performant systems than the ones in the medium AI category, please consider the systems from the heavy AI category ([Intel® Xeon® for Edge](#) with [Intel® Arc™ Pro dGPU](#)).

Please find Table 2 for the curated models presented in this section. They represent image classification, object detection, language, and generative tasks. Different numbers of parameters, input, and output will affect the performance results.

Table 2. Model Information

Name (Category)	Complexity (GFLOps)	# Parameters in Million	Input	Output
ResNet50 (Image Classification)	8.22	25	224x224px	1000 classes
YOLO11 (Object Detection)	6.50	3	640x640px	80 classes
Qwen3-8B (Language)	N/A	8,000	1024 tokens	128 tokens
Gemma3-4B (Language)	N/A	4,000	1024 tokens	128 tokens
StableDiffusion-V1-5 (Generative)	N/A	1,000	1024 tokens	256x256px



2.1 OpenVINO™ 2025 Performance Benchmarks

Table 3 and Table 4 show ResNet50 with FP16 quantization latency and throughput results, respectively. Whereas Table 5 and Table 6 show the INT8 quantization variant. Please note that INT8 quantized model will normally have accuracy penalty in favor of better performance. For FP16 model, the best latency for light AI category is 1.32ms running on 265H iGPU. For medium AI, a B50 dGPU can reduce latency to 0.53ms.

Similarly, for Ultralytics* YOLO11 model, light AI can achieve 2.94ms latency, and medium AI can reduce it down to 1.65ms. Please find the model latency and throughput on Table 7 and Table 8 respectively.

Table 9 and Table 10 show Qwen3-8B time to first token as well as the token per second, respectively. 288V iGPU (light AI) can get 21 tokens/s and B580 dGPU (medium AI) improves it to 67 tokens/s.

Table 11 and Table 12 show Gemma3-4B with 4 billion parameters as compared to Qwen3-8B with 8 billion parameters. Using fewer model parameters, 155H iGPU achieves 22 tokens/s, similar to 265H iGPU with 21 tokens/s, despite the 4x compute power difference. But please note that the time to the first token differs from 1.5s to 0.5s.

Table 13 shows StableDiffusion-V1-5 image generation latency. 288V iGPU (light AI) can generate 20 20-iteration image in 4 seconds, whereas B580 dGPU (medium AI) can do it in 1 second.

Table 14 and Table 15 show the benchmark hardware and software platforms.

§

Table 3. ResNet50 FP16 Latency in Milliseconds

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	-	-	-	-	N/A	N/A	N/A
iGPU	5.00	2.39	1.48	1.32	N/A	N/A	N/A
NPU	N/A	2.83	-	2.74	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	1.29	1.44	0.53

Table 4. ResNet50 FP16 Throughput in Frames per Second

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	-	-	-	-	N/A	N/A	N/A
iGPU	221.15	366.83	683.85	749.11	N/A	N/A	N/A
NPU	N/A	396.67	891.16	438.17	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	1144.83	1161.69	1469.32



Table 5. ResNet50 INT8 Latency in Milliseconds

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	3.17	6.14	4.53	2.56	N/A	N/A	N/A
iGPU	3.32	1.60	0.73	0.85	N/A	N/A	N/A
NPU	N/A	1.55	0.78	1.48	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	1.05	1.31	0.37

Table 6. ResNet50 INT8 Throughput in Frames per Second

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	430.8	414.55	297.91	594.78	N/A	N/A	N/A
iGPU	362.32	541.36	1394.68	1172.92	N/A	N/A	N/A
NPU	N/A	774.81	1829.65	836.85	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	1577.84	1487.67	1464.16

Table 7. Ultralytics* YOLO11 INT8 Latency in Milliseconds

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	6.44	-	8.79	10.37	N/A	N/A	N/A
iGPU	7.00	3.17	-	2.94	N/A	N/A	N/A
NPU	N/A	-	5.50	11.71	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	3.44	4.27	1.65

Table 8. Ultralytics* YOLO11 INT8 Throughput in Frames per Second

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	194.91		133.46	218.86	N/A	N/A	N/A
iGPU	177.40	271.16		343.97	N/A	N/A	N/A
NPU	N/A		265.99	108.12	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	425.29	335.8	514.59



Table 9. Qwen3-8B INT4 First Token Latency in Milliseconds

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	-	-	-	-	N/A	N/A	N/A
iGPU	-	2382.00	521.28	659.26	N/A	N/A	N/A
NPU	N/A	-	-	-	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	216.00	226.87	154.27

Table 10. Qwen3-8B INT4 Throughput in Tokens per Second

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	-	-	-	-	N/A	N/A	N/A
iGPU	-	14.00	21.09	18.03	N/A	N/A	N/A
NPU	N/A	-	-	-	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	52.60	45.68	67.11

Table 11. Gemma3-4B INT4 First Token Latency in Milliseconds

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	-	-	-	-	N/A	N/A	N/A
iGPU	-	1498.00	-	500.74	N/A	N/A	N/A
NPU	N/A	2603.00	614.04	2441.00	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	207.00	264.68	148.74

Table 12. Gemma3-4B INT4 Throughput in Tokens per Second

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	-	-	-	-	N/A	N/A	N/A
iGPU	-	22.60	-	21.72	N/A	N/A	N/A
NPU	N/A	13.10	23.27	14.66	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	41.90	35.97	53.96

Table 13. StableDiffusion-V1-5 INT8 Latency in Seconds

Device	12700H	155H	288V	265H	A750E	A770M	B580
CPU	-	-	-	-	N/A	N/A	N/A
iGPU	-	12.80	3.86	4.53	N/A	N/A	N/A
NPU	N/A	-	-	-	N/A	N/A	N/A
dGPU	N/A	N/A	N/A	N/A	1.90	2.12	0.98s

Table 14. OpenVINO™ Toolkit 2025 Performance Benchmarks Hardware Platforms

Platform	12700H	155H	288V	265H	A750E	A770M	B580
System	Intel® NUC, NUC12SNKi72	AsusTek* NUC, NUC14RVBU7	AsusTek* NUC 9-288V	AsusTek* NUC, NUC15CRBV7	Intel® NUC, NUC13SBBi9	Intel® NUC, NUC12SNKi72	ABS*, ASRock* B760M-C/D4
Processor	Intel® Core™ i7-12700H	Intel® Core™ Ultra 7 Processor 155H	Intel® Core™ Ultra 9 Processor 288V	Intel® Core™ Ultra 7 Processor 265H	Intel® Arc™ A750E Graphics	Intel® Arc™ A770M Graphics	Intel® Arc™ B580 Graphics
Launch Date	Q1'22	Q4'23	Q3'24	Q1'25	Q3'24	Q2'22	Q4'24
Code Name	Alder Lake	Meteor Lake	Lunar Lake	Arrow Lake	Alchemist	Alchemist	Battlemage
Vertical Segment	Mobile	Mobile	Mobile	Mobile	Embedded	Mobile	Desktop
Lithography	Intel® 7	Intel® 4	TSMC* N3B	TSMC* N3B	TSMC* N6	TSMC* N6	TSMC* N5
Total Cores	14	16	8	16	N/A	N/A	N/A

Platform	12700H	155H	288V	265H	A750E	A770M	B580
# of Performance-cores	6	6	4	6	N/A	N/A	N/A
# of Efficient-cores	8	8	N/A	8	N/A	N/A	N/A
# of Low Power Efficient-cores	N/A	2	4	2	N/A	N/A	N/A
Total Threads	20	22	8	16	N/A	N/A	N/A
Hyper Threading	On	On	Off	N/A	N/A	N/A	N/A
Turbo Setting	On	On	On	On	N/A	N/A	N/A
Processor Base Power	45 W	28 W	30 W	28 W	225 W	150 W	190 W
Memory	2 x 8 GB DDR4 3200 MT/s	2 x 16 GB DDR5 5600 MT/s	8 x 4 GB LPDDR5 7467 MT/s	2 x 16 GB DDR5 6400 MT/s	16 GB GDDR6	16 GB GDDR6	12 GB GDDR6
Max Memory Size	64 GB	96 GB	32 GB	128 GB	16 GB	16 GB	12 GB
GPU Name	Intel® Iris® Xe Graphics	Intel® Arc™ graphics	Intel® Arc™ 140V GPU	Intel® Arc™ 140T GPU	N/A	N/A	N/A
GPU Execution Units	96	128	128	128	N/A	N/A	N/A
X ^e -cores	N/A	8	8	8	28	32	20



Table 15. OpenVINO™ Toolkit 2025 Performance Benchmarks Software Platforms

Platform	12700H	155H	288V	265H	A750E	A770M	B580
Operating System	Ubuntu* 24.04.3 LTS	Windows* 11 Pro	Windows* 11 Enterprise	Windows* 11	Ubuntu* 24.04.3 LTS	Windows* 11	Windows* 11
Kernel Version	6.8.0-85-generic	10.0.26200 Build 26200	10.0.26100	10.0.26100 Build 26100	6.14.0-33-generic	10.0.26100 Build 26100.4061	10.0.26100 Build 26100.4061
BIOS Vendor	Intel®	AsusTek*	AsusTek*	AsusTek*	Intel®	AMI*	AMI*
BIOS Version	SNADL357.00 56.2022.1102.1 218	RVMTL357.00 46.2024.1122.11 09	LNLNLLNK.00 06.2024.1210.1 510	CRARLV57.00 24.2025.0507.1 536	SBRPL579.00 47.2022.1006.1 728	5.0.1	5.0.1
GPU Driver	25.05.32567.17	32.0.101.8247	32.0.101.8132	-	25.35.35096.9	-	32.0.101.8136
NPU Driver	N/A	32.0.100.4297	32.0.100.4297	-	N/A	N/A	N/A
OpenVINO	2025.4	2025.4	2025.4	2025.4	2025.4	2025.4	2025.4

§

3.0 System Setup

This section will guide you to set up your system with the Linux* or Windows* OS as including AI tools like Geti™ for model training and OpenVINO™ toolkit for model inferencing. Subsection 3.1 and 3.2 show the steps to set up the OpenVINO™ toolkit on bare metal and containerization, followed by subsection 3.3 for Geti™ on-premises setup.

Only subsection 3.1 contains the steps for both Linux* and Windows* OS. Subsection 3.2 as well as 3.3 only have Linux* steps. The Windows* setup can be done using [Windows* Subsystem for Linux*](#) (WSL). For WSL, please install WSL2 and follow the steps for Linux* OS.

3.1 OpenVINO™ Toolkit Bare Metal Setup with Virtual Environment

3.1.1 On Linux* OS

3.1.1.1 Install Operating System and Required Packages

Download Ubuntu* Desktop 24.04 LTS and perform a normal (not minimal) installation. Afterwards, install Python* virtual environment and development packages using Cmd 1.

Cmd 1. Commands to Install Required Packages

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install \
    git, curl, wget, gpg-agent, libtbb12, quilt, \
    libssl-dev, kernel-wedge, liblz4-tool, libelf-dev, \
    flex, bison, python3-pip, python3-venv, ca-certificates, \
    vim
```

3.1.1.2 Upgrade to the Latest Intel® GPU and NPU Drivers

Please use the drivers that were provided and evaluated by the OS vendors. Upgrading to the latest Intel® drivers is optional. Please read the terms and conditions when downloading the drivers.

You can get the latest Intel® GPU and NPU drivers, including the steps to install them, from the following links.

- GPU Driver: [Intel® Compute Runtime Releases](#)

- NPU Driver: [Intel® Linux* NPU Driver Releases](#)

Cmd 2 and Cmd 3 show the steps to install the GPU and NPU drivers, respectively.

Cmd 2. Commands to Install GPU Driver

```
$ wget -q0 - https://repositories.intel.com/gpu/intel-graphics.key | \
  sudo gpg --yes --dearmor --output /usr/share/keyrings/intel-
graphics.gpg
$ echo "deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-
graphics.gpg] https://repositories.intel.com/gpu/ubuntu noble client" | \
  sudo tee /etc/apt/sources.list.d/intel-gpu-noble.list
$ sudo apt update
$ sudo apt install \
  libze1, intel-level-zero-gpu, intel-opencl-icd, \
  intel-gsc, clinfo, vainfo, hwinfo, linux-firmware

$ mkdir -p /tmp/neo_gpu
$ cd /tmp/neo_gpu
$ wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-core-
2_2.5.6+18417_amd64.deb
$ wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-opencl-
2_2.5.6+18417_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-gpu-
dbgsym_1.6.32224.5_amd64.ddeb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-
gpu_1.6.32224.5_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-opencl-icd-
dbgsym_24.52.32224.5_amd64.ddeb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-opencl-
icd_24.52.32224.5_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/libigdgmm12_22.5.5_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/ww52.sum
$ sha256sum -c ww52.sum
$ sudo dpkg -i ./*.deb
```

Cmd 3. Commands to Install NPU Driver

```
$ sudo apt update && sudo apt install dkms
$ sudo dpkg --purge --force-remove-reinstreq intel-driver-compiler-npu
intel-fw-npu intel-level-zero-npu intel-level-zero-npu-dbgsym
```

```
$ wget https://github.com/intel/linux-npu-driver/releases/download/v1.28.0/linux-npu-driver-v1.28.0.20251218-20347000698-ubuntu2404.tar.gz
$ tar -xf linux-npu-driver-v1.28.0.20251218-20347000698-ubuntu2404.tar.gz

$ sudo apt update
$ sudo apt install libtbb12
$ sudo dpkg -i *.deb

$ wget https://github.com/oneapi-src/level-zero/releases/download/v1.24.2/level-zero_1.24.2+u24.04_amd64.deb
$ sudo dpkg -i level-zero*.deb
```

3.1.1.3 Install Intel® GPU Packages

The [Intel® GPU documentation](#) provides the installation steps for both data center GPUs and client GPUs. Cmd 4 shows the steps to install client GPU packages. Please make sure that the user is either root or in the render group to have access privileges to both GPU and NPU. Then, reboot your system.

Cmd 4. Commands to Install Client GPU Packages

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository ppa:kobuk-team/intel-graphics

$ sudo apt update
$ sudo apt install libze-intel-gpu1 libze1 \
  intel-metrics-discovery intel-opencl-icd clinfo intel-gsc
$ sudo apt install intel-media-va-driver-non-free \
  libmfx-gen1 libvpl2 libvpl-tools libva-glX2 \
  va-driver-all vainfo
$ sudo apt install -y libze-dev intel-ocloc

$ sudo gpasswd -a ${USER} render
$ newgrp render

$ sudo reboot now
```

To check if the GPU and NPU setups are done properly, please run Cmd 5, Cmd 6, Cmd 7, and Cmd 8. The `clinfo` command should display the number of GPUs on your system.

Cmd 5. Commands to Check GPU Driver

```
$ ls -l /dev/dri/
total 0
drwxr-xr-x  2 root root          80 Dec 18 17:46 by-path
```

```
crw-rw----+ 1 root video  226,   0 Dec 18 17:46 card0
crw-rw----+ 1 root render 226, 128 Dec 18 17:46 renderD128
```

Cmd 6. Commands to Check NPU Driver

```
$ ls -l /dev/accel
total 0
crw-rw---- 1 root render 261, 0 Oct  1 12:02 accel0
```

Cmd 7. Commands to Check User Privilege

```
$ groups ${USER}
user : ... render ...
```

Cmd 8. Commands to Show Detected GPUs

```
$ clinfo
Number of platforms      1
Platform Name            Intel(R) OpenCL Graphics
Platform Vendor          Intel(R) Corporation
...
```

3.1.1.4 Install OpenVINO™ Toolkit

Cmd 9 shows the steps to install the OpenVINO™ toolkit. Please refer to the [OpenVINO™ website](#) for more details on the requirements. Cmd 10 shows the available compute units that can be accessed by the OpenVINO™ toolkit.

Cmd 9. Commands to Install OpenVINO™

```
$ sudo apt install python3.12-venv
$ python3 -m venv openvino_env
$ source openvino_env/bin/activate
$ python -m pip install --upgrade pip
$ pip install openvino==2025.4.1
```

Cmd 10. Commands to Check Available Compute Units

```
$ benchmark_app --help
...
Available target devices:  CPU  GPU  NPU
```

3.1.1.5 Run the OpenVINO™ Toolkit Benchmark Application

Cmd 11 shows the steps to convert Ultralytics* YOLO model into the OpenVINO™ toolkit Intermediate Representation (IR) model with static input size and FP16 quantization. Please refer to the [Ultralytics* OpenVINO™ documentation](#) for more details.

Cmd 11. Commands to Convert Ultralytics* YOLO Model into OpenVINO™ IR Model

```
$ pip install ultralytics==8.3.142
$ vim get_yolo_openvino_model.py
from pathlib import Path
from ultralytics import YOLO
det_model = YOLO(f"yolo11s.pt")
det_model.export(format="openvino", dynamic=False, half=True)

$ python get_yolo_openvino_model.py
$ ls -l yolo11s_openvino_model/
total 18904
-rw-rw-r-- 1 user user      1525 Dec 18 17:58 metadata.yaml
-rw-rw-r-- 1 user user 18938232 Dec 18 17:58 yolo11s.bin
-rw-rw-r-- 1 user user   413491 Dec 18 17:58 yolo11s.xml
```

Cmd 12 shows the steps to run performance benchmark on CPU, GPU, and NPU using OpenVINO™ `benchmark_app`. The result includes both model latency in milliseconds and model throughput in frames per second. Please refer to [OpenVINO™ documentation](#) for the command manual.

Cmd 12. Commands to Perform Benchmark using Different Compute Units

```
$ benchmark_app -m yolo11s_openvino_model/yolo11s.xml \
  -shape [1,3,640,640] \
  -hint latency \
  -d CPU
$ benchmark_app -m yolo11s_openvino_model/yolo11s.xml \
  -shape [1,3,640,640] \
  -hint latency \
  -d GPU
$ benchmark_app -m yolo11s_openvino_model/yolo11s.xml \
  -shape [1,3,640,640] \
  -hint latency \
  -d NPU
```

3.1.1.6 Run the OpenVINO™ Toolkit GenAI Library

[OpenVINO™ GenAI](#) is a library to run generative AI pipeline using the OpenVINO™ toolkit as the backend. Cmd 13 shows the steps to install the OpenVINO™ toolkit GenAI library. Cmd 14 shows the steps to download the [OpenVINO™ IR models](#) from the Hugging Face* website. Lastly, Cmd 15 shows the steps to run the LLM pipeline, generating the query and benchmark results.

Cmd 13. Commands to Install the OpenVINO™ Toolkit GenAI Library

```
$ git clone https://github.com/openvinotoolkit/openvino.genai.git
$ cd openvino.genai
$ git checkout 2025.4.1.0
$ cd tools/llm_bench
```

```
$ pip install -r requirements.txt
```

Cmd 14. Commands to Download the OpenVINO™ Toolkit IR Models from Hugging Face* Website

```
$ hf download --repo-type model --local-dir "models/mistral-7b-instruct-v0.3-int4" "OpenVINO/Mistral-7B-Instruct-v0.3-int4-cw-ov"
$ hf download --repo-type model --local-dir "models/qwen3-8b-int4" "OpenVINO/Qwen3-8B-int4-cw-ov"
$ hf download --repo-type model --local-dir "models/phi-3-medium-4k-instruct-int4" "OpenVINO/Phi-3-medium-4k-instruct-int4-ov"
```

Cmd 15. Commands to Benchmark LLM Models

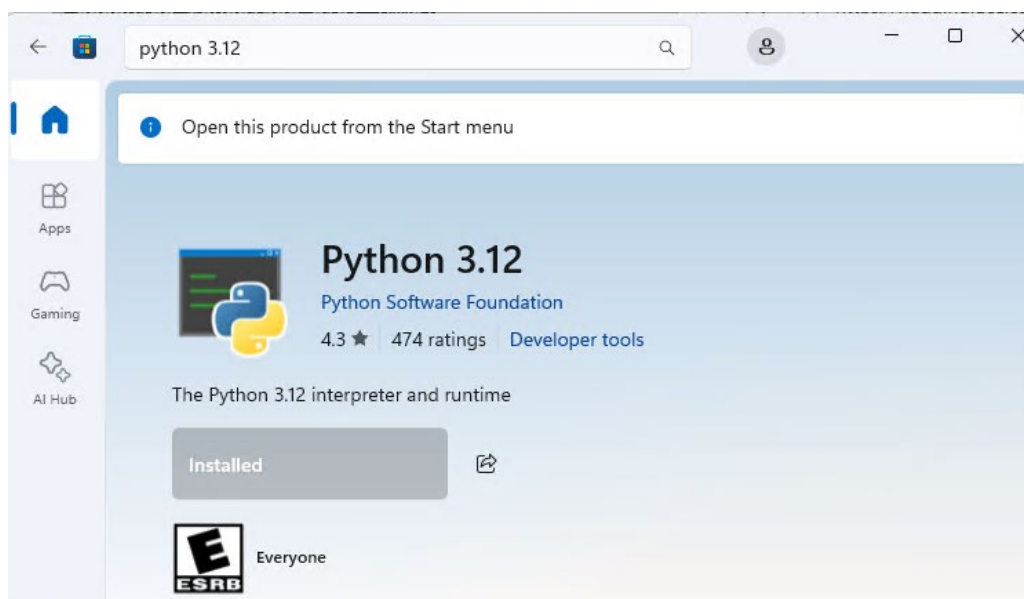
```
$ python benchmark.py \
-m "models/mistral-7b-instruct-v0.3-int4" \
-p "What is openvino?" \
-n 2 \
-d GPU
$ python benchmark.py \
-m "models/qwen3-8b-int4" \
-p "What is openvino?" \
-n 2 \
-d GPU
$ python benchmark.py \
-m "models/phi-3-medium-4k-instruct-int4" \
-p "What is openvino?" \
-n 2 \
-d GPU
```

3.1.2 Installation Steps for Windows* OS

3.1.2.1 Install Operating System and Required Packages

Install Microsoft Windows* 11 and the requirements for OpenVINO™ like [Microsoft Visual Studio*](#). Other requirements like Python* can be downloaded from Microsoft Store* as shown in Figure 1. Please refer to [OpenVINO™ website](#) for more details.

Figure 1. Python* Download Page on Microsoft Store*



3.1.2.2 Upgrade to the Latest Intel® GPU and NPU Drivers

Please use the drivers that were provided and evaluated by the OS vendors. Upgrading to the latest Intel® drivers is optional. Please read the terms and conditions when downloading the drivers.

Please find the following links to download and install the latest Intel® GPU and NPU drivers.

- GPU Driver: [Intel® Arc™ Graphics - Windows*](#) or [Intel® Arc™ Pro Graphics - Windows*](#)
- NPU Driver: [Intel® NPU Driver - Windows*](#)

3.1.2.3 Install OpenVINO™ Toolkit

Open Command Prompt and run Cmd 16 to install OpenVINO™. Use Cmd 17 to check all compute units accessible by the OpenVINO™ toolkit.

Cmd 16. Commands to Install OpenVINO™

```
> python3.12 -m venv openvino_env
> openvino_env\Scripts\activate
> python -m pip install --upgrade pip
> pip install openvino==2025.4.1
```

Cmd 17. Commands to Check Available Compute Units

```
> benchmark_app -help
...
Available target devices:  CPU  GPU  NPU
```

3.1.2.4 Run OpenVINO™ Benchmark Application

Create a Python* file based on Cmd 18 to convert Ultralytics* YOLO model into the OpenVINO™ toolkit IR model with a static input size and FP16 quantization. Then, run Cmd 19. Please refer to the [Ultralytics* OpenVINO™ documentation](#) for more details.

Cmd 18. Content of get_yolo_openvino_model.py File

```
from pathlib import Path
from ultralytics import YOLO
det_model = YOLO(f"yolo11s.pt")
det_model.export(format="openvino", dynamic=False, half=True)
```

Cmd 19. Commands to Convert Ultralytics* YOLO Model into OpenVINO™ IR Model

```
> pip install ultralytics==8.3.142
> python get_yolo_openvino_model.py
```

Use Cmd 20 to perform model inference benchmark on CPU, GPU, and NPU using the OpenVINO™ toolkit benchmark_app. The result includes both model latency in milliseconds and model throughput in frames per second. Please refer to the [OpenVINO™ documentation](#) for the command manual.

Cmd 20. Commands to Perform Benchmark using Different Compute Units

```
> benchmark_app -m yolo11s_openvino_model/yolo11s.xml -shape
[1,3,640,640] -hint latency -d CPU
> benchmark_app -m yolo11s_openvino_model/yolo11s.xml -shape
[1,3,640,640] -hint latency -d GPU
> benchmark_app -m yolo11s_openvino_model/yolo11s.xml -shape
[1,3,640,640] -hint latency -d NPU
```

3.1.2.5 Run OpenVINO™ Toolkit GenAI Library

[OpenVINO™ GenAI](#) is a generative AI pipeline library that uses OpenVINO™ as its backend. Cmd 21 shows the steps to install OpenVINO™ GenAI library. Cmd 22 shows the steps to download [OpenVINO™ IR models](#) from Hugging Face* website. And finally, Cmd 23 shows the steps to run LLM pipeline producing generated query and benchmark results.

Cmd 21. Commands to Install OpenVINO™ GenAI Library

```
> git clone https://github.com/openvinotoolkit/openvino.genai.git
> cd openvino.genai
> git checkout 2025.4.1.0
```

```
> cd tools/llm_bench
> pip install -r requirements.txt
```

Cmd 22. Commands to Download the OpenVINO™ Toolkit IR Models from the Hugging Face* Website

```
> hf download --repo-type model --local-dir "models/mistral-7b-instruct-v0.3-int4" "OpenVINO/Mistral-7B-Instruct-v0.3-int4-cw-ov"
> hf download --repo-type model --local-dir "models/qwen3-8b-int4" "OpenVINO/Qwen3-8B-int4-cw-ov"
> hf download --repo-type model --local-dir "models/phi-3-medium-4k-instruct-int4" "OpenVINO/Phi-3-medium-4k-instruct-int4-ov"
```

Cmd 23. Commands to Benchmark LLM Models

```
> python benchmark.py -m "models/mistral-7b-instruct-v0.3-int4" -p "What is openvino?" -n 2 -d GPU
> python benchmark.py -m "models/qwen3-8b-int4" -p "What is openvino?" -n 2 -d GPU
> python benchmark.py -m "models/phi-3-medium-4k-instruct-int4" -p "What is openvino?" -n 2 -d GPU
```

3.2 OpenVINO™ Toolkit Setup with Containerization

The following steps only apply to Linux* OS. For Windows* OS, install WSL and then follow these steps.

3.2.1 Install Operating System, Drivers, and Required Packages

Please refer to section 3.1.1 to install Linux* OS and the required packages, as well as the optional steps to upgrade the Intel® NPU and GPU drivers.

3.2.2 Install Docker*

Cmd 24 shows the steps to install Docker*.

Cmd 24. Commands to Install Docker*

```
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc

$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

$ sudo apt update
$ sudo apt install \
  docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin
```

For a secure system behind a firewall, follow Cmd 25 to run Docker* with proxy settings.

Cmd 25. Commands to Run Docker* with Proxy Setting

```
$ sudo mkdir -p /etc/systemd/system/docker.service.d

$ vim /etc/systemd/system/docker.service.d/http-proxy.conf
[Service]
Environment=HTTP_PROXY=$HTTP_PROXY
Environment=HTTPS_PROXY=$HTTPS_PROXY
Environment=FTP_PROXY=$FTP_PROXY
Environment=SOCKS_PROXY=$SOCKS_PROXY
Environment=NO_PROXY=$NO_PROXY

$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

3.2.3 Build OpenVINO™ xPU Docker* Image

OpenVINO™ xPU Docker* image adds the ability to access multiple compute units, like GPU and NPU, on top of the OpenVINO™ Toolkit Docker* image that can only access CPU. Note that upgrading the GPU and NPU drivers is optional. Please read the terms and conditions to use the new drivers.

Cmd 26 contains the Dockerfile, Cmd 27 contains the script to include the NPU driver, and Cmd 28 shows the steps to build the xPU Docker* image. Cmd 29 shows the steps to run the xPU Docker* image and to detect the available compute units. The result shows that all CPU, iGPU, and NPU are detected. If there are both iGPU and dGPU in the system, iGPU will be detected as GPU.0, whereas the dGPU will be detected as GPU.1.

Cmd 26. Content of OpenVINO™ xPU Dockerfile

```
$ vim Dockerfile_xpu
FROM openvino/ubuntu24_dev:2025.0.0

USER root
RUN apt-get update && \
  apt-get install -y \
    wget gnupg curl libtbbmalloc2 libtbb12 kmod ffmpeg
```

```
# Install GPU Driver
RUN wget -qO - https://repositories.intel.com/gpu/intel-graphics.key
| \
    gpg --yes --dearmor --output /usr/share/keyrings/intel-
graphics.gpg && \
    echo "deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-
graphics.gpg] https://repositories.intel.com/gpu/ubuntu noble client" |
\
    tee /etc/apt/sources.list.d/intel-gpu-noble.list
RUN apt update && \
    apt-get install -y \
        libze1 intel-opencl-icd clinfo intel-gsc clinfo \
        vainfo hwinfo

RUN mkdir -p /tmp/neo_gpu
WORKDIR /tmp/neo_gpu
RUN wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-core-
2_2.5.6+18417_amd64.deb && \
    wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-opencl-
2_2.5.6+18417_amd64.deb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-gpu-
dbgsym_1.6.32224.5_amd64.ddeb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-
gpu_1.6.32224.5_amd64.deb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-opencl-icd-
dbgsym_24.52.32224.5_amd64.ddeb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-opencl-
icd_24.52.32224.5_amd64.deb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/libigdgmm12_22.5.5_amd64.deb
RUN dpkg -i ./*.deb

# Install NPU Driver
COPY npu_container.sh /tmp/neo_npu/npu_container.sh
WORKDIR /tmp/neo_npu/
RUN bash npu_container.sh
# Finalizing
USER openvino
WORKDIR /home/openvino
```

Cmd 27. Content of NPU Script File

```
$ vim npu_container.sh
  wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-driver-compiler-
npu_1.13.0.20250131-13074932693_ubuntu24.04_amd64.deb
  wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-fw-npu_1.13.0.20250131-
13074932693_ubuntu24.04_amd64.deb
  wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-level-zero-npu_1.13.0.20250131-
13074932693_ubuntu24.04_amd64.deb
  wget https://github.com/oneapi-src/level-
zero/releases/download/v1.18.5/level-zero_1.18.5+u24.04_amd64.deb
  dpkg -i ./*.deb
```

Cmd 28. Commands to Build OpenVINO™ xPU Docker* Image

```
$ sudo docker build --progress=plain --no-cache \
--build-arg HTTP_PROXY="$HTTP_PROXY" \
--build-arg HTTPS_PROXY="$HTTPS_PROXY" \
--build-arg FTP_PROXY="$FTP_PROXY" \
--build-arg SOCKS_PROXY="$SOCKS_PROXY" \
--build-arg NO_PROXY="$NO_PROXY" \
-t openvino/ubuntu24_dev_xpu:2025.0.0 -f Dockerfile_xpu .
```

Cmd 29. Commands to Run OpenVINO™ xPU Docker* Image

```
$ sudo docker run -it \
--device=/dev/dri/renderD128 \
--device=/dev/accel/accel0 \
--user root \
-e HTTP_PROXY="$HTTP_PROXY" \
-e HTTPS_PROXY="$HTTPS_PROXY" \
-e FTP_PROXY="$FTP_PROXY" \
-e SOCKS_PROXY="$SOCKS_PROXY" \
-e NO_PROXY="$NO_PROXY" \
--rm openvino/ubuntu24_dev_xpu:2025.0.0 \
"/opt/intel/openvino/samples/cpp/samples_bin/samples_bin/benchmark_app"
"--help"
```

...

Available target devices: CPU GPU NPU

To secure the system behind a firewall, the Docker* container requires proxy configuration parameters to access the Internet. Removing the parameters is optional if the container does not need to access the internet or if the system does not have proxy settings.

3.2.4 Build OpenVINO™ Notebooks Docker* Image

OpenVINO™ Notebooks provides a collection of Jupyter* notebooks for running OpenVINO™ toolkit applications for learning and experimentation. Cmd 30 contains OpenVINO™ Notebooks Dockerfile and Cmd 31 shows the steps to build the image.

Cmd 30. Content of OpenVINO™ Notebooks Dockerfile

```
$ vim Dockerfile_xpu_nb
FROM openvino/ubuntu24_dev_xpu:2025.0.0

USER openvino
WORKDIR /home/openvino
RUN git clone --depth=1 -b 2025.0
https://github.com/openvinotoolkit/openvino_notebooks.git

WORKDIR /home/openvino/openvino_notebooks
USER root
RUN python3 -m pip install --upgrade pip
RUN pip3 install wheel setuptools
RUN pip3 install -r requirements.txt

# Finalizing
USER openvino
WORKDIR /home/openvino
```

Cmd 31. Commands to Build OpenVINO™ Notebooks Docker* Image

```
$ sudo docker build --progress=plain --no-cache \
--build-arg HTTP_PROXY="$HTTP_PROXY" \
--build-arg HTTPS_PROXY="$HTTPS_PROXY" \
--build-arg FTP_PROXY="$FTP_PROXY" \
--build-arg SOCKS_PROXY="$SOCKS_PROXY" \
--build-arg NO_PROXY="$NO_PROXY" \
-t openvino/ubuntu24_dev_xpu_nb:2025.0.0 -f Dockerfile_xpu_nb .
```

```
$ sudo docker images
```

REPOSITORY	TAG	...	SIZE
openvino/ubuntu24_dev_xpu_nb	2025.0.0	...	4.21GB
openvino/ubuntu24_dev_xpu	2025.0.0	...	3.37GB

Cmd 32 shows the steps to run OpenVINO™ Notebooks Docker* image. To access the notebooks Docker* container, please use a web browser and provide the IP address, the port, as well as the token. Then, navigate to `openvino_notebooks/notebooks` directory to run the notebooks.

Docker* container is stateless. After the container is killed, all newly created or modified files stored in it will be lost. They will be reverted to the original image. Cmd 32 uses a local results directory as the location to permanently store the files.

Cmd 32. Commands to Run OpenVINO™ Notebooks Docker* Image

```
$ mkdir results

$ sudo docker run -it \
  -p 8888:8888 \
  -v ./results:/opt/intel/openvino/results \
  --device=/dev/dri/renderD128 \
  --device=/dev/accel/accel0 \
  --user root \
  -e HTTP_PROXY="$HTTP_PROXY" \
  -e HTTPS_PROXY="$HTTPS_PROXY" \
  -e FTP_PROXY="$FTP_PROXY" \
  -e SOCKS_PROXY="$SOCKS_PROXY" \
  -e NO_PROXY="$NO_PROXY" \
  --rm openvino/ubuntu24_dev_xpu_nb:2025.0.0 \
  bash -c "jupyter lab --no-browser --allow-root --ip 0.0.0.0 --
port=8888"
```

3.3 Geti™ Application Setup on Premise

Geti™ application is a computer vision model training platform with dataset annotation features. The trained model can be optimized and exported as OpenVINO® IR model. The system for the following setup steps has two Intel® Arc™ B60 dGPUs. We will install it with Geti™ 2.12.1. Please refer to the [Geti™ installation and requirements documentation](#) for more information.

3.3.1 Install Operating System and Upgrade Linux* Kernel

Download and Install Ubuntu* Desktop 24.04 LTS. Since Intel® Arc™ B60 Graphics requires Linux* Kernel 6.14 and onward. Please follow Cmd 33 to upgrade the kernel. Please find [Intel® GPU Hardware Table](#) for the required Kernel version for each SKU. Skip this if the Kernel version has been satisfied.

Cmd 33. Commands to Upgrade Linux* Kernel

```
$ uname -a
... 6.11.0-29-generic ...

$ sudo add-apt-repository ppa:canonical-kernel-team/ppa
$ sudo apt update
$ sudo apt upgrade
...
The following NEW packages will be installed:
  linux-headers-6.14.0-24-generic linux-hwe-6.14-headers-6.14.0-24
linux-hwe-6.14-tools-6.14.0-24
  linux-image-6.14.0-24-generic linux-modules-6.14.0-24-generic linux-
modules-extra-6.14.0-24-generic
```

```
linux-tools-6.14.0-24-generic
...

$ sudo reboot now
$ uname -a
... 6.14.0-24-generic ...
```

3.3.2 Install Intel® GPU Packages

Cmd 34 shows the steps to install Intel® GPU Packages. Please add the user to `render` the group to be able to access the GPUs.

Cmd 34. Commands to Install Intel® GPU Packages

```
$ sudo apt update
$ sudo apt install -y software-properties-common
$ vim intel-gpu-noble.list
deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-graphics.gpg]
https://repositories.intel.com/gpu/ubuntu noble client
$ sudo apt update
$ sudo apt upgrade

$ sudo apt install -y intel-media-va-driver-non-free \
  libmfx-gen1 libvpl2 libvpl-tools libva-glx2 va-driver-all \
  vainfo
$ sudo apt install -y libze-dev intel-ocloc
$ sudo apt install intel-gpu-tools
$ sudo apt install -y \
  intel-onecl-icd intel-level-zero-gpu level-zero \
  intel-media-va-driver-non-free libmfxgen1 libvpl2 \
  libegl-mesa0 libegl1-mesa libegl1-mesa-dev libgbm1 \
  libgl1-mesa-dev libgl1-mesa-dri libglapi-mesa \
  libgles2-mesa-dev libglx-mesa0 libigdgmm12 libxatracker2 \
  mesa-va-drivers mesa-vidpau-drivers mesa-vulkan-drivers \
  va-driver-all vainfo hwinfo clinfo
$ sudo apt install libglu1-mesa-dev
$ sudo apt install vdpau-driver-all

$ sudo gpasswd -a ${USER} render
$ newgrp render
```

Since the Battlemage Pro is a new product, some packages are needed from the staging repository. This part is optional for mature products. Please follow Cmd 35 to upgrade the packages from the Intel® GPU staging repository.

Cmd 35. Commands to Install Intel® GPU Staging Packages

```
$ sudo add-apt-repository ppa:kobuk-team/intel-graphics-staging
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install \
  intel-gsc libigdgmm12 libigc2 intel-level-zero-gpu-raytracing \
  intel-media-va-driver-non-free libze1 vainfo libvpl2 \
  libvpl-tools libmetee4 intel-metrics-discovery \
  intel-metrics-library libmfx-gen1 libtbb12 libtbbmalloc2 \
  intel-opencl-icd libze-intel-gpu1 linux-intel clinfo
```

Cmd 36 shows the steps to install the Intel® GPU Profiling Tool. After completing the package installation, reboot the system.

Cmd 36. Commands to Install Intel® GPU Profiling Tool

```
$ wget
https://github.com/intel/xpumanager/releases/download/V1.3.0/xpu-
smi_1.3.0_20250707.103634.3db7de07.u24.04_amd64.deb
$ sudo dpkg -i xpu-smi_1.3.0_20250707.103634.3db7de07.u24.04_amd64.deb

$ sudo reboot now
```

Cmd 37 shows the steps to check the GPU information. The output shows three GPUs (one iGPU and two dGPUs). Please note that the `clinfo` command only reports two platforms (iGPU and B60 dGPU). Later, it will show that the B60 dGPU platform has two devices.

Cmd 37. Commands to Check Intel® GPU Information

```
$ clinfo
Number of platforms          2
...

$ sudo xpu-smi discovery
+-----+-----+
| Device ID | Device Information
+-----+-----+
| 0         | Device Name: Intel(R) Graphics ...
|           | DRM Device: /dev/dri/card1
|           | Function Type: physical
+-----+-----+
| 1         | Device Name: Intel(R) Graphics ...
|           | DRM Device: /dev/dri/card3
|           | Function Type: physical
+-----+-----+
| 2         | Device Name: Intel(R) Graphics ...
|           | DRM Device: /dev/dri/card2
```

```
|           | Function Type: physical
+-----+-----+

$ sudo xpu-smi stats -d 0
+-----+-----+
| Device ID           | 0
+-----+-----+
...
+-----+-----+
| GPU Power (W)       | 34
| GPU Frequency (MHz) | 400
| Media Engine Freq (MHz) | N/A
| GPU Core Temperature (C) | 46
| GPU Memory Temperature (C) | 46
| GPU Memory Read (kB/s) | 48288
| GPU Memory Write (kB/s) | 991830
| GPU Memory Bandwidth (%) | 0
| GPU Memory Used (MiB) | 212
| GPU Memory Util (%) | 1
| Xe Link Throughput (kB/s) | N/A
+-----+-----+
```

3.3.3 Install Geti™ Application

Cmd 38 shows the steps to install Geti application™. Please fill in the required information, including the admin username and password, the mail server, the certificate, the dataset location (in the /data directory), and the monitoring tool. After installation, the Geti™ application can be accessed via a web browser using the system IP.

Cmd 38. Commands to Install Geti™ Application

```
$ curl -LO
https://storage.geti.intel.com/v2.12.1/platform_installer.tar.gz
$ curl -LO
https://storage.geti.intel.com/v2.12.1/platform_installer.tar.gz.sha256
$ echo "$(cat platform_installer.tar.gz.sha256)
platform_installer.tar.gz" | sha256sum --check
installer: Ok
$ tar -xf platform_installer.tar.gz

$ sudo mkdir /data
$ sudo chmod 750 /data

$ cd platform_2.12.1
$ sudo ./platform_installer install
```

4.0 System Profiling

System profiling is useful for analyzing the resource consumption of an individual program or the overall system utilization. This section focuses on computing, memory, and power resources. The analysis results can then be used to determine which processor SKU is best for the application, the required memory capacity, and the chassis design for a specific deployment environment, given power-to-temperature implications. In the case of battery-powered deployment, it can be used to predict recharge cycle time.

For Linux* environment, the default `top` command can be used to monitor CPU utilization. Whereas Intel® GPU Top is for iGPU usage. The `top` command is also useful for monitoring CPU and memory usage, and Intel® XPU-SMI for Intel® dGPUs. Finally, for CPU and iGPU power consumption, you can use Intel® PCM. Please use Intel® XPU-SMI for dGPU power consumption.

For Windows*, Task Manager provides CPU, iGPU, dGPU, NPU, and memory utilizations. For power consumption, please use third-party applications.

4.1 `top`

The [top command](#) is useful for checking the system RAM and Swap capacities, as well as the top running programs and their CPU and memory utilization. Cmd 39 shows 64 GB total RAM and 8 GB of total Swap file. It shows two Python* programs running on CPU, utilizing 15 CPU threads and 1 CPU thread. It also shows 4 OpenVINO™ benchmark_app programs using less than 1% CPU thread each, as the workloads run on the NPU.

Cmd 39. Commands to Run `top`

```
$ top
top-20:03:48 up 4 days,3:37,8 users,load average:4.68,3.65, 3.11
Tasks: 422 total,2 running,419 sleeping,1 stopped,0 zombie
%Cpu(s):99.4 us,0.4 sy,0.0 ni,0.2 id,0.0 wa,0.0 hi,0.0 si,0.0 st
MiB Mem:63859.1 total,24491.3 free,14319.7 used,26642.5 buff/cache
MiB Swap:8192.0 total,8116.2 free,75.8 used. 49539.4 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM ...COMMAND
... user 20 0 3512488 652808 281000 R 1493 1.0 ...python
... user 20 0 7779680 1.0g 268536 S 100 1.7 ...python
... user 20 0 2580904 468252 376808 S 0.7 0.7 ...benchmark_app
... user 20 0 9948 3608 3340 S 0.3 0.0 ...bash
... user 20 0 2580904 468456 377020 S 0.3 0.7 ...benchmark_app
... user 20 0 2580904 468340 376904 S 0.3 0.7 ...benchmark_app
... user 20 0 2580904 468496 377056 S 0.3 0.7 ...benchmark_app
...
```

Intel® GPU Top

The `intel_gpu_top` command is useful to display the top running programs utilizing Intel® iGPU. Cmd 40 shows the steps to install `intel_gpu_top`. Cmd 41 shows one Python* program that utilizes 81% of the processor iGPU.

Cmd 40. Commands to Install Intel® GPU Top

```
$ sudo apt install intel-gpu-tools
```

Cmd 41. Commands to Run Intel® GPU Top

```
$ sudo intel_gpu_top
intel-gpu-top:8086:7d51 @ /dev/dri/card0-1084/1173 MHz; 52% RC6;
10.77/27.92 W; 723 irqs/s
```

ENGINES		BUSY		MI_SEMA		MI_WAIT	
	Render/3D	0.00%				0%	0%
	Blitter	0.00%				0%	0%
	Video	0.00%				0%	0%
VideoEnhance	0.00%					0%	0%
	Compute	80.83%	████████████████████			0%	0%
PID	Render/3D	Blitter	Video	VideoEnhance	Compute	NAME	
...					██████	python3	
...						xdg-deskto	
...						mutter-x11	
...						gnome-shel	
...						ibus-x11	
...						ibus-exten	
...						ibus-ui-gt	
...						Xorg	
...						ibus-x11	
...						mutter-x11	
...						Xwayland	
...						systemd	

Intel® Performance Counter Monitor

[Intel® Performance Counter Monitor \(Intel® PCM\)](#) is useful to monitor the power consumption of Intel® CPU and iGPU. Cmd 42 shows the steps to install and run Intel® PCM. If there is an error mentioning “unsupported processor,” please get the latest Intel® PCM from the repository and follow Cmd 43 to build the binary from the [source code](#).

Cmd 42. Commands to Install and Run Intel® PCM

```
$ sudo apt install pcm
$ sudo pcm -silent
```

Cmd 43. Commands to Build and Run the Latest Intel® PCM

```
$ unzip pcm-master.zip
$ cd pcm-master

$ mkdir build
$ cd build
$ cmake ..
$ cmake --build . --parallel --config Release

$ cd bin
$ sudo ./pcm -silent
```

Figure 2 shows the sample output after running the Intel® PCM. CPU energy (0.83 Joules) represents the total energy of the entire processor (including the CPU, GPU, NPU, and other I/Os such as memory controllers). PP0 energy (0.05 Joule) represents only the CPU cores, whereas PP1 energy (0.00 Joule) is for the iGPU. To get the Watts, divide the energy in Joules by the interval time collected in seconds. The default result is already in Watt as it is collected every second.

Figure 2. Intel® PCM Sample Output

[illegible]

4.4 Intel® XPU-SMI

[Intel® XPU System Management Interface \(SMI\)](#) is a command line (CLI) version to manage and monitor Intel® Data Center GPUs. It is useful to collect power consumption of Intel® dGPUs. Cmd 44 shows the steps to install Intel® XPU-SMI and Cmd 45 shows the steps to run it. You can also use the GPU Power to monitor the dGPU utilization.

Cmd 44. Commands to Install Intel® XPU-SMI

```
$ apt install xpu-smi
```

Cmd 45. Commands to Run Intel® XPU-SMI

```
$ sudo xpu-smi discovery
```

Device ID	Device Information
0	Device Name: Intel(R) Graphics ... DRM Device: /dev/dri/card1 Function Type: physical
1	Device Name: Intel(R) Graphics ... DRM Device: /dev/dri/card3 Function Type: physical
2	Device Name: Intel(R) Graphics ... DRM Device: /dev/dri/card2 Function Type: physical

```
$ sudo xpu-smi stats -d 0
```

Device ID	
0	
...	
GPU Power (W)	34
GPU Frequency (MHz)	400
Media Engine Freq (MHz)	N/A
GPU Core Temperature (C)	46
GPU Memory Temperature (C)	46
GPU Memory Read (kB/s)	48288
GPU Memory Write (kB/s)	991830
GPU Memory Bandwidth (%)	0
GPU Memory Used (MiB)	212
GPU Memory Util (%)	1



Xe Link Throughput (kB/s)	N/A
+-----+-----s	
§	

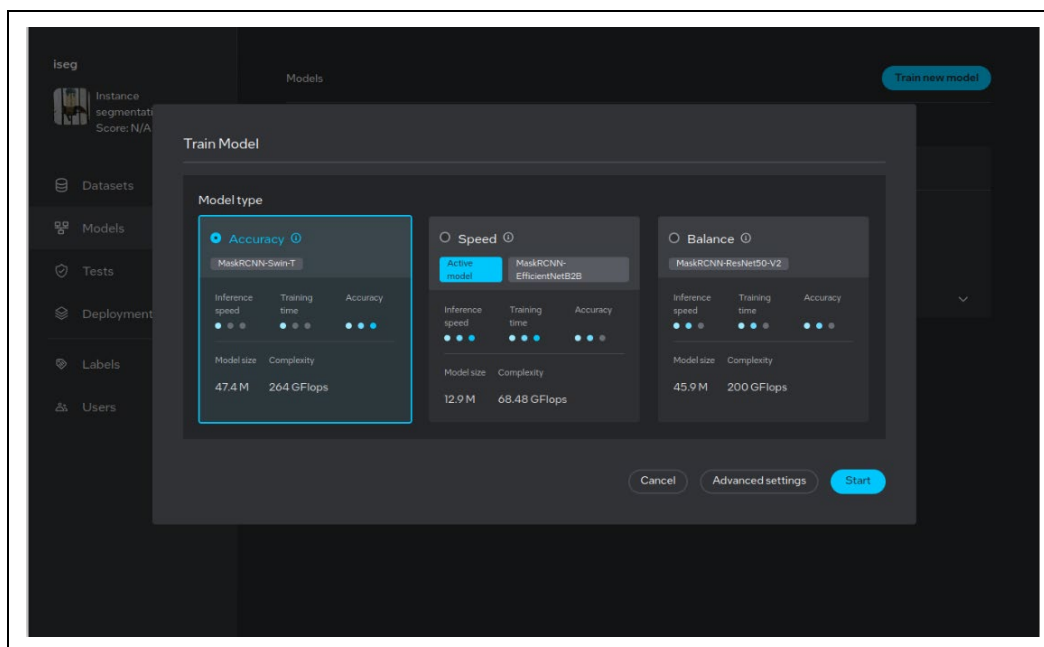
5.0 Accuracy Optimization

This section provides tips to tune the Geti™ application-trained model for better accuracy.

5.1 Geti™ Application Accuracy Model Selection

The Geti™ application supports multiple models for each task. One of the models is recommended for better accuracy as shown in Figure 3. You can retrain using the model if it is not the default model for the task. Other models are for speed as well as the balance between accuracy and speed.

Figure 3. Geti™ Application Model Selection



5.2 Geti™ Application Model Input Selection

Each model is suitable for a specific input size to achieve better accuracy, depending on its learning capacity. Geti™ application supports multiple methods to handle different input sizes. The tiling method as shown in Figure 4, is suitable to break down large images into smaller chunks where resizing the input images into smaller images is not possible as the objects to be detected are already too small. On the other hand, Figure 5 shows another method in which the original input is too small. In this case, it is 640x640 pixels. You can modify the input to be larger; for example, 1024x1024 pixels for better accuracy.

Figure 4. Geti™ Application Model Input Tiling

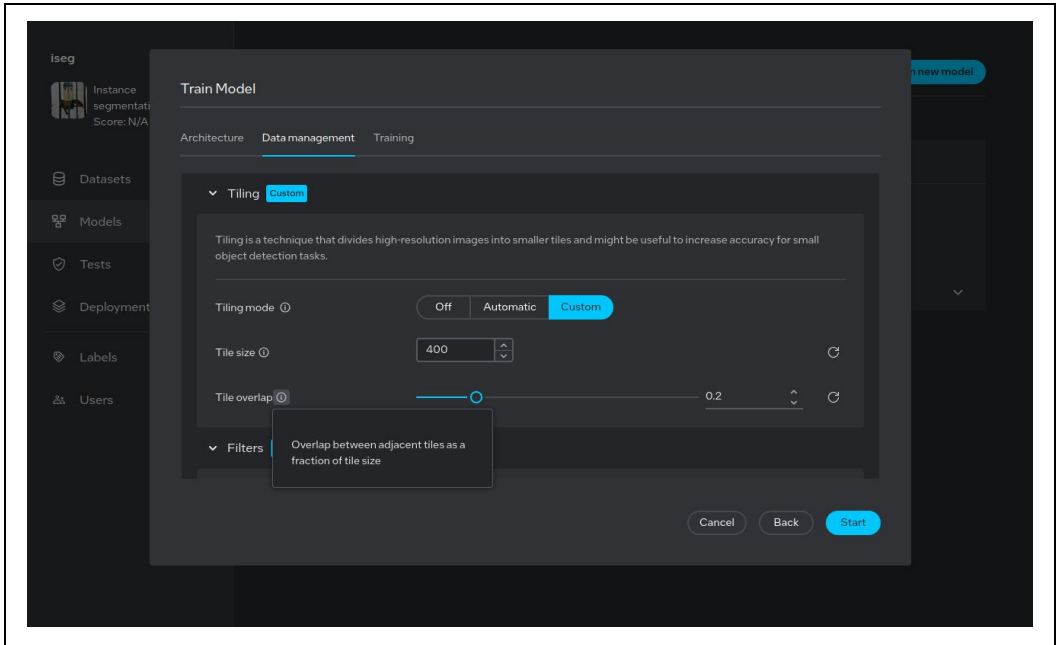
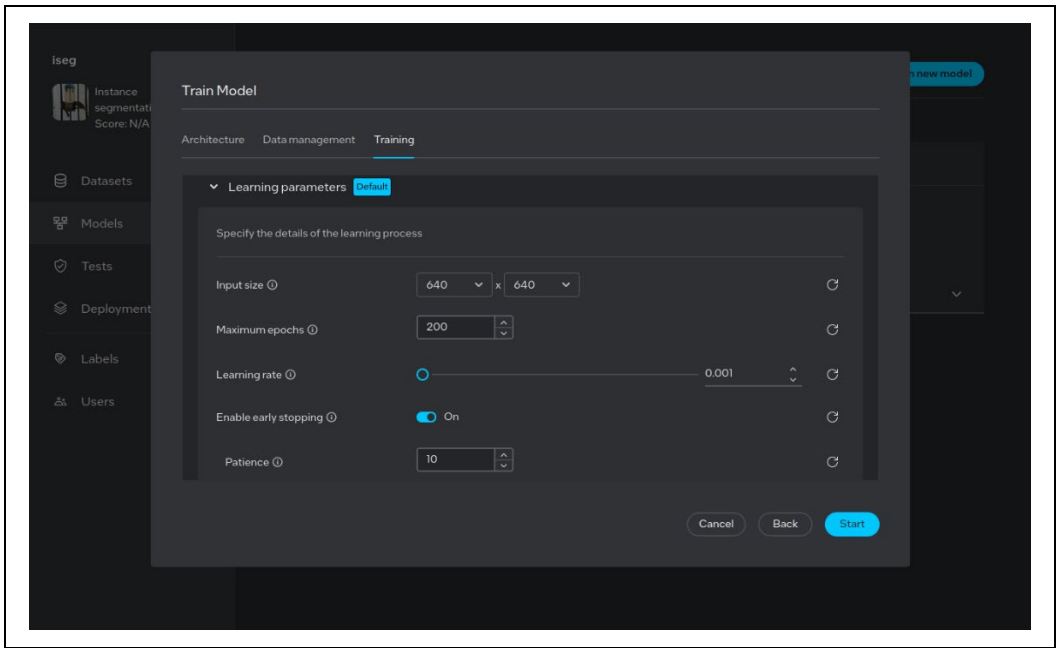


Figure 5. Geti™ Application Model Input Size



6.0 Performance Optimization

The motivation of OpenVINO™ toolkit is to accelerate AI inferencing. This section will primarily use OpenVINO™ tools to perform model conversion, compression, quantization, input channel swapping, input reshaping, input batching, input/output resizing, heterogeneous processing, asynchronous inference, and caching. Accelerated inferencing means responsive implementation, smaller memory footprint, and improved stability.

6.1 OpenVINO™ Toolkit Model Conversion

[Converting models](#) from TensorFlow*, PyTorch*, or other deep learning framework models to OpenVINO™ Intermediate Representation (IR) models has the benefit of running the models with the best performance on Intel® processors. This applies to all compute units (CPU, iGPU, NPU, and dGPU). After converting the model, it is recommended to run OpenVINO™ `benchmark_app` to obtain optimal model parameters. Cmd 46 shows the sample parameters.

Cmd 46. OpenVINO™ Toolkit `benchmark_app` Optimal Parameters

```
[Step 8/11] Querying optimal runtime parameters
[ INFO ] Model:
[ INFO ]   NETWORK_NAME: TensorFlow_Frontend_IR
[ INFO ]   OPTIMAL_NUMBER_OF_INFER_REQUESTS: 1
[ INFO ]   NUM_STREAMS: 1
[ INFO ]   INFERENCE_NUM_THREADS: 12
[ INFO ]   PERF_COUNT: NO
[ INFO ]   INFERENCE_PRECISION_HINT: <Type: 'float32'>
[ INFO ]   PERFORMANCE_HINT: LATENCY
[ INFO ]   EXECUTION_MODE_HINT: ExecutionMode.PERFORMANCE
[ INFO ]   PERFORMANCE_HINT_NUM_REQUESTS: 0
[ INFO ]   ENABLE_CPU_PINNING: False
[ INFO ]   ENABLE_CPU_RESERVATION: False
[ INFO ]   SCHEDULING_CORE_TYPE: SchedulingCoreType.ANY_CORE
[ INFO ]   MODEL_DISTRIBUTION_POLICY: set()
[ INFO ]   ENABLE_HYPER_THREADING: False
[ INFO ]   EXECUTION_DEVICES: ['CPU']
[ INFO ]   CPU_DENORMALS_OPTIMIZATION: False
[ INFO ]   LOG_LEVEL: Level.NO
[ INFO ]   CPU_SPARSE_WEIGHTS_DECOMPRESSION_RATE: 1.0
[ INFO ]   ENABLE_TENSOR_PARALLEL: False
[ INFO ]   DYNAMIC_QUANTIZATION_GROUP_SIZE: 32
[ INFO ]   KV_CACHE_PRECISION: <Type: 'uint8_t'>
[ INFO ]   KEY_CACHE_PRECISION: <Type: 'uint8_t'>
[ INFO ]   VALUE_CACHE_PRECISION: <Type: 'uint8_t'>
```

```
[ INFO ] KEY_CACHE_GROUP_SIZE: 0
[ INFO ] VALUE_CACHE_GROUP_SIZE: 0
```

Cmd 47 shows the steps to set up OpenVINO™ model conversion and to install the requirements.

Cmd 47. Commands to Setup OpenVINO™ Model Conversion Requirements

```
$ sudo apt install python3.12-venv
$ python3.12 -m venv openvino_env
$ source openvino_env/bin/activate
$ python -m pip install --upgrade pip
$ pip install openvino==2025.4.1

$ pip install tensorflow-cpu==2.19.1
$ pip install tf-keras==2.19.0
$ pip install torch==2.8.0+cpu \
  torchvision==0.23.0+cpu torchaudio==2.8.0+cpu \
  --index-url https://download.pytorch.org/whl/cpu
$ pip install onnx==1.19.0 onnxscript==0.4.0 onnxruntime==1.22.1
$ pip install ultralytics==8.3.189
$ pip install transformers==4.53.3 optimum[openvino]==1.27.0
```

Cmd 48 shows the example to convert TensorFlow* to OpenVINO™ IR model. Whereas Cmd 49, Cmd 50, Cmd 51, Cmd 52, Cmd 53, and Cmd 54 show the PyTorch* to OpenVINO™ IR model conversions.

Cmd 48. Commands to Convert ResNet50 TensorFlow* to OpenVINO™ IR Model

```
$ mkdir -p models/public/resnet-50-tf/FP32
$ mkdir -p models/public/resnet-50-tf/FP16

$ vim download_resnet-50-tf.py
import tensorflow as tf
import openvino as ov

tf.keras.config.set_image_data_format('channels_first')
# Download model from TensorFlow
tf_model = tf.keras.applications.ResNet50(
    include_top=True,
    weights='imagenet',
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation='softmax'
)
```

```
# Convert model to OpenVINO IR
ov_model = ov.convert_model(
    tf_model,
    input=[1, 3, 224, 224]
)
ov.save_model(
    ov_model,
    './models/public/resnet-50-tf/FP32/resnet-50-tf.xml',
    compress_to_fp16=False)
ov.save_model(
    ov_model,
    './models/public/resnet-50-tf/FP16/resnet-50-tf.xml',
    compress_to_fp16=True)

$ python download_resnet-50-tf.py
$ ls -l ./models/public/resnet-50-tf/FP32
total 99896
-rw-rw-r-- 1 ... 102121932 ... resnet-50-tf.bin
-rw-rw-r-- 1 ... 164323 ... resnet-50-tf.xml
$ ls -l ./models/public/resnet-50-tf/FP16
total 50092
-rw-rw-r-- 1 ... 51060986 ... resnet-50-tf.bin
-rw-rw-r-- 1 ... 227649 ... resnet-50-tf.xml
```

Cmd 49. Commands to Convert MobileNet V2 PyTorch* to OpenVINO™ IR Model

```
$ mkdir -p models/public/mobilenet_v2-pt/FP32
$ mkdir -p models/public/mobilenet_v2-pt/FP16

$ vim download_mobilenet_v2-pt.py
import torchvision
import torch
import openvino as ov

# Download model from PyTorch
pt_model = torchvision.models.mobilenet_v2(
    weights='IMAGENET1K_V2'
)

# Convert model to OpenVINO IR
ov_model = ov.convert_model(
    pt_model,
    input=[1,3,224,224]
)
ov.save_model(
    ov_model,
    './models/public/mobilenet_v2-pt/FP32/mobilenet_v2-pt.xml',
```

```

        compress_to_fp16=False)
ov.save_model(
    ov_model,
    './models/public/mobilenet_v2-pt/FP16/mobilenet_v2-pt.xml',
    compress_to_fp16=True)

$ python download_mobilenet_v2-pt.py
$ ls -l ./models/public/mobilenet_v2-pt/FP32
total 13764
-rw-rw-r-- 1 ... 13951296 ... mobilenet_v2-pt.bin
-rw-rw-r-- 1 ... 138521 ... mobilenet_v2-pt.xml
$ ls -l ./models/public/mobilenet_v2-pt/FP16
total 7012
-rw-rw-r-- 1 ... 6975664 ... mobilenet_v2-pt.bin
-rw-rw-r-- 1 ... 199717 ... mobilenet_v2-pt.xml

```

Cmd 50. Commands to Convert DeepLabV3 PyTorch* to OpenVINO™ IR Model

```

$ mkdir -p models/public/deeplabv3/FP32
$ mkdir -p models/public/deeplabv3/FP16

$ vim download_deeplabv3.py
import torchvision
import torch
import openvino as ov

# Download model from PyTorch
pt_model =
torchvision.models.segmentation.deeplabv3_mobilenet_v3_large(
    weights='COCO_WITH_VOC_LABELS_V1'
)
torch.onnx.export(
    pt_model,
    torch.randn(1,3,513,513),
    './models/public/deeplabv3/FP32/deeplabv3.onnx",
    opset_version = 11
)

ov_model = ov.convert_model(
    './models/public/deeplabv3/FP32/deeplabv3.onnx",
    input=[1,3,513,513]
)
ov.save_model(
    ov_model,
    './models/public/deeplabv3/FP32/deeplabv3.xml',
    compress_to_fp16=False)
ov.save_model(
    ov_model,

```

```

'./models/public/deeplabv3/FP16/deeplabv3.xml',
compress_to_fp16=True)

$ python download_deeplabv3.py
$ ls -l models/public/deeplabv3/FP32/
total 86364
-rw-rw-r-- 1 ... 44061384 ... deeplabv3.bin
-rw-rw-r-- 1 ... 44113745 ... deeplabv3.onnx
-rw-rw-r-- 1 ... 255251 ... deeplabv3.xml
$ ls -l models/public/deeplabv3/FP16/
total 21876
-rw-rw-r-- 1 user user 22037640 ... deeplabv3.bin
-rw-rw-r-- 1 user user 359288 ... deeplabv3.xml

```

Cmd 51. Commands to Convert Faster R-CNN PyTorch* to OpenVINO™ IR Model

```

$ mkdir -p models/public/fasterrcnn_resnet50_fpn_v2_coco/FP32
$ mkdir -p models/public/fasterrcnn_resnet50_fpn_v2_coco/FP16

$ vim download_fasterrcnn_resnet50_fpn_v2_coco.py
import torchvision
import torch
import openvino as ov

pt_model = torchvision.models.detection.fasterrcnn_resnet50_fpn_v2(
    weights='COCO_V1'
)
torch.onnx.export(
    pt_model,
    torch.randn(1,3,640,640),

"./models/public/fasterrcnn_resnet50_fpn_v2_coco/FP32/fasterrcnn_resnet
50_fpn_v2_coco.onnx",
    input_names = ['input'],
    output_names = ['boxes', 'labels', 'scores'],
    opset_version = 11
)

ov_model = ov.convert_model(

"./models/public/fasterrcnn_resnet50_fpn_v2_coco/FP32/fasterrcnn_resnet
50_fpn_v2_coco.onnx",
    input=[1,3,640,640]
)
ov.save_model(ov_model,
'./models/public/fasterrcnn_resnet50_fpn_v2_coco/FP32/fasterrcnn_resnet
50_fpn_v2_coco.xml', compress_to_fp16=False)
ov.save_model(ov_model,

```

```
'./models/public/fasterrcnn_resnet50_fpn_v2_coco/FP16/fasterrcnn_resnet50_fpn_v2_coco.xml', compress_to_fp16=True)

$ python download_fasterrcnn_resnet50_fpn_v2_coco.py
$ ls -l models/public/fasterrcnn_resnet50_fpn_v2_coco/FP32
total 342280
-rw-rw-r-- ... 174732324 ... fasterrcnn_resnet50_fpn_v2_coco.bin
-rw-rw-r-- ... 174971580 ... fasterrcnn_resnet50_fpn_v2_coco.onnx
-rw-rw-r-- ... 775006 ... fasterrcnn_resnet50_fpn_v2_coco.xml
$ ls -l models/public/fasterrcnn_resnet50_fpn_v2_coco/FP16
total 86200
-rw-rw-r-- ... 87366878 ... fasterrcnn_resnet50_fpn_v2_coco.bin
-rw-rw-r-- ... 898189 ... fasterrcnn_resnet50_fpn_v2_coco.xml
```

Cmd 52. Commands to Convert Mask R-CNN PyTorch* to OpenVINO™ IR Model

```
$ mkdir -p models/public/maskrcnn_resnet50_fpn_v2_coco/FP32
$ mkdir -p models/public/maskrcnn_resnet50_fpn_v2_coco/FP16

$ vim download_maskrcnn_resnet50_fpn_v2_coco.py
import torchvision
import torch
import openvino as ov

pt_model = torchvision.models.detection.maskrcnn_resnet50_fpn_v2(
    weights='COCO_V1'
)
torch.onnx.export(
    pt_model,

    torch.randn(1,3,640,640),

    './models/public/maskrcnn_resnet50_fpn_v2_coco/FP32/maskrcnn_resnet50_fpn_v2_coco.onnx",
    input_names = ['input'],
    output_names = ['boxes', 'labels', 'scores', 'masks'],
    opset_version = 11
)

ov_model = ov.convert_model(

    './models/public/maskrcnn_resnet50_fpn_v2_coco/FP32/maskrcnn_resnet50_fpn_v2_coco.onnx",
    input=[1,3,640,640]
)
ov.save_model(ov_model,
    './models/public/maskrcnn_resnet50_fpn_v2_coco/FP32/maskrcnn_resnet50_fpn_v2_coco.xml', compress_to_fp16=False)
ov.save_model(ov_model,
```

```
'./models/public/maskrcnn_resnet50_fpn_v2_coco/FP16/maskrcnn_resnet50_fpn_v2_coco.xml', compress_to_fp16=True)

$ python download_maskrcnn_resnet50_fpn_v2_coco.py
$ ls -l models/public/maskrcnn_resnet50_fpn_v2_coco/FP32
total 363192
-rw-rw-r-- ... 185316850 ... maskrcnn_resnet50_fpn_v2_coco.bin
-rw-rw-r-- ... 185613564 ... maskrcnn_resnet50_fpn_v2_coco.onnx
-rw-rw-r-- ... 973565 ... maskrcnn_resnet50_fpn_v2_coco.xml
$ ls -l models/public/maskrcnn_resnet50_fpn_v2_coco/FP16
total 91580
-rw-rw-r-- ... 92659190 ... maskrcnn_resnet50_fpn_v2_coco.bin
-rw-rw-r-- ... 1115978 ... maskrcnn_resnet50_fpn_v2_coco.xml
```

Cmd 53. Commands to Convert Keypoint R-CNN PyTorch* to OpenVINO™ Toolkit IR Model

```
$ mkdir -p models/public/keypointtrcnn_resnet50_fpn/FP32
$ mkdir -p models/public/keypointtrcnn_resnet50_fpn/FP16

$ vimdownload_keypointtrcnn_resnet50_fpn.py
import torchvision
import torch
import openvino as ov

pt_model = torchvision.models.detection.keypointtrcnn_resnet50_fpn(
    weights='COCO_V1'
)
torch.onnx.export(
    pt_model,
    torch.randn(1,3,640,640),

    './models/public/keypointtrcnn_resnet50_fpn/FP32/keypointtrcnn_resnet50_fpn.onnx',
    input_names = ['input'],
    output_names = ['boxes', 'labels', 'scores', 'keypoints'],
    opset_version = 11
)

ov_model = ov.convert_model(

    './models/public/keypointtrcnn_resnet50_fpn/FP32/keypointtrcnn_resnet50_fpn.onnx',
    input=[1,3,640,640]
)
ov.save_model(ov_model,
    './models/public/keypointtrcnn_resnet50_fpn/FP32/keypointtrcnn_resnet50_fpn.xml', compress_to_fp16=False)
ov.save_model(ov_model,
```

```

'./models/public/keypointtrcnn_resnet50_fpn/FP16/keypointtrcnn_resnet50_fpn.xml', compress_to_fp16=True)

$ python download_keypointtrcnn_resnet50_fpn.py
$ ls -l models/public/keypointtrcnn_resnet50_fpn/FP32
total 463532
-rw-rw-r-- ... 236657298 ... keypointtrcnn_resnet50_fpn.bin
-rw-rw-r-- ... 237075165 ... keypointtrcnn_resnet50_fpn.onnx
-rw-rw-r-- ... 918245 Sep 1 20:06 keypointtrcnn_resnet50_fpn.xml
$ ls -l models/public/keypointtrcnn_resnet50_fpn/FP16
total 116588
-rw-rw-r-- ... 118329474 ... keypointtrcnn_resnet50_fpn.bin
-rw-rw-r-- ... 1051261 ... keypointtrcnn_resnet50_fpn.xml

```

Cmd 54. Commands to Convert Wav2Vec2-Base PyTorch* to OpenVINO™ IR Model

```

$ mkdir -p models/public/wav2vec2-base/FP32
$ mkdir -p models/public/wav2vec2-base/FP16

$ vim download_wav2vec2-base.py
import torch
import transformers
import openvino as ov

pt_model = transformers.Wav2Vec2ForCTC.from_pretrained(
    "facebook/wav2vec2-base-960h")
torch.onnx.export(
    pt_model,
    torch.randn(1,30480),
    "./models/public/wav2vec2-base/FP32/wav2vec2-base.onnx",
    input_names=["input"],
    output_names=["hidden_states"],
    dynamic_axes={
        "input":{0:"batch_size"},
        "hidden_states":{0:"batch_size"}
    },
    opset_version=14
)

ov_model = ov.convert_model(
    "./models/public/wav2vec2-base/FP32/wav2vec2-base.onnx",
    input=[1,30480]
)
ov.save_model(
    ov_model,
    './models/public/wav2vec2-base/FP32/wav2vec2-base.xml',
    compress_to_fp16=False)
ov.save_model(

```

```

ov_model,
'./models/public/wav2vec2-base/FP16/wav2vec2-base.xml',
compress_to_fp16=True)

$ python download_wav2vec2-base.py
$ ls -l models/public/wav2vec2-base/FP32
total 738132
-rw-rw-r-- 1 ... 377581924 ... wav2vec2-base.bin
-rw-rw-r-- 1 ... 377843102 ... wav2vec2-base.onnx
-rw-rw-r-- 1 ... 414966 ... wav2vec2-base.xml
$ ls -l models/public/wav2vec2-base/FP16
total 184900
-rw-rw-r-- 1 ... 188791074 ... wav2vec2-base.bin
-rw-rw-r-- 1 ... 542050 ... wav2vec2-base.xml

```

Cmd 55 shows model conversion from Ultralytics* to OpenVINO™ IR. This method is applicable not only for object detection models but also for the segmentation as well as the pose estimation models.

Cmd 55. Commands to Convert Ultralytics* to OpenVINO™ IR Model

```

$ mkdir -p models/public/yolo11x/FP16
$ mkdir -p models/public/yolo11x-nms/FP16

$ vim download_yolo11x.py
import shutil
from ultralytics import YOLO

det_model = YOLO(f"yolo11x.pt")
det_model.export(format="openvino", dynamic=False, half=True)
shutil.rmtree('models/public/yolo11x/FP16')
shutil.move(
    'yolo11x_openvino_model',
    'models/public/yolo11x/FP16')

det_model.export(
    format="openvino", dynamic=False, half=True, nms=True)
shutil.rmtree('models/public/yolo11x-nms/FP16')
shutil.move(
    'yolo11x_openvino_model', 'models/public/yolo11x-nms/FP16')
shutil.move(
    'models/public/yolo11x-nms/FP16/yolo11x.xml',
    'models/public/yolo11x-nms/FP16/yolo11x-nms.xml')
shutil.move(
    'models/public/yolo11x-nms/FP16/yolo11x.bin',
    'models/public/yolo11x-nms/FP16/yolo11x-nms.bin')

$ python download_yolo11x.py
$ ls -l models/public/yolo11x/FP16/

```

```
total 112000
-rw-rw-r-- 1 user user      1517 ... metadata.yaml
-rw-rw-r-- 1 user user 113889544 ... yolo11x.bin
-rw-rw-r-- 1 user user   788971 ... yolo11x.xml
$ ls -l models/public/yolo11x-nms/FP16/
total 112060
-rw-rw-r-- 1 user user      1516 ... metadata.yaml
-rw-rw-r-- 1 user user 113889722 ... yolo11x-nms.bin
-rw-rw-r-- 1 user user   848728 ... yolo11x-nms.xml
```

6.2 OpenVINO™ Toolkit Model Compression

[Model compression](#) is targeted to improve inference performance and reduce the required storage and memory sizes. [Neural Network Compression Framework](#) is the default OpenVINO™ toolkit model compression tool that offers a minimal accuracy impact. NNCF has been integrated into many tools, and one of them is [Hugging Face* Optimum Intel](#).

Please follow Cmd 56 to set up the OpenVINO™ toolkit model compression and33h install the requirements using [OpenVINO™ GenAI library](#).

Cmd 56. Commands to Setup OpenVINO™ Toolkit Model Compression Requirements

```
$ python3 -m venv ov-llm-bench-env
$ source ov-llm-bench-env/bin/activate
$ pip install --upgrade pip

$ git clone https://github.com/openvinotoolkit/openvino.genai.git
$ cd openvino.genai
$ git checkout 2025.4.1.0
$ cd tools/llm_bench
$ pip install -r requirements.txt
```

Cmd 57 shows the steps to compress and benchmark Visual Language Models (VLMs) using Intel® Optimum CLI.

Cmd 57. Commands to Compress and Benchmark VLMs*

```
$ hf auth login
$ optimum-cli export openvino \
  --model google/gemma-3-12b-it \
  --weight-format int4 \
  models/gemma-3-12b-it-int4
$ python benchmark.py \
  -m "models/gemma-3-12b-it-int4" \
  -p "What is openvino?" \
  -n 1 \
  --task visual_text_gen \
```

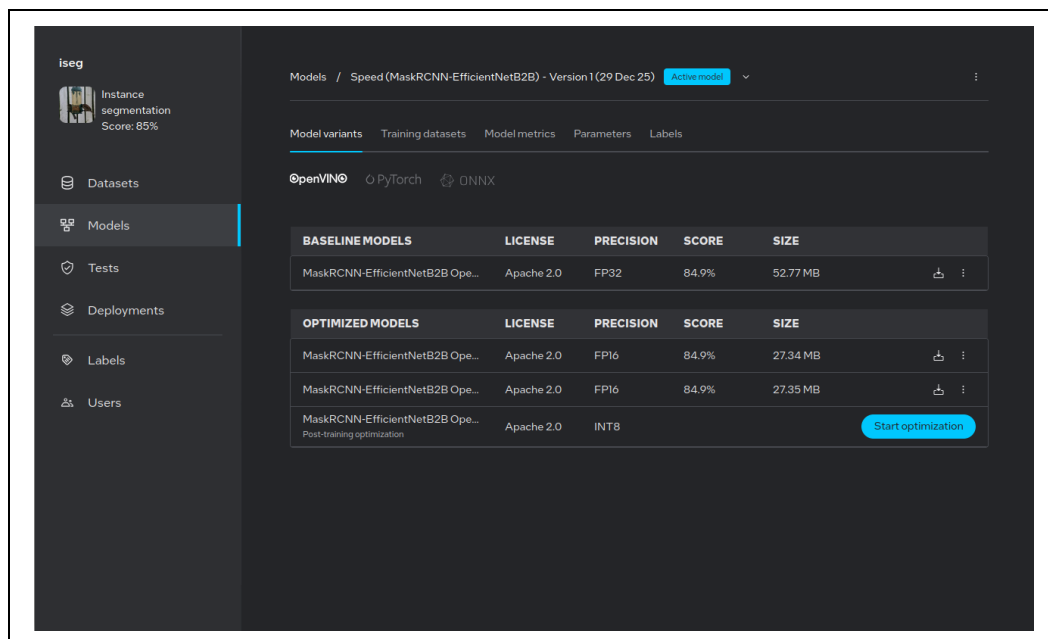
```
-i ./image1080p.png \
-d GPU
```

Refer to the [Hugging Face* website hosts optimized OpenVINO™ Toolkit IR models](#) for faster and easier benchmarking and deployment. It also has the [LLM models specifically optimized for Intel® NPU](#).

6.3 Geti™ Application Model Quantization

After model training, the Geti™ application automatically generates the OpenVINO™ toolkit models in FP32 and FP16 precisions. The Geti™ application also provides INT8 post-training quantization as shown in Figure 6. As the model is quantized using the training dataset, it provides better accuracy than the model quantized using random inputs.

Figure 6. Geti™ Application Model Quantization



6.4 Channels-First and Channels-Last

Historically, channels-last input shape runs best on Intel® processor compute units. For the latest OpenVINO™ toolkit 2025, channels-first and channels-last inference performances are identical on the CPU and GPU. The NPU, on the other hand, will benefit significantly (up to 10%) when using the channels-last.

Cmd 48 shows the implementation of the channels-first with 1x3x224x224 pixels input shape. You can modify it to channels-last with the channels_last parameter and a 1x224x224x3 pixels input shape.

Since changing from channels-first to channels-last and vice versa does not modify the model weight (just the shape), it is recommended to generate both channels and then test the performance on different compute units using the OpenVINO™ `benchmark_app` tool.

6.5 Static and Dynamic Input Shapes

Static shape is recommended for better performance as compared to dynamic shape, especially when running on a GPU or NPU. When using the OpenVINO™ toolkit, there are two locations to define the shape: during [model conversion](#) and during [model compilation](#). Cmd 48 also shows the steps to use a static shape during model conversion.

NLP models normally have variable input length. Therefore, it is recommended to use dynamic input shape running on CPU. For some models with dynamic shape, running them on GPU or NPU are not even an option.

6.6 Input Batching

Batching multiple inputs is recommended to improve performance as the model layers can be executed concurrently in a pipeline manner. The OpenVINO™ toolkit has implemented automatic batching. Developers are not required to concatenate multiple inputs and send them to a model. Instead, developers can just send the individual inputs to the model.

Sending a large batch to a compute unit could cause the memory to crash. The latest OpenVINO™ toolkit NPU implementation (since version 2025.4) will instead break the batch into one and perform the inference individually. Internally, the NPU is still performing memory-aware automatic batching.

6.7 Input and Output Sizes

Subsection 5.2 increases model input size to improve accuracy. On the contrary, reducing the input size can improve performance at the cost of lower accuracy. Input resizing is part of post-training optimization and does not affect the model's weights. It can be done during [model conversion](#) and during [model compilation](#). Cmd 48 also shows the steps to set the input size during model conversion.

Reducing output size can also increase performance. If the implementation needs only 10 classes, pre-trained models with 1000 classes will have lower inference performance. Reducing output size can be done through transfer learning or training from scratch. The Geti™ application can help with reducing the output size using transfer learning.

6.8 Heterogeneous Processing

Some models have a part of the layers that are highly parallelized, and another part consists of sequential operations. The highly parallelized portion runs best on a GPU or an NPU, whereas the sequential if-else portion runs best on a CPU. [YOLOv7](#) is one of the models. The backbone consists of highly parallelized convolutional layers, and the head performs many if-else operations to calculate Non-Maximum Suppression.

The following example uses the YOLOv7 model to run the backbone on the GPU and the head on the CPU. The sample code will use the Intel® Deep Learning Streamer (DL Streamer) which will be explored in more detail in subsection 7.1. Table 16 shows the Software BOMs.

Table 16. YOLOv7 Heterogeneous Processing Software BOMs

Component	Information
Operating System	Ubuntu* 2022.04.1 LTS
Python*	3.8.10
OpenVINO™ Toolkit	2022.3
ONNX	1.12.0
ONNX Runtime	1.12.1
PyTorch*	1.13.0+cpu
NNCF	2.4.0
Intel® DL Streamer	2022.2.0-ubuntu20-gpu815-dpcpp-devel

Cmd 58 shows the steps to export YOLOv7 backbone to ONNX model. The `export.py` code can be found in [YOLOv7 code repository](#). Afterwards, convert the ONNX model into OpenVINO™ Toolkit IR model with INT8 quantization as shown in Cmd 59. To perform quantization, please install NNCF and download [COCO dataset](#).

Cmd 58. Commands to Export YOLOv7 PyTorch* Model to ONNX Model

```
$ python export.py \
  --weights yolov7-w6/yolov7-w6.pt \
  --img-size 832 \
  --grid

$ mv yolov7-w6/yolov7-w6.onnx yolov7-w6/yolov7-w6_backbone.onnx
```

Cmd 59. Commands to Export YOLOv7 ONNX Model to OpenVINO™ IR INT8 Quantized Model

```
from openvino.tools import mo
from openvino.runtime import serialize
```

```

from opencv.runtime import Layout, PartialShape

model = mo.convert_model(
    'yolov7-w6/yolov7-w6_backbone.onnx',
    input_shape = PartialShape([1, 3, 832, 832]),
    layout = Layout("NCHW"),
    mean_values = [0, 0, 0],
    scale_values = [255, 255, 255],
)

from collections import namedtuple
import yaml
from utils.datasets import create_dataloader
from utils.general import check_dataset, box_iou, xywh2xyxy, colorstr

DATA_CONFIG = 'data/coco.yaml'
with open(DATA_CONFIG) as f:
    data = yaml.load(f, Loader=yaml.SafeLoader)

TASK = 'val' # path to train/val/test images
Option = namedtuple('Options', ['single_cls'])
opt = Option(False)
dataloader = create_dataloader(
    data[TASK], 832, 1, 32, opt, pad=0.5,
    prefix=colorstr(f'{TASK}: ')
)[0]

def transform_fn(data_item, preprocess: bool = False):
    img = data_item[0].numpy()
    if preprocess:
        return img
    input_tensor = prepare_input_tensor(img)
    return input_tensor

quantization_dataset = nncf.Dataset(dataloader, transform_fn)
quantized_model = nncf.quantize(
    model, quantization_dataset,
    preset=nncf.QuantizationPreset.MIXED)
serialize(quantized_model, 'yolov7-w6/yolov7-w6.xml')

```

Cmd 60 shows the YOLOv7 post-processing that consists of the Non-Maximum Suppression calculation as well as the prediction result parsing.

Cmd 60. Commands of YOLOv7 Post-Processing

```

$ vim object_detection_stream_yolov7_ir_post_proc.py
import sys
sys.path.insert(0, './yolov7')

```

```
import numpy as np
import torch
from gstgva import VideoFrame
from utils.general import non_max_suppression

NAMES = ['person', 'bicycle', 'car', 'motorcycle',
'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
'fire hydrant', 'stop sign', 'parking meter',
'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
'elephant', 'bear', 'zebra', 'giraffe', 'backpack',
'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat',
'baseball glove', 'skateboard', 'surfboard', 'tennis racket',
'bottle', 'wine glass', 'cup', 'fork', 'knife',
'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange',
'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake',
'chair', 'couch', 'potted plant', 'bed', 'dining table',
'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard',
'cell phone', 'microwave', 'oven', 'toaster', 'sink',
'refrigerator', 'book', 'clock', 'vase', 'scissors',
'teddy bear', 'hair drier', 'toothbrush']

class YOLOv7_IR_Post_Proc:
    def __init__(self):
        self.conf_thres = 0.25
        self.iou_thres = 0.45
        self.classes = None
        self.agnostic_nms = None

    def process_frame(self, frame: VideoFrame) -> bool:
        if len([i for i in frame.tensors()]):
            # reshape tensor to YOLOv7 format
            for tensor in frame.tensors():
                pred = torch.reshape(
                    torch.from_numpy(tensor.data()),
                    tensor.dims()
                )

            # non-max suppression
            pred = non_max_suppression(
                pred, self.conf_thres, self.iou_thres,
                classes=self.classes, agnostic=self.agnostic_nms)

            # enumerate through each prediction in format
            # [x1, y1, x2, y2, score, label]
            for i, det in enumerate(pred[0]):
                # get bounding box size
```

```

        box = np.array([det[0], det[1], det[2], det[3]])
        box = box.round().astype(np.int32).tolist()
        frame.add_region(
            x=det[0],
            y=det[1],
            w=box[2] - box[0],
            h=box[3] - box[1],
            label=NAMES[int(det[5])],
            confidence=det[4],
            normalized=False,
        )
    return True

```

Having both the backbone in INT8 as well as the Non-Maximum Suppression head, the Cmd 61 shows the steps to run the Intel® DL Streamer pipeline utilizing both GPU for the backbone and CPU for the head.

Cmd 61. Commands to Run YOLOv7 Heterogeneous Processing

```

$ env \
DETECTION_MODEL_YOLOv7_W6_IR_INT8_832x832="./yolov7-
w6/INT8/yolov7-w6.xml"
$ env \
DETECTION_MODEL_YOLOv7_IR_POST_PROC="./object_detection_stream_yo
lov7_ir_post_proc.py"

$ gst-launch-1.0 \
  filesrc location=./horses.jpg ! \
  jpegdec ! \
  videoscale ! video/x-raw,width=832,height=832 ! \
  gvainference model=${DETECTION_MODEL_YOLOv7_W6_IR_INT8_832x832}
device=GPU ! \
  gvapython module=${DETECTION_MODEL_YOLOv7_IR_POST_PROC}
class=YOLOv7_IR_Post_Proc function=process_frame ! \
  gvametaconvert json-indent=4 source=horses.jpg ! \
  gvametapublish method=file ! \
  fakesink sync=false

```

6.9 Asynchronous Inferencing using Geti™ Application SDK

The Geti™ application trained model can be deployed using the Geti™ application SDK. More details on Geti™ application SDK can be found in subsection 7.3. The Geti™ application SDK has the `set_asynchronous_callback` function to run post-processing asynchronously to the model inference. If both post-processing and model inference latencies are about the same, the total inference time can be down in half. Please find [Geti™ Application Asynchronous Inference notebook](#) for the sample code.

6.10 Core Pinning

Core pinning helps maintain inference performance stability. On Linux*, the OpenVINO™ toolkit enables CPU pinning by default. Other than core pinning, the OpenVINO™ toolkit is able to assign model inference to a specific core type, either P-core or E-core. Please refer to [OpenVINO™ Performance Hints and Thread Scheduling](#) documentation for the core pinning parameters.

6.11 Model Caching

Another method to improve system performance and stability is to use model caching. Since the OpenVINO™ toolkit supports multiple compute units (CPU, GPU, and NPU), it must compile the model for the target compute unit when loading it. Model compilation consumes high CPU load that may cause system instability. Model caching can help to reduce the high CPU load and model loading time. Please start with [OpenVINO™ Toolkit Model Caching Overview](#) for more details.

§

7.0 Deployment

This section focuses on two types of inference pipelines for model deployment: streaming and batching. Streaming is crucial for edge AI implementation that requires real-time response using latency optimization. Whereas batching is suitable for backend data processing, relying on throughput optimization.

Intel® DL Streamer is one of the tools for streaming. It is built on top of GStreamer* that supports multiple input/output image, video, and audio formats. Intel® DL Streamer adds support for inferencing with the OpenVINO™ toolkit and Geti™-Trained application models, along with pre- and post-processing.

For batching tools, this section will use the OpenVINO™ Model Server (OVMS) for OpenVINO™ toolkit models as well as Geti™ SDK for Geti™ trained models.

7.1 Intel® DL Streamer

This subsection will go through the steps to deploy the Geti™ application-trained model using [Intel® DL Streamer](#). Examples include segmentation and anomaly detection tasks. It is tested using Ubuntu* 24.04 LTS and Geti™ 2.10.1.

7.1.1 Modify Pre- and Post-Processing

Figure 7 shows the location to modify image loading for the OpenVINO™ toolkit inference flow. The sample code updates channel information for model pre-processing. `openvino_image_inference.cpp` also contains the functions to load OpenVINO™ toolkit IR model binary (.bin) and the configuration (.xml) files, as well as the function to reshape input model.

Figure 8 shows the example to swap input channels in the pre- and post-processing (`inference_impl.cpp`) file. Inside the file, the `UpdateConfigWithLayerInfo` function can be modified to alter the input normalization calculation.

Figure 9 shows the location to modify post-processing code for the Geti™ application-trained model. The sample code modifies both threshold normalization and prediction scoring calculations.

Figure 7. Intel® DL Streamer OpenVINO™ Toolkit Input Image Loading

```

1131 1131
1132 + // IE color conversion on OPENCV flow
1133 + if (pp_type == ImagePreprocessorType::OPENCV) {
1134 + // OpenCV reads image in to BGR format
1135 + const auto color_format_pair = get_ov_color_format("BGR");
1136 + input.tensor().set_color_format(color_format_pair.first, color_format_pair.second);
1137 + // Convert input tensor color space into model color space if necessary
1138 + const auto model_format_pair = get_ov_color_format(config.model_format());
1139 + if (color_format_pair.first != model_format_pair.first)
1140 + input.preprocess().convert_color(model_format_pair.first);
1141 + }
1142 +
1132 1143 // OV preprocessing is configured only for IE or VAAPI_SURFACE_SHARING
1133 1144 if (pp_type == ImagePreprocessorType::VAAPI_SURFACE_SHARING || pp_type == ImagePreprocessorType::IE) {
1134 1145

```

Figure 8. Intel® DL Streamer Pre-Processing

```

430 430 int reverse_channels = 0; // TODO: verify that channel reversal works correctly with mean and std!
431 431 if (gst_structure_get_int(it->params, "reverse_input_channels", &reverse_channels)) {
432 - config[KEY_BASE][KEY_MODEL_FORMAT] = reverse_channels ? "RGB" : "BGR";
432 + config[KEY_BASE][KEY_MODEL_FORMAT] = reverse_channels ? "BGR" : "RGB";
433 433 }
434 434 }
435 435 }

```

Figure 9. Intel® DL Streamer Geti™ Application Post-Processing

```

33 33 double pred_score = 0.0;
34 - double image_threshold_norm = normalize(image_threshold, 0.0f);
34 + double image_threshold_norm = image_threshold / normalization_scale;
35 35 cv::Mat anomaly_map;
36 36 cv::Mat pred_mask;
37 37 std::string pred_label = "";
38 38
39 39 @ -87,6 +87,12 @@ TensorsTable DetectionAnomalyConverter::convert(const OutputBlobs &output_blobs)
87 87 pred_label);
88 88 }
89 89
90 + pred_score = (pred_score - image_threshold_norm) + 0.5f;
91 + pred_score = std::min(std::max(pred_score, 0.), 1.);
92 + if (pred_label == labels[0]) { // normal label
93 + pred_score = 1 - pred_score; // Score of normal is 1 - score of anomaly
94 + }
95 +
96 96 GVA_INFO("pred_label: %s, pred_score: %f, image_threshold: %f, "
97 97 "image_threshold_norm: %f, normalization_scale: %f, #normal: %u, #anomaly: %u",
98 98 pred_label.c_str(), pred_score, image_threshold, image_threshold_norm, normalization_scale,

```

7.1.2 Build Intel® DL Streamer Docker* Image

Cmd 62 shows the steps to build the Intel® DL Streamer Docker* image. After cloning the Edge AI Libraries repository, modify the pre- and post-processing before building the Docker* image. Building the Docker* image will also compile the pre- and post-processing C++ codes.

Cmd 62. Commands to Build Intel® DL Streamer Docker* Image

```
$ sudo snap install docker

$ git clone \
  https://github.com/open-edge-platform/edge-ai-libraries.git
$ cd [path_to]/edge-ai-libraries
$ git checkout -b 47a0f7f

$ cd [path_to]/edge-ai-libraries/libraries/dl-streamer/thirdparty
$ rm -r spdlog
$ git clone https://github.com/gabime/spdlog.git
$ cd spdlog
$ git checkout -b 6fa3601

$ cd [path_to]/edge-ai-libraries/libraries/dl-streamer
$ sudo docker build \
  -f [path_to]/edge-ai-libraries/libraries/dl-
streamer/docker/devel/ubuntu24/dlstreamer_dev_ubuntu24.Dockerfile \
  -t dlstreamer:main-geti-dev-ubuntu24 .

$ sudo docker images
REPOSITORY      TAG                ...    SIZE
dlstreamer      main-geti-dev-ubuntu24  ...    7.85GB
```

7.1.3 Run Intel® DL Streamer Docker* Image

Download the Geti™ application model deployment package, extract it, and run the model inference. Please follow Cmd 63 for instance segmentation task and Cmd 64 for the anomaly detection task.

Cmd 63. Commands to Run Intel® DL Streamer Docker* Image for Instance Segmentation

```
$ sudo docker run \
  -v
  [path_to_input_video]/products.mp4:/home/dlstreamer/intel/dl_streamer/d
  atasets/products.mp4 \
  -v
  "[path_to_input_image]/is_1080p.jpg":/home/dlstreamer/intel/dl_streamer
  /datasets/is_1080p.jpg \
  -v "[path_to_geti_model]/geti_sdk-deployment/deployment/Instance
```

```
segmentation/model/model.xml":/home/dlstreamer/intel/dl_streamer/models
/intel/geti-segmentation/FP16/geti-segmentation.xml \
-v "[path_to_geti_model]/geti_sdk-deployment/deployment/Instance
segmentation/model/model.bin":/home/dlstreamer/intel/dl_streamer/models
/intel/geti-segmentation/FP16/geti-segmentation.bin \
-e DISPLAY=$DISPLAY \
-v /tmp/.X11-unix:/tmp/.X11-unix \
--net=host \
--privileged \
--user root \
--device /dev/dri \
--device /dev/accel \
-it --rm \
dlstreamer:main-geti-dev-ubuntu24

# cd
/home/dlstreamer/dlstreamer/samples/gstreamer/gst_launch/geti_deploymen
t
# export MODELS_PATH="/home/dlstreamer/intel/dl_streamer/models/"
# bash geti_sample.sh \
instance-segmentation \
intel/geti-segmentation/FP16/geti-segmentation.xml \
GPU \
ie \
/home/dlstreamer/intel/dl_streamer/datasets/is_1080p.jpg \
json

# cat output.json
{"objects":[{"detection":{"bounding_box":{"x_max":0.6959558725357056,"x
_min":0.49132728576660156,"y_max":0.5861016379462348,"y_min":0.17015473
047892252},"confidence":0.7433352470397949,"label":"bag","label_id":4},
"h":449,"region_id":122,"roi_type":"bag","tensors":[{"confidence":0.743
3352470397949,"label_id":4,"layer_name":"labels\\masks\\boxes","layout"
:"ANY","model_name":"main_graph","name":"detection","precision":"UNSPEC
IFIED"},{"data":[0.05509582534432411,...
,0.003786597866564989],"dims":[28,28],"format":"segmentation_mask","lay
er_name":"labels\\masks\\boxes","layout":"ANY","model_name":"main_graph
","name":"mask_rcnn","precision":"FP32"},{"w":393,"x":943,"y":184},...
],"w":327,"x":976,"y":739},"resolution":{"height":1080,"width":1920},"
timestamp":0}
```

Cmd 64. Commands to Run Intel® DL Streamer Docker* Image for Anomaly Detection

```
$ sudo docker run \
-v
"[path_to_input_image]/a_1080p.jpg":/home/dlstreamer/intel/dl_streamer/
datasets/a_1080p.jpg \
-v
```

```

"[path_to_input_image]/n_1080p.jpg":/home/dlstreamer/intel/dl_streamer/
datasets/n_1080p.jpg \
-v "[path_to_geti_model]/geti_sdk-deployment/deployment/Anomaly
classification/model/model.xml":/home/dlstreamer/intel/dl_streamer/mode
ls/intel/geti-anomaly-detection/padim/FP16/geti-anomaly-detection.xml \
-v "[path_to_geti_model]/geti_sdk-deployment/deployment/Anomaly
classification/model/model.bin":/home/dlstreamer/intel/dl_streamer/mode
ls/intel/geti-anomaly-detection/padim/FP16/geti-anomaly-detection.bin \
-e DISPLAY=$DISPLAY \
-v /tmp/.X11-unix:/tmp/.X11-unix \
--net=host \
--privileged \
--user root \
--device /dev/dri \
--device /dev/accel \
-it --rm \
dlstreamer:main-geti-dev-ubuntu24

# cd
/home/dlstreamer/dlstreamer/samples/gstreamer/gst_launch/geti_deploymen
t
# export MODELS_PATH="/home/dlstreamer/intel/dl_streamer/models/"
# bash geti_sample.sh \
anomaly-detection \
intel/geti-anomaly-detection/padim/FP16/geti-anomaly-detection.xml \
CPU \
ie \
/home/dlstreamer/intel/dl_streamer/datasets/n_1080p.jpg \
json

# cat output.json
{"objects":[{"classification_layer_name:output1":{"confidence":0.809818
8751191611,"label":"Anomaly","model":{"name":"Model6"}},"h":1080,"tenso
rs":[{"confidence":0.8098188751191611,"label":"Anomaly","layer_name":"o
utput1","layout":"ANY","model_name":"Model6","name":"classification_lay
er_name:output1","precision":"FP32"},"w":1920,"x":0,"y":0},"resolution":
{"height":1080,"width":1920},"timestamp":0}]}

```

7.2 OpenVINO™ Toolkit Model Server

This subsection provides an example of model deployment using [OpenVINO™ Model Server](#) (OVMS). The sample application will be an image classification task using the ResNet50 model.

7.2.1 Spin Up the OpenVINO™ Toolkit Model Server

Cmd 65 shows the steps to run the OpenVINO™ toolkit Model Server that includes downloading and extracting OVMS 2025.3, downloading ResNet50 OpenVINO™ IR model, and spinning up OVMS using dGPU as the target compute unit.

Cmd 65. Commands to Run OpenVINO™ Toolkit Model Server

```
$ wget
https://github.com/openvinotoolkit/model_server/releases/download/v2025
.3/ovms_ubuntu24.tar.gz
$ tar -xzf ovms_ubuntu24.tar.gz

$ sudo apt update -y && sudo apt install -y libxml2 curl
$ export LD_LIBRARY_PATH=${PWD}/ovms/lib
$ export PATH=$PATH:${PWD}/ovms/bin

$ curl --create-dirs -k
https://storage.openvinotoolkit.org/repositories/open_model_zoo/2022.1/
models_bin/2/resnet50-binary-0001/FP32-INT1/resnet50-binary-0001.xml -o
models/resnet50/1/model.xml
$ curl --create-dirs -k
https://storage.openvinotoolkit.org/repositories/open_model_zoo/2022.1/
models_bin/2/resnet50-binary-0001/FP32-INT1/resnet50-binary-0001.bin -o
models/resnet50/1/model.bin
$ chmod -R 755 models

$ ovms \
  --model_path models/resnet50/ \
  --model_name resnet50 \
  --port 9000 --rest_port 8000 --log_level DEBUG \
  --target_device GPU.1 \
  --plugin_config '{"PERFORMANCE_HINT": "LATENCY"}'
```

When the server is running, use the following URLs to check the deployed model information.

- [http://\[system_ip\]:8000/v1/models/resnet50/versions/1](http://[system_ip]:8000/v1/models/resnet50/versions/1)
- [http://\[system_ip\]:8000/v1/models/resnet50/versions/1/metadata](http://[system_ip]:8000/v1/models/resnet50/versions/1/metadata)

7.2.2 Perform Client Inference

Cmd 66 shows the steps to run inference using the images in the `input_images.txt` file. The file contains paths to images and their corresponding class IDs. In this case, the cat class has an ID of 281. The sample image classification code returns the predicted classes and compares them to the expected classes.

Regarding the parameters, you can get the model's input and output layer names by inspecting the model metadata at the previous URL. For the ResNet50 model, the input and output layer names are 0 and 1463, respectively.

Cmd 66. Commands to Run Client Inference

```
$ pip install tensorflow-serving-api opencv-python

$ vim input_images.txt
[path_to]/cat.jpg 281

$ git clone https://github.com/openvinotoolkit/model_server.git
$ cd model_server/demos/image_classification/python

$ python image_classification.py \
  --grpc_port 9000 \
  --input_name 0 \
  --output_name 1463 \
  --images_list input_images.txt \
  --model_name resnet50
```

7.3 Geti™ Application SDK

This subsection uses the Geti™ application SDK to deploy the Geti™ application trained models. Please note that it is recommended to use the same Geti™ SDK version as the one used to train the models. In this example, we will use Geti™ application SDK 1.2.2 for the model trained by Geti™ application 1.2.2.

Please note that the Geti™ application training requires exclusive system usage. Thus, use the Geti™ training server only for training and use another system for the Geti™ application SDK.

7.3.1 Build Geti™ Application SDK with Jupyter* Notebook Docker* Image

Cmd 67 shows the steps to build [Geti™ SDK](#) Docker* image. Cmd 68 shows the steps to add Jupyter* Notebook on top of the Geti™ SDK Docker* image. This is useful for running both Python* CLI and Python* notebook sample code.

Cmd 67. Commands to Build Geti™ Application SDK Docker* Image

```
$ sudo snap install docker

$ git clone https://github.com/openvinotoolkit/geti-sdk.git
$ cd geti_sdk
$ git checkout tags/v1.2.2
$ docker build \
  -f Dockerfile \
```

```
-t geti-sdk:1.2.2 .

$ docker images
REPOSITORY    TAG        ...    SIZE
geti-sdk      1.2.2     ...    2.98GB
python        3.8       ...    907MB
```

Cmd 68. Commands to Build Geti™ Application SDK with Jupyter* Notebook Docker* Image

```
$ vim Dockerfile-nb
FROM geti-sdk:1.2.2

WORKDIR /app

RUN pip install -r requirements/requirements-notebooks.txt

ENTRYPOINT ["sh", "-c", "jupyter lab --allow-root -y --no-browser --
ip=0.0.0.0"]

$ docker build \
  -f Dockerfile-nb \
  -t geti-sdk-nb:1.2.2 .

$ docker images
REPOSITORY    TAG        ...    SIZE
geti-sdk-nb   1.2.2     ...    3.19GB
geti-sdk      1.2.2     ...    2.98GB
python        3.8       ...    907MB
```

7.3.2 Run Geti™ Application SDK with Jupyter* Notebook Docker* Image

Cmd 69 shows the steps to run the Docker* image. Please access the container using a web browser with the system IP and port 8885. The Python* CLI sample codes are in the examples directory, whereas the Python* notebook sample codes are in the notebooks directory.

Cmd 69. Commands to Run Geti™ SDK with Jupyter* Notebook Docker* Image

```
$ docker run -it \
  -p 8885:8888 \
  --rm geti-sdk-nb:1.2.2
```

§



8.0 Applications

This section will show case multi-workload consolidation utilizing all compute units (CPU, GPU, and NPU) in the Intel® Core™ Ultra Processors. A CPU with low-latency processing is suitable for business-critical and AI applications that process sequential input and output. GPU as the most powerful compute unit gets significant performance improvement from generation to generation. Therefore, it can handle increasing loads from text LLM to image VLM, and to video VLM applications using the iGPU from Intel® Core™ Ultra Series 1 to Series 2, and to Series 3 respectively. Finally, NPU as the power efficient compute unit is suitable for 24x7 streaming applications like video analytics. We will use it to perform object detection from multiple cameras running around the clock.

For system setups, we will use the Linux* Kernel from multiple sources: the Ubuntu* default Kernel, the Ubuntu* OEM Kernel, and the Intel® Kernel. Similarly, for drivers, we have Ubuntu*-verified drivers and Intel® latest drivers. Finally, for the packages, the sources can be from the Ubuntu* OS, recommended packages, or Intel® packages. It is always recommended to use the Kernel, drivers, and packages verified by the OS vendor.

8.1 Text QnA

For a real-time Q&A application, we are targeting about 15 tokens/s with 7 billion INT4-quantized parameter LLM models. This will be translated into about 10 words/s, which is sufficient for human-level reading and listening. This QnA requirement can be run on Intel® Core™ Ultra Processor Series 1 iGPU. To complete the end-to-end implementation, we can add Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) running on the CPU. Finally, we can use the NPU for low-power, always-on object detection.

As AI workloads need additional features to be integrated directly into current business-critical applications, we will use containerization to ensure AI applications do not have access to them. Another benefit is the ability to limit resources (CPU and memory) allocated to the container. Please find Table 17 for the hardware and software BOMs.

Table 17. Text QnA Hardware and Software BOMs

Component	Information
Processor	Intel® Core™ Ultra 7 Processor 155H 6P+8E+2LPE Cores, 22 Threads 24 MB Intel® Smart Cache, up to 4.80 GHz Maximum Memory Size 96 GB

Component	Information
Memory	32 GB LPDDR5 64 GB Swap File
Storage	500 GB SSD
Operating System	Ubuntu* Desktop 22.04 LTS, Kernel 6.5
Graphics Driver	Version 24.13.29138.29
NPU Driver	Version 1.5.0
Docker*	Version 24.0.5
OpenVINO™ Toolkit	Version 2023.3 LTS
OpenVINO™ Notebooks	Version 2024.2
Ultralytics*	Version 8.1.42

8.1.1 Architecture

Figure 10 shows the text QnA as well as the object detection hardware components diagram. The user can ask questions through the microphone and receive answers through the speaker. For the object detection application, the input can be a multi-camera setup, and the detected outputs can be displayed on a monitor. The Intel® Core™ Ultra Processor supports all interfaces, including a 3.5mm speaker and microphone, Ethernet, USB, HDMI, and DP connectors. The cameras can be connected using USB or through PoE switch using Ethernet.

Figure 10. Text QnA Hardware Components Diagram

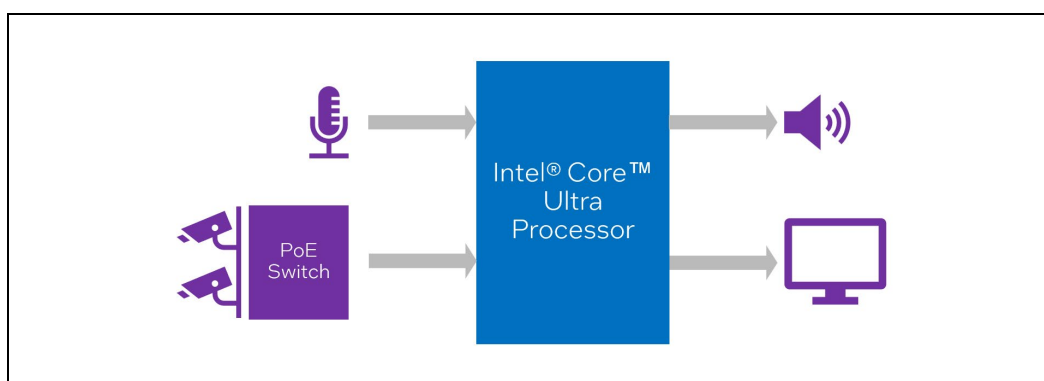


Figure 11 shows a software deployment diagram using containerization. The AI workloads run on Docker* on Ubuntu* 22.04 LTS, while the business-critical applications (for example, PoS) in the retail industry run on bare metal. The text Q&A application runs in an OpenVINO™ Toolkit container with iGPU driver pass-through. On another note, the object detection application for video surveillance analytics runs

in another OpenVINO™ container with access to NPU. Both containers can give access to the peripherals needed by the applications, like a microphone, a speaker, cameras, and a display.

Figure 11. Text QnA Software Deployment Diagram

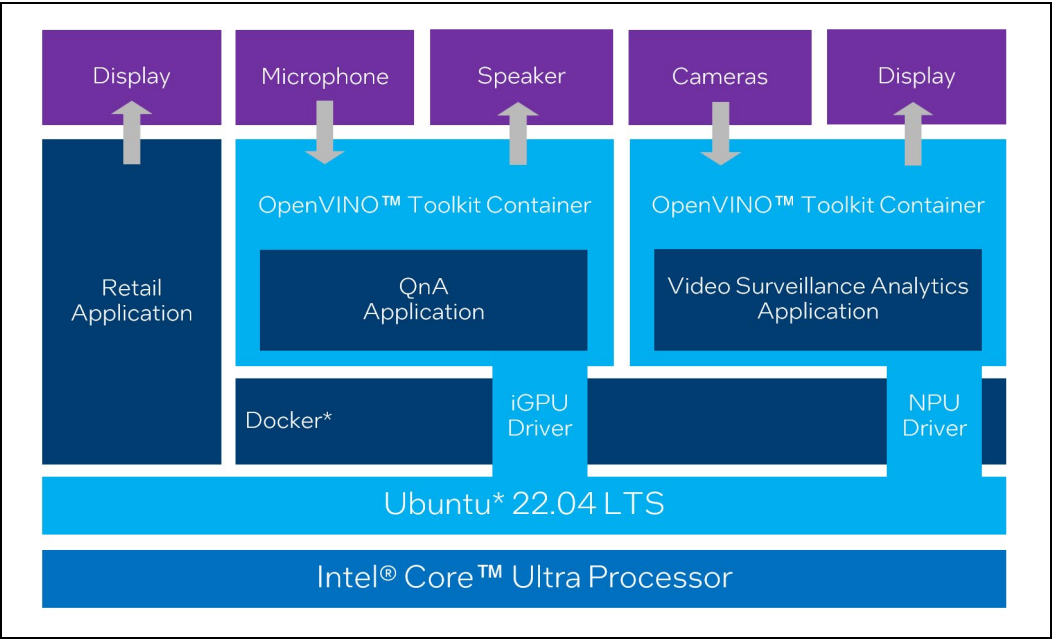
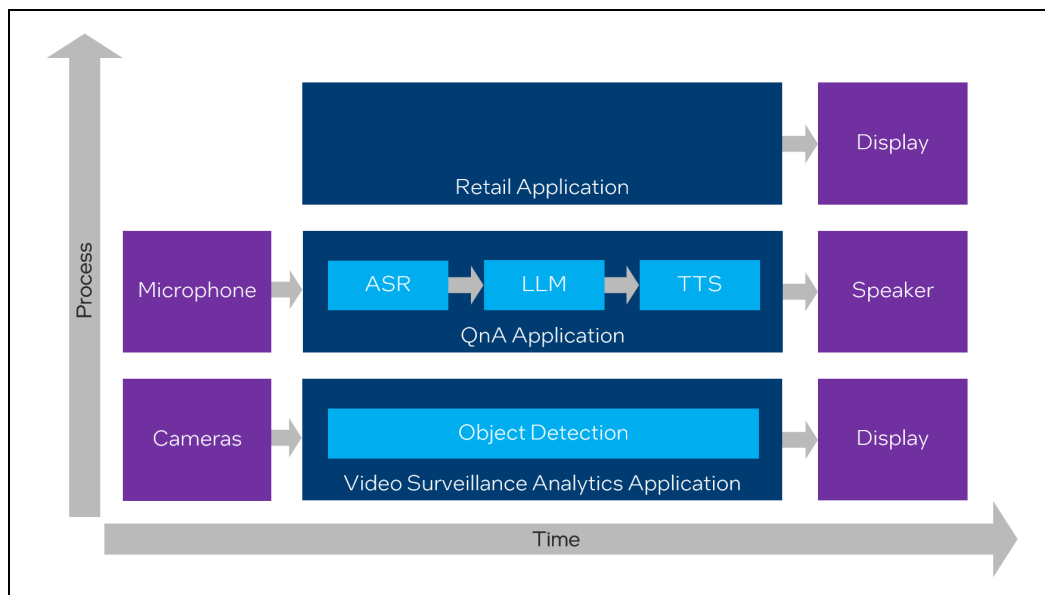


Figure 12 shows both text QnA and video surveillance analytics applications run in parallel. Whereas the components in the QnA, from ASR, LLM, to TTS run sequentially and do not utilize the computing units at the same time.

Figure 12. Text QnA Software Sequence Diagram



8.1.2 Implementation

Download Ubuntu* OS Desktop 22.04 LTS with minimal Kernel version 6.5. Perform a normal installation (not a minimal installation). Refer to the setup subsection 3.1 to install or upgrade the GPU and NPU drivers.

Cmd 70 shows the Dockerfile content to build a Docker* image with both GPU and NPU drivers. Please follow Cmd 71 and Cmd 72 to build and test run the Docker* image.

Cmd 70. Content of xPU Dockerfile

```
$ vim Dockerfile_xpu
FROM openvino/ubuntu22_dev:2023.3.0

# Install Basics
USER root
ARG DEBIAN_FRONTEND=noninteractive
RUN apt-get update -y \
    && apt-get upgrade -y
RUN apt-get update -y \
    && apt-get install -y apt-utils git curl ca-certificates bzip2 unzip
wget cmake pkg-config tree htop iotop g++ gcc libc6-dev make sudo \
    && apt-get install -y libglib2.0-0 libsm6 libxext6 libxrender-dev
python3-dev python3-opencv ffmpeg \
    && apt-get install -y libtbb12 libtbbmalloc2 \
    && apt-get install -y gpg
```

```
# Install GPU Packages
RUN wget -qO - https://repositories.intel.com/gpu/intel-graphics.key |
\
  gpg --dearmor --output /usr/share/keyrings/intel-graphics.gpg
RUN echo "deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-
graphics.gpg] https://repositories.intel.com/gpu/ubuntu jammy client" |
\
  tee /etc/apt/sources.list.d/intel-gpu-jammy.list
RUN apt-get update -y \
  && apt-get install -y \
    intel-ocl-icd intel-level-zero-gpu level-zero \
    intel-media-va-driver-non-free libmfx1 libmfxgen1 libvpl2 \
    libegl-mesa0 libegl1-mesa libegl1-mesa-dev libgbm1 libgl1-mesa-dev
libgl1-mesa-dri \
    libglapi-mesa libgles2-mesa-dev libglx-mesa0 libigdgmm12
libxatracker2 mesa-va-drivers \
    mesa-va-pau-drivers mesa-vulkan-drivers va-driver-all vainfo hwinfo
clinfo

# Install NPU Packages
RUN wget https://github.com/intel/linux-npu-
driver/releases/download/v1.5.0/intel-driver-compiler-
npu_1.5.0.20240619-9582784383_ubuntu22.04_amd64.deb
RUN dpkg -i intel-driver-compiler-npu_1.5.0.20240619-
9582784383_ubuntu22.04_amd64.deb
RUN wget https://github.com/intel/linux-npu-
driver/releases/download/v1.5.0/intel-fw-npu_1.5.0.20240619-
9582784383_ubuntu22.04_amd64.deb
RUN dpkg -i intel-fw-npu_1.5.0.20240619-
9582784383_ubuntu22.04_amd64.deb
RUN wget https://github.com/intel/linux-npu-
driver/releases/download/v1.5.0/intel-level-zero-npu_1.5.0.20240619-
9582784383_ubuntu22.04_amd64.deb
RUN dpkg -i intel-level-zero-npu_1.5.0.20240619-
9582784383_ubuntu22.04_amd64.deb

RUN apt-get remove -y level-zero
RUN wget https://github.com/oneapi-src/level-
zero/releases/download/v1.17.2/level-zero_1.17.2+u22.04_amd64.deb
RUN dpkg -i level-zero_1.17.2+u22.04_amd64.deb

# Clean Workspace
RUN rm *.deb

# Finalizing
USER openvino
WORKDIR /opt/intel/openvino
```

Cmd 71. Commands to Build xPU Docker* Image

```
$ sudo docker build \
  -f Dockerfile_xpu \
  -t openvino/ubuntu22_dev_xpu:2023.3.0 .

$ sudo docker images
REPOSITORY          TAG             ...    SIZE
openvino/ubuntu22_dev_xpu  2023.3.0       ...    5.87GB
```

Cmd 72. Commands to Test xPU Docker* Image

```
$ sudo docker run -it \
  --device=/dev/dri/renderD128 \
  --device=/dev/accel/accel0 \
  --user root \
  --rm openvino/ubuntu22_dev_xpu:2023.3.0 \

"/opt/intel/opencvino/samples/cpp/samples_bin/samples_bin/benchmark_app"
"--help"

...
Available target devices:  CPU  GNA  GPU  NPU
```

Cmd 73 shows the content of Dockerfile with OpenVINO™ Notebooks and Cmd 74 shows the steps to build the Docker* image.

Cmd 73. Content of OpenVINO™ Notebooks Dockerfile

```
$ vim Dockerfile_xpu_nb
FROM openvino/ubuntu22_dev_xpu:2023.3.0

# Install Basics
USER root
RUN pip3 install tensorflow==2.12
RUN pip3 install keras==2.12

# Install OpenVINO Open Model Zoo
RUN wget
https://github.com/openvinotoolkit/open_model_zoo/archive/refs/tags/202
3.3.0.zip \
  && unzip 2023.3.0.zip \
  && mv open_model_zoo-2023.3.0 open_model_zoo
RUN pip3 install -r open_model_zoo/demos/requirements.txt

# Install OpenVINO Notebooks
RUN git clone -b 2024.2
https://github.com/openvinotoolkit/openvino_notebooks.git
WORKDIR /opt/intel/opencvino
RUN python3 -m pip install --upgrade pip
```

```

RUN pip3 install wheel
RUN pip3 install jupyterlab ipywidgets "ipykernel>=5.0"
"ipython>=7.16.3" "setuptools<70"
RUN rm 2023.3.0.zip

# Finalizing
USER openvino
WORKDIR /opt/intel/openvino

```

Cmd 74. Commands to Build OpenVINO™ Notebooks Docker* Image

```

$ sudo docker build \
  -f Dockerfile_xpu_nb \
  -t openvino/ubuntu22_dev_xpu_nb:2023.3.0 .

$ sudo docker images

```

REPOSITORY	TAG	...	SIZE
openvino/ubuntu22_dev_xpu_nb	2023.3.0	...	11.5GB
openvino/ubuntu22_dev_xpu	2023.3.0	...	5.87GB

Since quantizing LLM models with 7 billion parameters requires about 64 GB of RAM and the process is done only once, you can use Cmd 75 to temporarily increase the RAM using Swap file. You do not need the Swap file during the model inference if you have a minimum of 32 GB RAM.

Cmd 75. Commands to Increase Memory Size using Swap File

```

$ free -m

```

	total	...
Mem:	32767	...
Swap:	0	...

```

$ sudo fallocate -l 64G /swapfile
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
Setting up swspace version 1, size = 64 GiB (68719472640 bytes)
no label, UUID=4c2fcc31-5b73-4a34-97e3-d1b8056f66b6
$ sudo swapon /swapfile
$ free -m

```

	total	...
Mem:	32767	...
Swap:	65535	...

Cmd 76 shows the steps to run the OpenVINO™ Toolkit container for text QnA application. In this case, we allocate 12 CPU threads for the application. Since a container is stateless and all modifications within it are discarded, please store the modified files in the results directory for permanent storage. It connects to both GPU (/dev/dri/renderD128) and NPU (/dev/accel/accel0). For the microphone and

speaker, connect to the sound (/dev/snd) device accordingly. You can access the notebooks using a web browser, the system IP, and port 8887.

Cmd 76. Commands to Run QnA Notebooks

```
$ mkdir results

$ sudo docker run -it \
  --cpus="12.0" \
  -p 8887:8887 \
  -v ./results:/opt/intel/opencvino/results \
  --device=/dev/dri/renderD128 \
  --device=/dev/accel/accel0 \
  --user root \
  --rm opencvino/ubuntu22_dev_xpu_nb:2023.3.0 \
  bash -c "jupyter lab --no-browser --allow-root --ip 0.0.0.0 --
port=8887"
```

The following are the sample codes for the end-to-end text Q&A application.

- ASR
 - [Automatic Speech Recognition using Distil-Whisper and OpenVINO™ Toolkit](#)
 - [Video Subtitle Generation using Whisper and OpenVINO™ Toolkit](#)
- LLM
 - [LLM Instruction-following Pipeline with OpenVINO™ Toolkit](#)
- TTS
 - [Voice Tone Cloning with OpenVoice and OpenVINO™ Toolkit](#)

Cmd 77 shows the steps to run OpenVINO™ Toolkit container for video surveillance analytics. For the application, we need about 4 CPU threads to implement 4 cameras. It connects to both GPU (/dev/dri/renderD128) and NPU (/dev/accel/accel0). For USB cameras, connect to the USB (/dev/bus/usb) devices accordingly. You can access the container via a web browser at the system IP and port 8886.

Cmd 77. Commands to Run Video Surveillance Analytics Notebooks

```
$ mkdir results

$ sudo docker run -it \
  --cpus="4.0" \
  -p 8886:8886 \
  -v ./results:/opt/intel/opencvino/results \
  --device=/dev/dri/renderD128 \
  --device=/dev/accel/accel0 \
```



```
--user root \  
--rm openvino/ubuntu22_dev_xpu_nb:2023.3.0 \  
bash -c "jupyter lab --no-browser --allow-root --ip 0.0.0.0 --  
port=8886"
```

The following is a sample code for a video surveillance analytics application using the YOLOv8 model from Ultralytics*. For OpenVINO™ toolkit version 2023.3 LTS, please pip install Ultralytics* package version 8.1.42 instead of 8.2.24.

- Object Detection
 - [Convert and Optimize YOLOv8 real-time object detection with OpenVINO™](#)

8.2 Image QnA

Intel® Core™ Ultra Processor Series 2 (Arrow Lake-H) iGPU has four times the compute capability as compared to the predecessor, Intel® Core™ Ultra Processor Series 1 (Meteor Lake-H) iGPU. Using quantized INT4 models, it can run 32-billion-parameter LLMs or 11-billion-parameter VLM models using INT8.

This subsection will cover the implementation of Visual Question Answering (VQA) using an image as input. Instead of running the VQA all the time, this implementation will use object detection to trigger the image Q&A application. The object detection will be the one running around the clock on the low-power NPU. Refer to Table 18 for the hardware and software BOMs.

Table 18. Image QnA Hardware and Software BOMs

Component	Information
Processor	Intel® Core™ Ultra 9 Processor 285H 6P+8E+2LPE-cores, 16 Threads 24M Cache, up to 5.40 GHz Maximum Memory Size 128 GB
Memory	32 GB 96 GB Swap File
Storage	500 GB SSD
Operating System	Ubuntu* Desktop 24.04 LTS, Kernel linux-oem-24.04b
Graphics Driver	Version 24.52.32224.5
NPU Driver	Version 1.13.0
Docker* Software	Version 28.0.1
OpenVINO™ Toolkit	Version 2025.0

Component	Information
OpenVINO™ Notebooks	Version 2025.0 OpenVINO™ Notebooks 2025.0 has up to YOLOv11 models only. For YOLOv12 models, please use version 2025.1.

8.2.1 Architecture

Figure 13 shows the image QnA hardware components diagram with the required peripherals. Object detection gets input images from the cameras. When it detects a certain object, it will display the object detected and trigger the image QnA application. The image QnA application takes the image and performs VQA with the prompt input from a microphone. The application then provides the output through a speaker. The Intel® Core™ Ultra Processor Series 2 supports peripherals via USB, Ethernet, a 3.5mm microphone and speaker, HDMI, and DP connectors. The cameras can be directly connected using USB or through PoE switch using Ethernet connector.

Figure 14 shows the VQA application in a containerized manner using Docker*, whereas the business-critical application runs on bare metal. By doing so, AI applications, as additional features, are unable to access information from the business-critical application. We can also easily control the resources used by the containers.

The VQA consists of image QnA as well as object detection applications. The end-to-end image QnA application also includes ASR, VLM, and TTS. The applications need access to both GPU and NPU as well as other peripherals like cameras, microphones, and speakers.

Figure 13. Image QnA Hardware Components Diagram

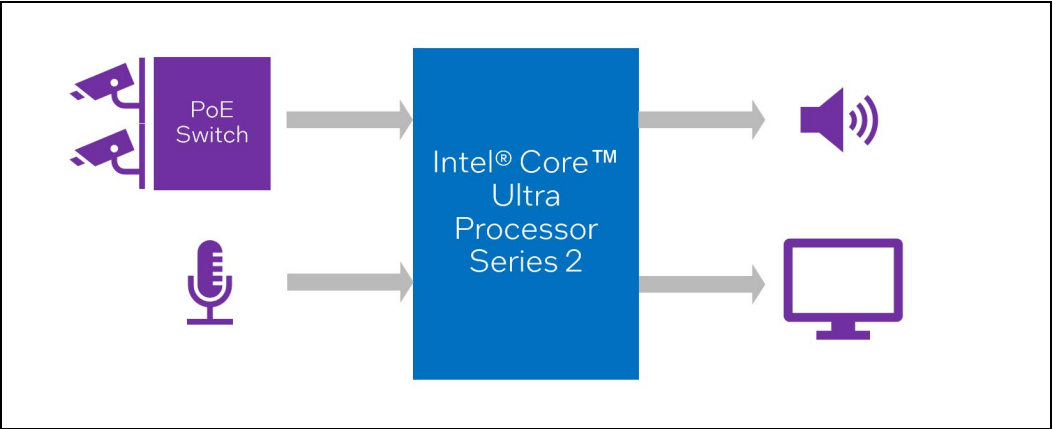


Figure 14. Image QnA Software Deployment Diagram

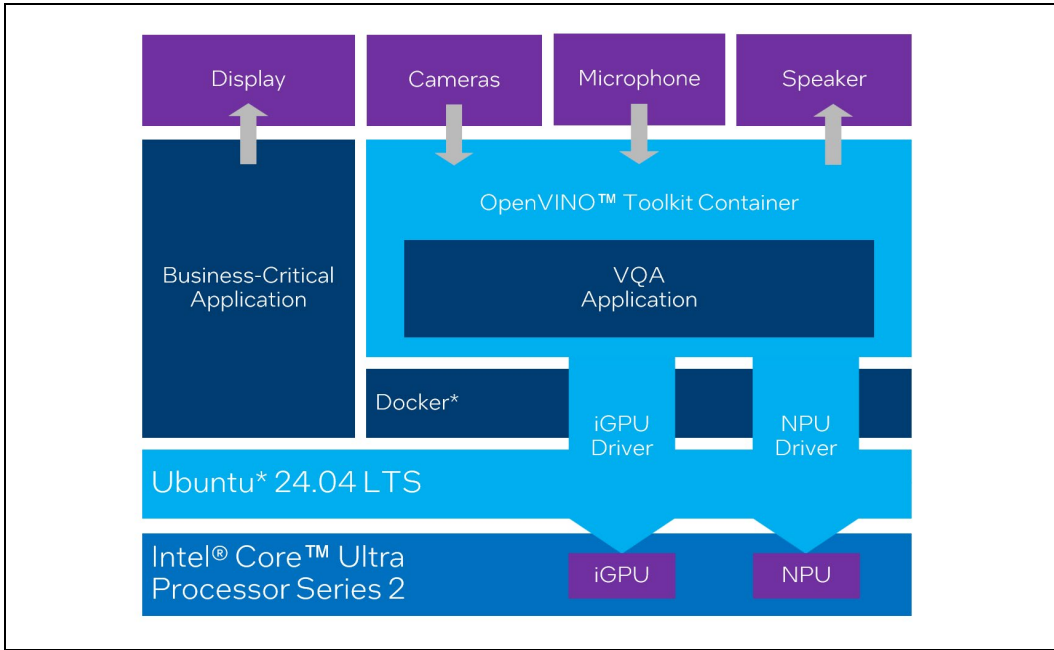
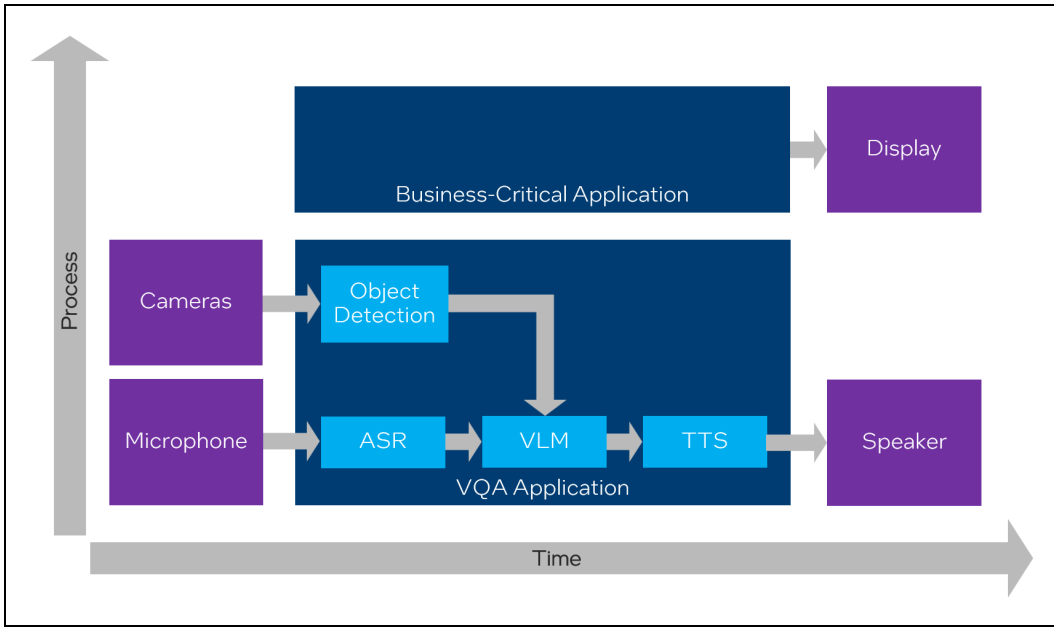


Figure 15 shows the process sequence diagram in which the business-critical application is running in parallel to the VQA application, whereas some components in the VQA application like ASR, VLM, and TTS are running sequentially.

Figure 15. Image QnA Software Sequence Diagram



8.2.2 Implementation

For this implementation, we will use [Intel® Ubuntu* packages](#) and [Ubuntu* OEM Kernel](#). It is for OS vendors to test and integrate new processor functionalities from Intel. If you are not an OS vendor, please use the kernel, drivers, and packages recommended by the OS vendors.

Download Ubuntu* Desktop 24.04 LTS with minimal Kernel linux-oem-24.04b. Please perform a normal installation (not a minimal installation). Cmd 78 shows the steps to install the required graphics and development packages. Cmd 79 shows the steps to upgrade to the minimum required Linux* kernel version.

Cmd 78. Commands to Install Required Packages

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install \
    git, clinfo, curl, wget, gpg-agent, libtbb12, quilt, \
    libssl-dev, kernel-wedge, liblz4-tool, libelf-dev, \
    flex, bison, python3-pip, python3-venv, ca-certificates, \
    vim
```

Cmd 79. Commands to Upgrade Linux* Kernel

```
$ vim /etc/apt/sources.list.d/intel-arl.list
deb https://download.01.org/intel-linux-overlay/ubuntu noble main
non-free multimedia kernels
deb-src https://download.01.org/intel-linux-overlay/ubuntu noble main
non-free multimedia kernels
```

```
$ sudo wget https://download.01.org/intel-linux-
overlay/ubuntu/E6FA98203588250569758E97D176E3162086EE4C.gpg -O
/etc/apt/trusted.gpg.d/ar1.gpg
```

```
$ vim /etc/apt/preferences.d/intel-ar1
Package: *
Pin: release o=intel-iot-linux-overlay-noble
Pin-Priority: 2000
```

```
$ sudo apt update
$ sudo apt install linux-oem-24.04b
```

Update GRUB_CMDLINE_LINUX_DEFAULT in the /etc/default/grub file with the following value. Then, follow Cmd 80 to update GRUB and reboot the system.

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash i915.enable_guc=3
i915.max_vfs=7 i915.force_probe=* udmabuf.list_limit=8192"
```

Cmd 80. Commands to Update GRUB

```
$ sudo update-grub
$ sudo reboot now
```

Cmd 81 show the steps to install Docker* and Cmd 82 shows optional steps if your system runs behind the firewall.

Cmd 81. Commands to Install Docker* Computer Software

```
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc

$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-
  by=/etc/apt/keyrings/docker.asc]
  https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

$ sudo apt update
$ sudo apt install \
  docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin
```

Cmd 82. Commands to Run Docker* with Proxy Setting

```
$ sudo mkdir -p /etc/systemd/system/docker.service.d

$ vim /etc/systemd/system/docker.service.d/http-proxy.conf
[Service]
Environment=HTTP_PROXY=$HTTP_PROXY
Environment=HTTPS_PROXY=$HTTPS_PROXY
Environment=FTP_PROXY=$FTP_PROXY
Environment=SOCKS_PROXY=$SOCKS_PROXY
Environment=NO_PROXY=$NO_PROXY

$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

Cmd 83 and Cmd 84 show the steps to install both GPU and NPU drivers using [GPU version 24.52.32224.5](#) and [NPU version 1.13.0](#) respectively.

Cmd 83. Commands to Install GPU Driver

```
$ wget -qO - https://repositories.intel.com/gpu/intel-graphics.key | \
  sudo gpg --yes --dearmor --output /usr/share/keyrings/intel-
  graphics.gpg
```

```

$ echo "deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-
graphics.gpg] https://repositories.intel.com/gpu/ubuntu noble client" |
\
  sudo tee /etc/apt/sources.list.d/intel-gpu-noble.list
$ sudo apt update
$ sudo apt install \
  libze1, intel-level-zero-gpu, intel-ocl-icd, \
  intel-gsc, clinfo, vainfo, hwinfo, linux-firmware

$ mkdir -p /tmp/neo_gpu
$ cd /tmp/neo_gpu
$ wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-core-
2_2.5.6+18417_amd64.deb
$ wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-ocl-
2_2.5.6+18417_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-gpu-
dbgsym_1.6.32224.5_amd64.ddeb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-
gpu_1.6.32224.5_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-ocl-icd-
dbgsym_24.52.32224.5_amd64.ddeb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-ocl-
icd_24.52.32224.5_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/libigdgmm12_22.5.5_amd64.deb
$ wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/ww52.sum
$ sha256sum -c ww52.sum
$ sudo dpkg -i ./*.deb

```

Cmd 84. Commands to Install NPU Driver

```

$ sudo dpkg --purge --force-remove-reinstreq \
  intel-driver-compiler-npu intel-fw-npu intel-level-zero-npu
$ sudo apt install --fix-broken
$ sudo apt update

$ mkdir -p /tmp/neo_npu
$ cd /tmp/neo_npu
$ wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-driver-compiler-
npu_1.13.0.20250131-13074932693_ubuntu24.04_amd64.deb

```

```
$ wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-fw-npu_1.13.0.20250131-
13074932693_ubuntu24.04_amd64.deb
$ wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-level-zero-npu_1.13.0.20250131-
13074932693_ubuntu24.04_amd64.deb
$ wget https://github.com/oneapi-src/level-
zero/releases/download/v1.18.5/level-zero_1.18.5+u24.04_amd64.deb
$ sudo dpkg -i ./*.deb
```

Cmd 85 shows the content of OpenVINO™ Dockerfile that can access both GPU and NPU. Cmd 86 shows the script to install NPU driver.

Cmd 85. Content of OpenVINO™ Toolkit xPU Dockerfile

```
$ vim Dockerfile_xpu
FROM openvino/ubuntu24_dev:2025.0.0

USER root
RUN apt-get update && \
    apt-get install -y \
        wget gnupg curl libtbbmalloc2 libtbb12 kmod ffmpeg

# Install GPU Driver
RUN wget -qO - https://repositories.intel.com/gpu/intel-graphics.key
| \
    gpg --yes --dearmor --output /usr/share/keyrings/intel-
graphics.gpg && \
    echo "deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-
graphics.gpg] https://repositories.intel.com/gpu/ubuntu noble client" |
\
    tee /etc/apt/sources.list.d/intel-gpu-noble.list
RUN apt update && \
    apt-get install -y \
        libze1 intel-opencl-icd clinfo intel-gsc clinfo \
        vainfo hwinfo

RUN mkdir -p /tmp/neo_gpu
WORKDIR /tmp/neo_gpu
RUN wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-core-
2_2.5.6+18417_amd64.deb && \
    wget https://github.com/intel/intel-graphics-
compiler/releases/download/v2.5.6/intel-igc-opencl-
2_2.5.6+18417_amd64.deb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-gpu-
dbgsym_1.6.32224.5_amd64.ddeb && \
```

```
wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-level-zero-
gpu_1.6.32224.5_amd64.deb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-openccl-icd-
dbgsym_24.52.32224.5_amd64.ddeb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/intel-openccl-
icd_24.52.32224.5_amd64.deb && \
    wget https://github.com/intel/compute-
runtime/releases/download/24.52.32224.5/libigdgmm12_22.5.5_amd64.deb
RUN dpkg -i ./*.deb

# Install NPU Driver
COPY npu_container.sh /tmp/neo_npu/npu_container.sh
WORKDIR /tmp/neo_npu/
RUN bash npu_container.sh
# Finalizing
USER openvino
WORKDIR /home/openvino
```

Cmd 86. Content of NPU Driver Script File for Docker*

```
$ vim npu_container.sh
    wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-driver-compiler-
npu_1.13.0.20250131-13074932693_ubuntu24.04_amd64.deb
    wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-fw-npu_1.13.0.20250131-
13074932693_ubuntu24.04_amd64.deb
    wget https://github.com/intel/linux-npu-
driver/releases/download/v1.13.0/intel-level-zero-npu_1.13.0.20250131-
13074932693_ubuntu24.04_amd64.deb
    wget https://github.com/oneapi-src/level-
zero/releases/download/v1.18.5/level-zero_1.18.5+u24.04_amd64.deb
    dpkg -i ./*.deb
```

Cmd 87 shows the steps to build the OpenVINO™ Docker* image and Cmd 88 shows the steps to test-run it. The commands to build and run the Docker* image include the proxy parameters for secure systems running behind a firewall.

Cmd 87. Commands to Build OpenVINO™ Toolkit xPU Docker* Image

```
$ sudo docker build --progress=plain --no-cache \
    --build-arg HTTP_PROXY="$HTTP_PROXY" \
    --build-arg HTTPS_PROXY="$HTTPS_PROXY" \
    --build-arg FTP_PROXY="$FTP_PROXY" \
    --build-arg SOCKS_PROXY="$SOCKS_PROXY" \
    --build-arg NO_PROXY="$NO_PROXY" \
    -t openvino/ubuntu24_dev_xpu:2025.0.0 -f Dockerfile_xpu .
```

Cmd 88. Commands to Test OpenVINO™ Toolkit xPU Docker* Image

```
$ sudo docker run -it \
  --device=/dev/dri/renderD128 \
  --device=/dev/accel/accel0 \
  --user root \
  -e HTTP_PROXY="$HTTP_PROXY" \
  -e HTTPS_PROXY="$HTTPS_PROXY" \
  -e FTP_PROXY="$FTP_PROXY" \
  -e SOCKS_PROXY="$SOCKS_PROXY" \
  -e NO_PROXY="$NO_PROXY" \
  --rm openvino/ubuntu24_dev_xpu:2025.0.0 \
  "/opt/intel/openvino/samples/cpp/samples_bin/samples_bin/benchmark_app"
"--help"
```

...

Available target devices: CPU GPU NPU

Cmd 89 shows the content of OpenVINO™ Notebooks Dockerfile and Cmd 90 shows the steps to build it.

Cmd 89. Content of OpenVINO™ Toolkit Notebooks Dockerfile

```
$ vim Dockerfile_xpu_nb
FROM openvino/ubuntu24_dev_xpu:2025.0.0

USER openvino
WORKDIR /home/openvino
RUN git clone --depth=1 -b 2025.0
https://github.com/openvinotoolkit/openvino_notebooks.git

WORKDIR /home/openvino/openvino_notebooks
USER root
RUN python3 -m pip install --upgrade pip
RUN pip3 install wheel setuptools
RUN pip3 install -r requirements.txt

# Finalizing
USER openvino
WORKDIR /home/openvino
```

Cmd 90. Commands to Build OpenVINO™ Toolkit Notebooks Docker* Image

```
$ sudo docker build --progress=plain --no-cache \
  --build-arg HTTP_PROXY="$HTTP_PROXY" \
  --build-arg HTTPS_PROXY="$HTTPS_PROXY" \
  --build-arg FTP_PROXY="$FTP_PROXY" \
  --build-arg SOCKS_PROXY="$SOCKS_PROXY" \
  --build-arg NO_PROXY="$NO_PROXY" \
```

```
-t openvino/ubuntu24_dev_xpu_nb:2025.0.0 -f Dockerfile_xpu_nb .
```

```
$ sudo docker images
REPOSITORY          TAG      ...  SIZE
openvino/ubuntu24_dev_xpu_nb  2025.0.0  ...  4.21GB
openvino/ubuntu24_dev_xpu    2025.0.0  ...  3.37GB
```

VLM model with 11 billion parameters needs about 100 GB RAM during the INT8 quantization process, whereas for the inference, the model needs only 32 GB of RAM. As the quantization will be done only once, we will use a Swap file to increase the 32 GB system memory by 96 GB to 128 GB. Later, we can remove the Swap file during the inference. Please find Cmd 91 to do so.

Cmd 91. Commands to Increase Memory Capacity using Swap File

```
$ free -m
              total
Mem:          32...
Swap:         0

$ sudo fallocate -l 96G /swapfile
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
Setting up swspace version 1, size = 96 GiB ...
$ sudo swapon /swapfile

$ free -m
              total
Mem:          32...
Swap:         96...
```

Cmd 92 shows the steps to run the notebooks for all the VQA components (object detection, ASR, VLM, and TTS). As a container is stateless, please store the new or modified files in the `results` directory. The container has 6 CPU threads and connects to both GPU (`/dev/dri/renderD128`) as well as NPU (`/dev/accel/accel0`). For USB cameras, microphone, and speaker, connect to the USB (`/dev/bus/usb`) and sound (`/dev/snd`) devices accordingly. Please use a web browser, the system IP, and port 8888 to access the notebooks.

Cmd 92. Commands to Run VQA Notebooks

```
$ mkdir results

$ sudo docker run -it \
  --cpus="6.0" \
  -p 8888:8888 \
  -v ./results:/opt/intel/openvino/results \
  --device=/dev/dri/renderD128 \
  --device=/dev/accel/accel0 \
```

```
--user root \  
-e HTTP_PROXY="$HTTP_PROXY" \  
-e HTTPS_PROXY="$HTTPS_PROXY" \  
-e FTP_PROXY="$FTP_PROXY" \  
-e SOCKS_PROXY="$SOCKS_PROXY" \  
-e NO_PROXY="$NO_PROXY" \  
--rm openvino/ubuntu24_dev_xpu_nb:2025.0.0 \  
bash -c "jupyter lab --no-browser --allow-root --ip 0.0.0.0 --  
port=8888"
```

Please find the following sample notebooks for the VQA application.

- Object Detection
 - [Convert and Optimize YOLOv12 with OpenVINO™ Toolkit](#)
Figure 16 and Figure 17 show object detection sample using YOLO12s model running on NPU as the low-power streaming accelerator. Figure 18 and Figure 19 show the input and output images.
- ASR
 - [Automatic speech recognition using Distil-Whisper and OpenVINO™ Toolkit](#)
 - [Video Subtitle Generation with OpenAI* Whisper and OpenVINO™ Toolkit Generate API](#)
- VLM
 - [Visual-language assistant with Llama-3.2-11B-Vision and OpenVINO™](#)
Figure 20 shows the VLM target accelerator using iGPU. Figure 22 shows the VLM input prompt and the generated output. Due to the maximum memory limitation of Arrow Lake-H (128 GB), please remove Scale Estimation and use only AWQ for the quantization method as shown in Figure 21.
- TTS
 - [Voice tone cloning with OpenVoice and OpenVINO™ Toolkit](#)

Figure 16. Object Detection Model Selection

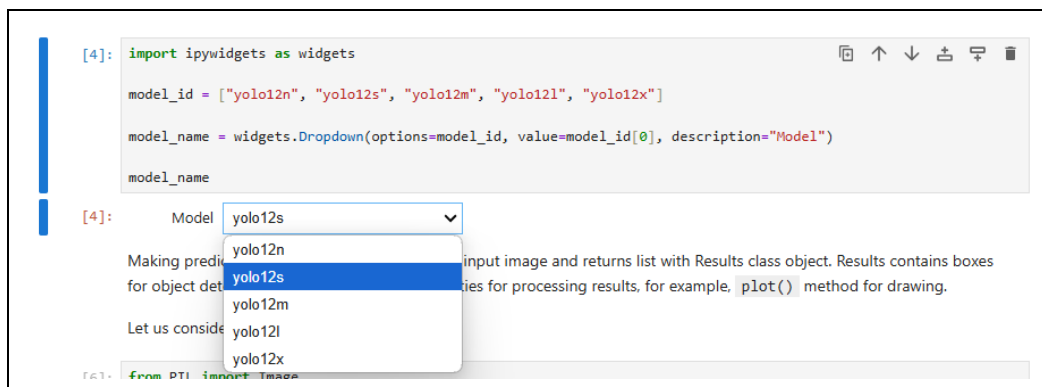


Figure 17. Object Detection Compute Unit Selection

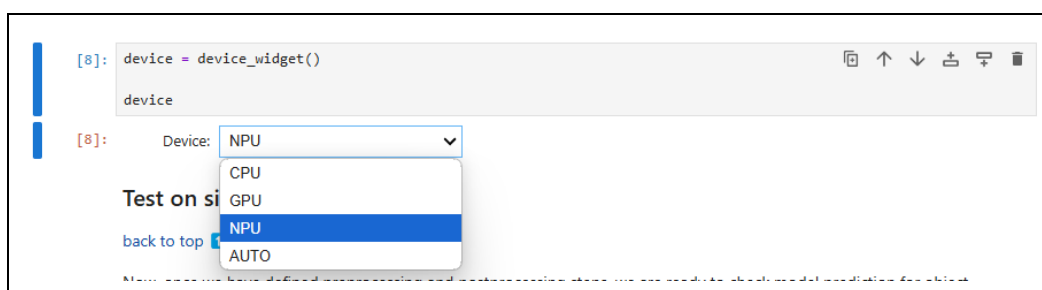


Figure 18. Object Detection and VLM Input Image



Figure 19. Object Detection Output Image

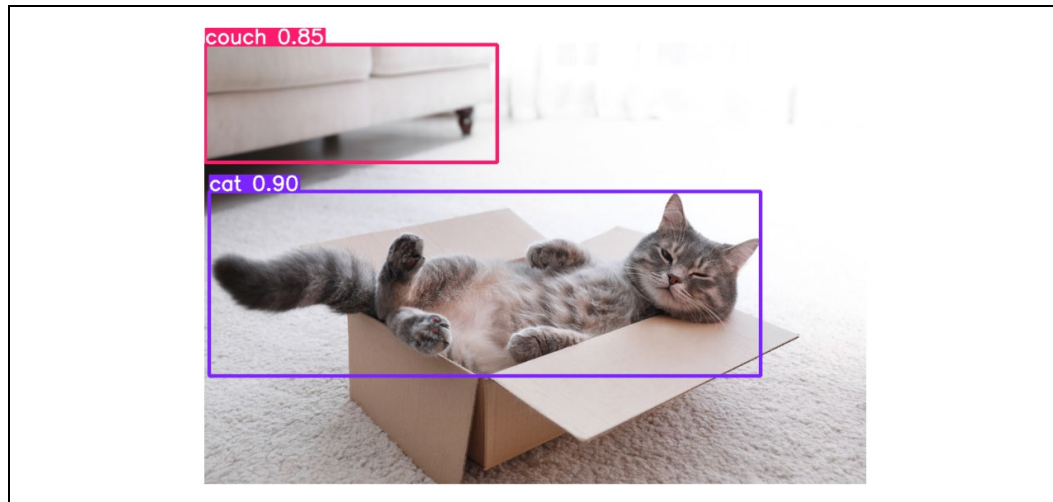


Figure 20. Image QnA Compute Unit Selection

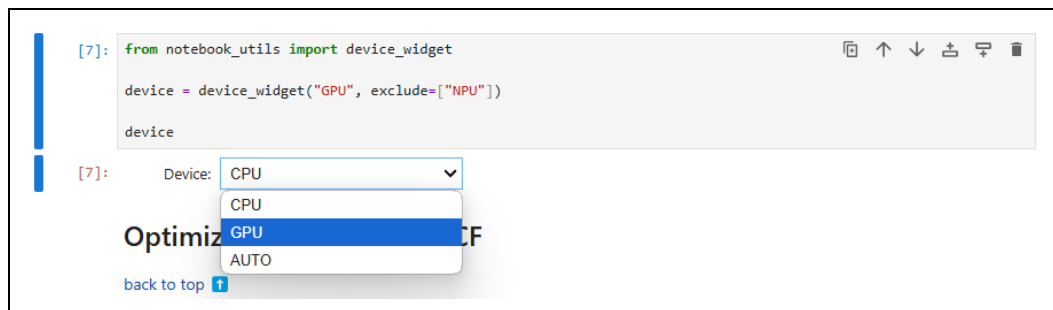


Figure 21. Image QnA Model Quantization Selection

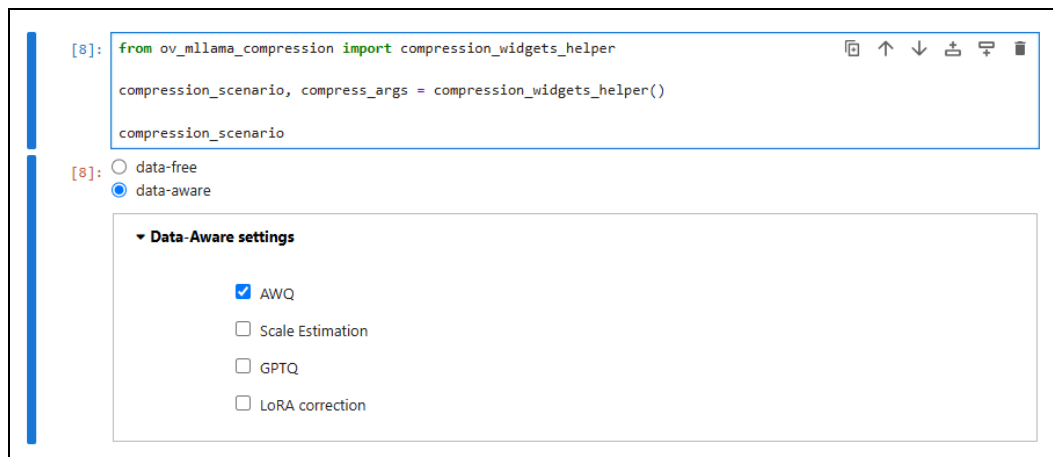


Figure 22. Image QnA Output Text



8.3

Video QnA

The Intel® Core™ Ultra Processor Series 3 (Panther Lake-H) iGPU has about 60% more capability than its predecessor, the Intel® Core™ Ultra Processor Series 2 (Arrow Lake-H) iGPU. With additional compute power, we will use video input for the VQA application rather than an image. Using a static image, the AI can only tell if a person is holding an item. The benefit of using a video is that the AI can now tell if a person is picking up, putting down, or just carrying an item.

Instead of running the VQA around the clock, we can use object detection running on a low-power NPU to trigger the video QnA application running on a GPU. The sample video will be about a 10-second interval time. Please find Table 19 for the hardware and software BOMs.

Table 19. Video QnA Hardware and Software BOMs

Component	Information
Processor	Intel® Core™ Ultra X7 Processor 358H 4P+8E+4LPE-cores, 16 Threads 18M Cache, up to 4.80 GHz Maximum Memory Size 96 GB
Memory	64 GB

Component	Information
Storage	500 GB SSD
Operating System	Ubuntu* Desktop 24.04 LTS, Kernel 6.17-intel
Graphics Driver	Version 25.48.36300.8
NPU Driver	Version 1.28.0
OpenVINO™ Toolkit	Version 2025.3
OpenVINO™ Notebooks	Version 2025.3

9.1.1 Architecture

Figure 23 shows the video QnA hardware components diagram with the input and output components. An object detection application receives input from cameras to detect objects of interest. Using video recording applications, the video can then be used as the input to the video QnA application together with the prompt from the microphone. The QnA provides the output through a speaker.

The Intel® Core™ Ultra Processor Series 3 supports hardware components via USB, Ethernet, a 3.5mm microphone and speaker, HDMI, and DP connectors. The cameras can be directly connected using USB or through a PoE switch using an Ethernet connector.

Figure 23. Video QnA Hardware Components Diagram

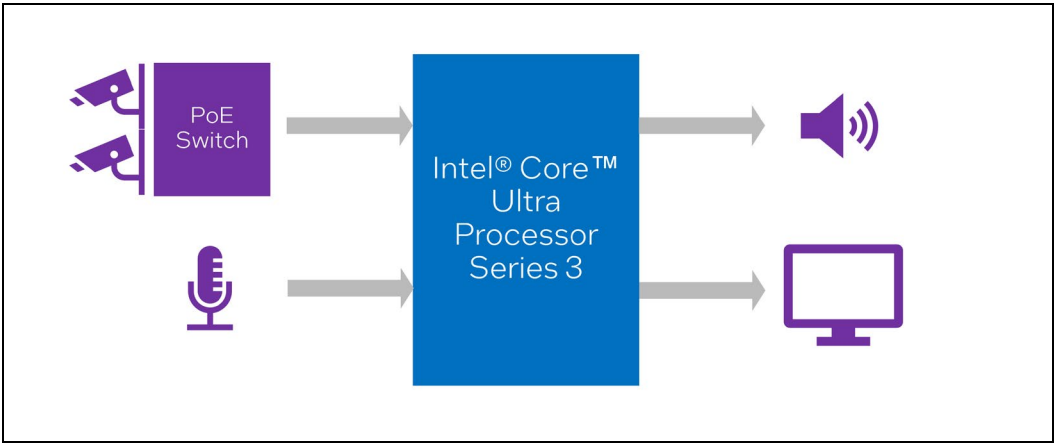


Figure 24 shows the VQA application running solely on bare metal. The VQA consists of video QnA as well as object detection applications. The end-to-end video QnA application has ASR, VLM, and TTS components. The applications have access to both GPUs and NPUs, as well as other peripherals such as cameras, a microphone, a speaker, and a display.

Figure 24. Video QnA Software Deployment Diagram

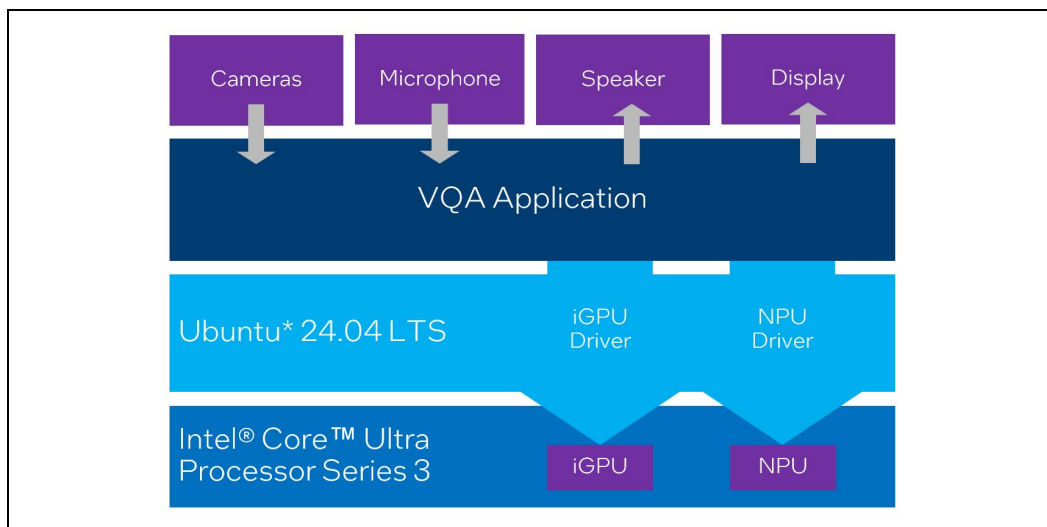
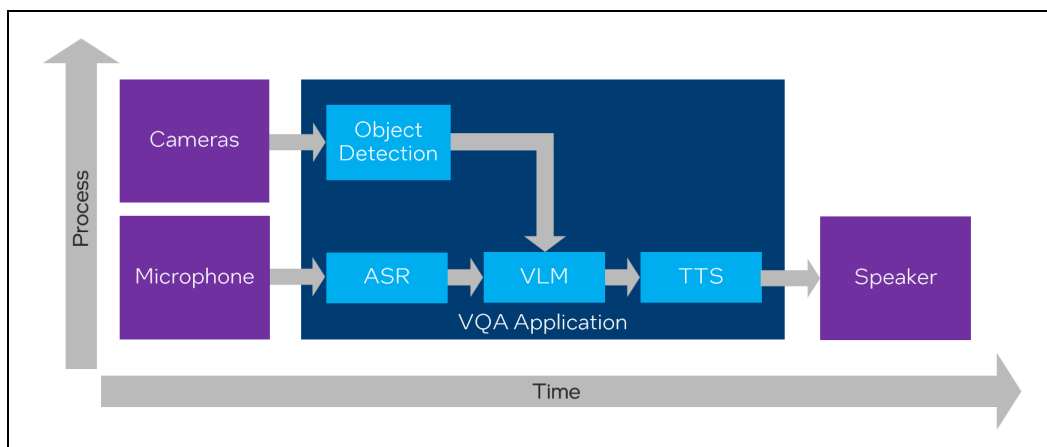


Figure 25 shows the process sequence diagram in which the components in the VQA application, like ASR, VLM, and TTS, are running sequentially.

Figure 25. Video QnA Software Sequence Diagram



9.1.2 Implementation

For this implementation, we will use both [Intel® Ubuntu* packages](#) and [Intel® Kernel](#), as well as an option to compile the Kernel. It is for OS vendors to test and integrate new Intel® processor features. If you are not an OS vendor, please use the kernel, drivers, and packages recommended by the OS vendors.

Download Ubuntu* OS Desktop 24.04 LTS. Please perform a normal installation (not a minimal installation). Cmd 93 shows the steps to install Intel® packages.

Cmd 93. Commands to Install the Required Packages

```
$ sudo vim /etc/apt/sources.list.d/intel-ptl.list
deb https://download.01.org/intel-linux-overlay/ubuntu noble main non-
free multimedia kernels
deb-src https://download.01.org/intel-linux-overlay/ubuntu noble main
non-free multimedia kernels

$ sudo wget https://download.01.org/intel-linux-
overlay/ubuntu/E6FA98203588250569758E97D176E3162086EE4C.gpg -O
/etc/apt/trusted.gpg.d/ptl.gpg

$ sudo vim /etc/apt/preferences.d/intel-ptl
Package: *
Pin: release o=intel-iot-linux-overlay-noble
Pin-Priority: 2000

$ sudo apt update
$ export DEBIAN_FRONTEND=noninteractive
$ sudo apt install \
libigfxcmt-dev libigfxcmt7 xdp-tools nano ocl-icd-libopencl1 \
curl openssh-server net-tools gir1.2-gst-plugins-bad-1.0 \
gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 \
gir1.2-gst-rtsp-server-1.0 gstreamer1.0-alsa gstreamer1.0-gl \
gstreamer1.0-gtk3 gstreamer1.0-opencv gstreamer1.0-plugins-bad \
gstreamer1.0-plugins-bad-apps gstreamer1.0-plugins-base \
gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-good \
gstreamer1.0-plugins-ugly gstreamer1.0-pulseaudio \
gstreamer1.0-qt5 gstreamer1.0-rtsp gstreamer1.0-tools \
gstreamer1.0-x intel-media-va-driver-non-free libdrm-amdgpu1 \
libdrm-common libdrm-dev libdrm-intel1 libdrm-nouveau2 \
libdrm-radeon1 libdrm-tests libdrm2 libgstrtspserver-1.0-dev \
libgstrtspserver-1.0-0 libgstreamer-glib1.0-0 \
libgstreamer-opencv1.0-0 libgstreamer-plugins-bad1.0-0 \
libgstreamer-plugins-bad1.0-dev libgstreamer-plugins-base1.0-0 \
libgstreamer-plugins-base1.0-dev libgstreamer-plugins-good1.0-0 \
libgstreamer-plugins-good1.0-dev libgstreamer1.0-0 \
libgstreamer1.0-dev libigdgmm-dev libigdgmm12 libmfx-gen1.2 \
libtpms-dev libtpms0 libva-dev libva-drm2 libva-glx2 \
libva-wayland2 libva-x11-2 libva2 libwayland-bin \
libwayland-client0 libwayland-cursor0 libwayland-dev \
libwayland-doc libwayland-egl-backend-dev libwayland-egl1 \
libwayland-server0 libxatracker2 linux-firmware mesa-utils \
mesa-va-drivers mesa-vdpau-drivers mesa-vulkan-drivers \
libvpl-dev libvpl-tools libmfx-gen-dev onevpl-tools ovmf \
ovmf-ia32 qemu-block-extra qemu-guest-agent qemu-system \
qemu-system-arm qemu-system-common qemu-system-data \
qemu-system-gui qemu-system-mips qemu-system-misc \
qemu-system-ppc qemu-system-s390x qemu-system-sparc \
```

```

qemu-system-x86 qemu-user qemu-user-binfmt qemu-utils \
va-driver-all vainfo weston xserver-xorg-core libvirt0 \
libvirt-clients libvirt-daemon libvirt-daemon-config-network \
libvirt-daemon-config-nwfilter libvirt-daemon-driver-lxc \
libvirt-daemon-driver-qemu \
libvirt-daemon-driver-storage-gluster \
libvirt-daemon-driver-storage-iscsi-direct \
libvirt-daemon-driver-storage-rbd \
libvirt-daemon-driver-storage-zfs libvirt-daemon-driver-vbox \
libvirt-daemon-driver-xen libvirt-daemon-system \
libvirt-daemon-system-systemd libvirt-dev libvirt-doc \
libvirt-login-shell libvirt-sanlock libvirt-wireshark \
libnss-libvirt swtpm swtpm-tools bmap-tools adb autoconf \
automake libtool cmake g++ gcc git intel-gpu-tools libssl3 \
libssl-dev make mosquitto mosquitto-clients build-essential \
apt-transport-https default-jre docker-compose ffmpeg git-lfs \
gnuplot lbzip2 libglew-dev libglm-dev libsdl2-dev mc openssl \
pciutils python3-pandas python3-pip python3-seaborn \
terminator wmctrl wayland-protocols gdbserver ethtool iperf3 \
msr-tools powertop linuxptp lsscsi tpm2-tools tpm2-abrmd \
binutils cifs-utils i2c-tools xdotool gnupg lsb-release \
iproute2 qemu-system-modules-opengl socat virt-viewer \
spice-client-gtk util-linux-extra dbus-x11 sg3-utils rpm \
-y --allow-downgrades

```

For the Kernel, please select one of Cmd 94 or Cmd 95 to either manually compile or use pre-compiled Intel® Kernel.

Cmd 94. Commands to Compile and Install Linux* OS Kernel

```

$ sudo apt install \
  quilt libssl-dev kernel-wedge liblz4-tool libelf-dev flex \
  bison git libdw-dev

$ mkdir neo_kernel_intel
$ cd neo_kernel_intel
$ sudo git clone \
  -b "mainline-tracking/v6.17-ubuntu" \
  https://github.com/intel/linux-kernel-overlay.git
$ cd linux-kernel-overlay
$ sudo vim build.sh

...
    KERNELRELEASE="6.17-intel"
    nice make -j"${nproc}" bindeb-pkg \
      LOCALVERSION= \
      KDEB_PKGVERSION="${KSRC_UPSTREAM_TAG:1}"-"$build_id" \
      KERNELRELEASE="6.17-intel" \
      KDEB_SOURCENAME=linux-"${KERNELRELEASE}"
...

```

```
$ sudo ./build.sh

$ ls -l
...
... 10013474 ... linux-headers-6.17-intel_6.17-0_amd64.deb
... 103961378 ... linux-image-6.17-intel_6.17-0_amd64.deb
... 1448723852 ... linux-image-6.17-intel-dbg_6.17-0_amd64.deb
... 1501132 ... linux-libc-dev_6.17-0_amd64.deb
...

$ sudo dpkg -i linux-image-6.17-intel_6.17-0_amd64.deb
$ sudo dpkg -i linux-headers-6.17-intel_6.17-0_amd64.deb
```

Cmd 95. Commands to Upgrade Linux* OS Kernel

```
$ sudo apt install linux-image-6.17-intel
$ sudo apt install linux-headers-6.17-intel
```

Cmd 96 shows the steps to update the GRUB. Please restart the system afterward and check if the new Kernel version has been reflected.

Cmd 96. Commands to Update GRUB

```
$ sudo vim /etc/default/grub
...
GRUB_CMDLINE_LINUX="quiet splash xe.max_vfs=7 xe.force_probe=*
modprobe.blacklist=i915 udmabuf.list_limit=8192"
...

$ sudo grep submenu /boot/grub/grub.cfg
... 'Advanced options for Ubuntu' ...
$ sudo grep 6.17 /boot/grub/grub.cfg
... 'Ubuntu, with Linux 6.17-intel' ...
$ sudo vim /etc/default/grub
...
GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux 6.17-
intel"
...

$ sudo update-grub
$ sudo reboot now

$ uname -a
Linux user 6.17-intel ...
```

Cmd 97 and Cmd 98 show the steps to install both GPU and NPU drivers using [GPU version 25.48.36300.8](#) and [NPU version 1.28.0](#) respectively.

Cmd 97. Commands to Install GPU Driver

```
$ mkdir neo_gpu
$ cd neo_gpu

$ wget https://github.com/intel/intel-graphics-
  compiler/releases/download/v2.24.8/intel-igc-core-
  2_2.24.8+20344_amd64.deb
$ wget https://github.com/intel/intel-graphics-
  compiler/releases/download/v2.24.8/intel-igc-openc-
  2_2.24.8+20344_amd64.deb
$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/intel-ocloc-
  dbgsym_25.48.36300.8-0_amd64.ddeb
$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/intel-ocloc_25.48.36300.8-
  0_amd64.deb
$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/intel-openc-icd-
  dbgsym_25.48.36300.8-0_amd64.ddeb
$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/intel-openc-icd_25.48.36300.8-
  0_amd64.deb
$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/libigdgmm12_22.8.2_amd64.deb
$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/libze-intel-gpu1-
  dbgsym_25.48.36300.8-0_amd64.ddeb
$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/libze-intel-gpu1_25.48.36300.8-
  0_amd64.deb

$ wget https://github.com/intel/compute-
  runtime/releases/download/25.48.36300.8/ww48.sum
$ sha256sum -c ww48.sum
intel-ocloc-dbgsym_25.48.36300.8-0_amd64.ddeb: OK
intel-ocloc_25.48.36300.8-0_amd64.deb: OK
intel-openc-icd-dbgsym_25.48.36300.8-0_amd64.ddeb: OK
intel-openc-icd_25.48.36300.8-0_amd64.deb: OK
libigdgmm12_22.8.2_amd64.deb: OK
libze-intel-gpu1-dbgsym_25.48.36300.8-0_amd64.ddeb: OK
libze-intel-gpu1_25.48.36300.8-0_amd64.deb: OK

$ sudo dpkg -i *.deb
$ sudo apt install ocl-icd-libopenc-1
$ sudo apt --fix-broken install

$ sudo add-apt-repository ppa:kobuk-team/intel-graphics
$ sudo apt update
```

```
$ sudo apt install \
intel-media-va-driver-non-free libigdgmm12 libigc2 clinfo
$ sudo apt install \
vainfo libvpl2 libvpl-tools libmetee4 \
intel-metrics-discovery intel-metrics-library
$ sudo apt install \
libmfx-gen1 libtbb12 libtbbmalloc2 intel-opencl-icd \
libze-intel-gpu1
$ sudo apt install intel-ocloc

$ sudo usermod -a -G render user
$ groups user
user : ... render ...

$ sudo reboot now

$ clinfo
Number of platforms      1
Platform Name            Intel(R) OpenCL Graphics
Platform Vendor          Intel(R) Corporation
Platform Version         OpenCL 3.0
...
```

Cmd 98. Commands to Install NPU Driver

```
$ sudo apt update && sudo apt install dkms
$ sudo dpkg --purge --force-remove-reinstreq \
intel-driver-compiler-npu intel-fw-npu intel-level-zero-npu \
intel-level-zero-npu-dbgsym

$ wget https://github.com/intel/linux-npu-
driver/releases/download/v1.28.0/linux-npu-driver-v1.28.0.20251218-
20347000698-ubuntu2404.tar.gz
$ tar -xf linux-npu-driver-v1.28.0.20251218-20347000698-
ubuntu2404.tar.gz

$ sudo apt update
$ sudo apt install libtbb12
$ sudo dpkg -i *.deb

$ sudo gpasswd -a ${USER} render
$ newgrp render
$ groups user
user : ... render ...

$ sudo reboot now
```

Cmd 99 shows the steps to install and run the OpenVINO™ toolkit notebooks. Since the VQA application is running on bare metal, it can access all CPU threads, the GPU,

and the NPU, as well as the USB cameras, microphone, and speaker through the USB and sound devices. Please use a web browser, the system IP, and port 8888 to access the notebooks.

Cmd 99. Commands to Install and Run OpenVINO™ Toolkit Notebooks

```
$ sudo apt install python3.12-venv
$ python3 -m venv openvino_env
$ source openvino_env/bin/activate
$ python -m pip install --upgrade pip
$ pip install openvino==2025.3.0

$ benchmark_app --help
...
Available target devices:  CPU  GPU  NPU

$ git clone --depth=1 -b 2025.3 \
  https://github.com/openvinotoolkit/openvino_notebooks.git
$ cd openvino_notebooks
$ python -m pip install --upgrade pip wheel setuptools
$ pip install -r requirements.txt

$ jupyter lab notebooks
```

Please find the following sample notebooks for the VQA application.

- Object Detection
 - [Convert and Optimize YOLOv12 with OpenVINO™ Toolkit](#)
- ASR
 - [Automatic speech recognition using Distil-Whisper and OpenVINO™ Toolkit](#)
 - [Video Subtitle Generation with OpenAI* Whisper and OpenVINO™ Toolkit Generate API](#)
- VLM
 - [Omnimodal assistant with Qwen2.5-Omni and OpenVINO™](#)
The Panther Lake-H iGPU can run both the 3 billion-parameter (Qwen2.5-Omni-3B) and the 7 billion-parameter (Qwen2.5-Omni-7B) models. Figure 26 shows the sample prompt, the video input, and the generated output text.
- TTS
 - [Voice tone cloning with OpenVoice and OpenVINO™ Toolkit](#)

Figure 26. Video QnA Output Text

Question:
Describe the video



Answer:
[===start thinker===]
Well, in the video, there's a black dog on a leash. It's walking on a sidewalk. The dog is wearing a collar and it seems to be moving at a steady pace. There's also a person walking next to the dog, but we can only see part of them, like their legs and feet. The sidewalk is made of concrete and there are some leaves scattered around. The background has a white wall. It looks like a normal day, just a dog being walked. So, what do you think about it? Do you have any other questions?

§