Intel® AI Edge Application Ready Verification

Guide

January 2026

Revision 2.1

Contents

Revision History

Revision	Description	Date
1.0	Initial release.	Sep 2025
1.1	Added Appendix A.	Nov 2025
2.1	Updated the “Access a System” section. Updated the “Submit Your Application Ready Verification Form” section.	Jan 2026

Introduction

This document will guide you through the process of onboarding your AI Edge Application and verifying its performance on an Intel qualified AI edge system. You will be required to complete the outlined steps, answer several questions, submit your responses, and attach an automated report to an email for evaluation.

Overview

Accelerate your impact—verify your application with an Intel AI Edge Systems to ensure peak performance on trusted, pre-qualified hardware. Deliver scalable, real-world AI solutions and stand out in a competitive market. Begin your integration today.

By joining Intel's AI Edge ecosystem, your application gains access to premium technical support, and high-impact co-marketing opportunities—empowering you to expand your reach, enhance visibility, and drive adoption with confidence.

How Does it Work?

This five-step verification process enables ISVs to verify their applications on qualified Intel AI Edge Systems. It ensures your solution is optimized for deployment on Intel hardware, backed by proven reliability that earns trust across the Intel ecosystem, enhancing credibility and accelerating market readiness.

Step 1: Acquire an Intel® AI Edge System & Software

Step 2: Download & Install Edge Sizing Tool from GitHub

Step 3: Configure Your Workload

Step 4: Submit Your Application Onboarding Request

Step 5: Application Review, Promotion and Benefits Activation

Step 1: Acquire an Intel® AI Edge System & Software

1a. Access a System

You can begin your application verification with one of the following options, either by procuring one of the Edge AI systems listed in our catalog, or by requesting access to our testbed where we host a number of verified Edge AI systems.

- Procure a system from our Recommended Systems Catalog.
- Request access to the Edge AI Testbed.

1b. System Software Requirements

For a smooth onboarding experience, you are required to install the following list of software ingredients. Refer to the steps shown here.

Category	Component	Version / Detail	Description
Operating System	Ubuntu*	22.04 or 24.04 LTS Desktop Linux	Validated Target OS
	Windows*	Windows 11	
AI Toolkit	OpenVINO Toolkit	Latest	AI inference toolkit optimized for Intel CPUs, GPUs, and NPUs.
Essential Runtime Component	Python	3.10+	A core runtime used for running OpenVINO pipelines.
Essential Runtime Component (Web)	Node.js	22+	Required for Edge AI Sizing Tool
Intel GPU Support	Intel GPU Driver	v25.09.32961.5	Required for OpenVINO GPU acceleration[1].
Intel NPU Support	Intel NPU Driver	v1.13.0	Required for OpenVINO NPU acceleration[1].
Essential Runtime Component	intel-gpu-tools	1.28-1ubuntu2	Required for Intel GPU performance & fine-tuning AI pipeline execution.

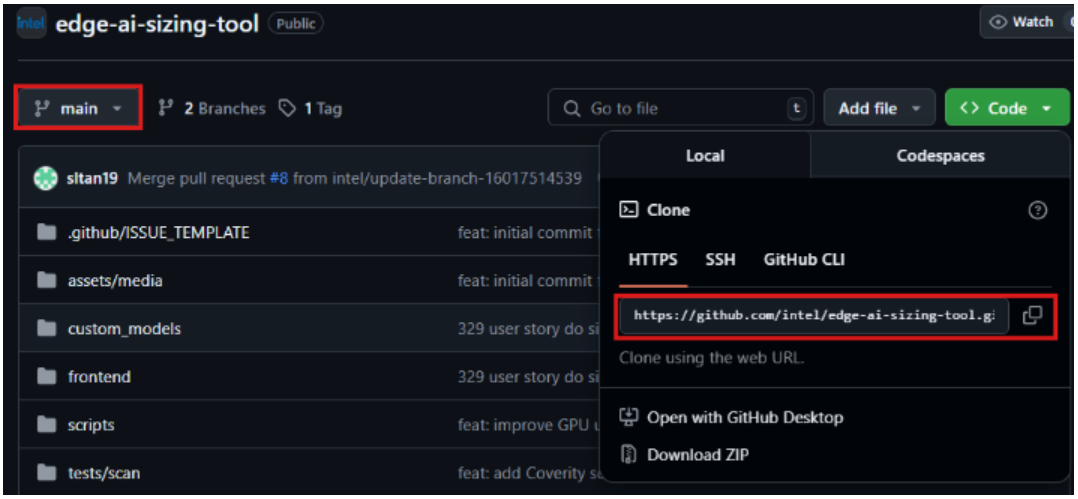
[1] *Systems that do not possess a discrete GPU or NPU will not require these drivers.*

Step 2: Download & Install Edge Sizing Tool

NOTE: Refer to Appendix A to see an alternate approach to completing Steps 2 and 3.

2a. Download the Edge AI Sizing Tool *(two options are available)*

Option 1: Cloning with Git tool (recommended)

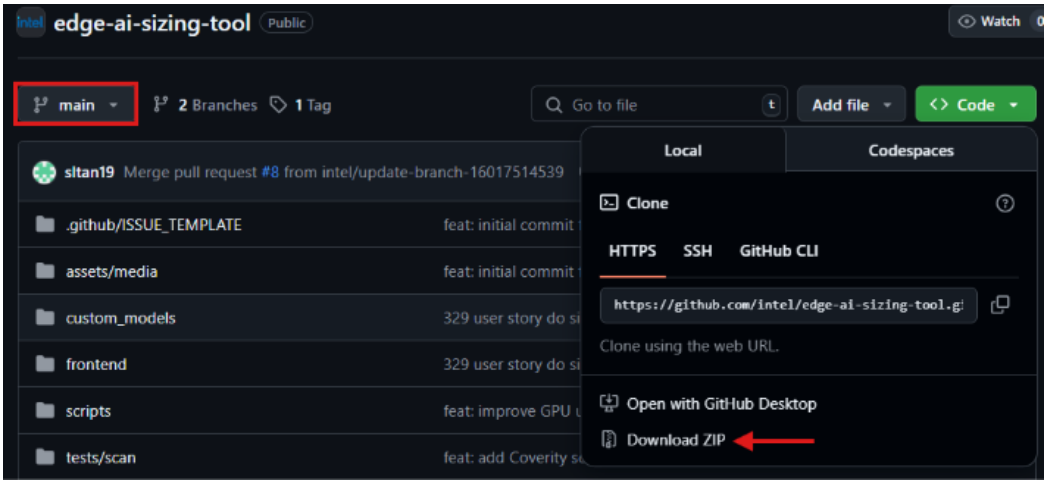


- Set the branch to "main" and click the <> Code button to see the download options.
- Start a command line console and copy the command below to begin download.

  $git clone https://github.com/intel/edge-ai-sizing-tool.git

  Note: Some network environments may require a network proxy setup.

Option 2: Direct Download



- Download the sizing tool at: *https://github.com/intel/edge-ai-sizing-tool*
- Set the branch to “main” and click the <> Code button to see the download options.
- Click “Download ZIP” to begin package download.

2b. Install Software and Test Run the Intel AI Edge Sizing Tool

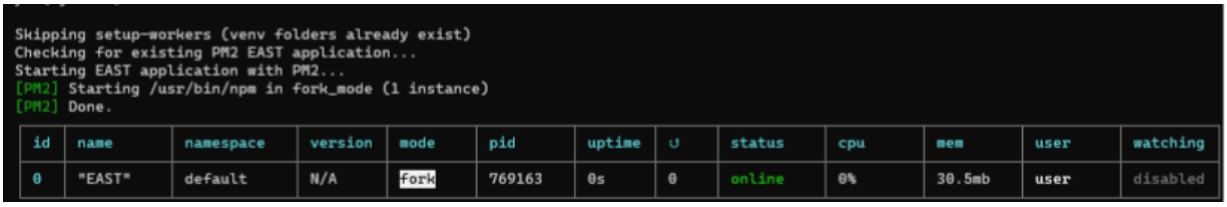
- ***(This step is only required if you performed a direct download)***
  Install unzip package and decompress edge-ai-sizing-tool-main.zip with the commands below.

  $sudo apt-get install unzip -y
  $sudo unzip edge-ai-sizing-tool-main.zip
  $mv edge-ai-sizing-tool-main edge-ai-sizing-tool

- Enter edge-ai-sizing-tool directory and install required dependencies and start application with commands below.

  $cd edge-ai-sizing-tool
  $ ./install.sh
  $./start.sh

Note: When you execute start.sh, the setup process will automatically generate a random secret key for the PAYLOAD_SECRET variable in the .env file. By running this script, you acknowledge and accept the terms of use of this automatically generated secret.



- Once the start script has been initiated, use a web browser application and go to: http://localhost:8080.

  Note: For correct operation, this application requires port 8080 and ports between 5000 to 6000 to be available for frontend and worker services respectively.

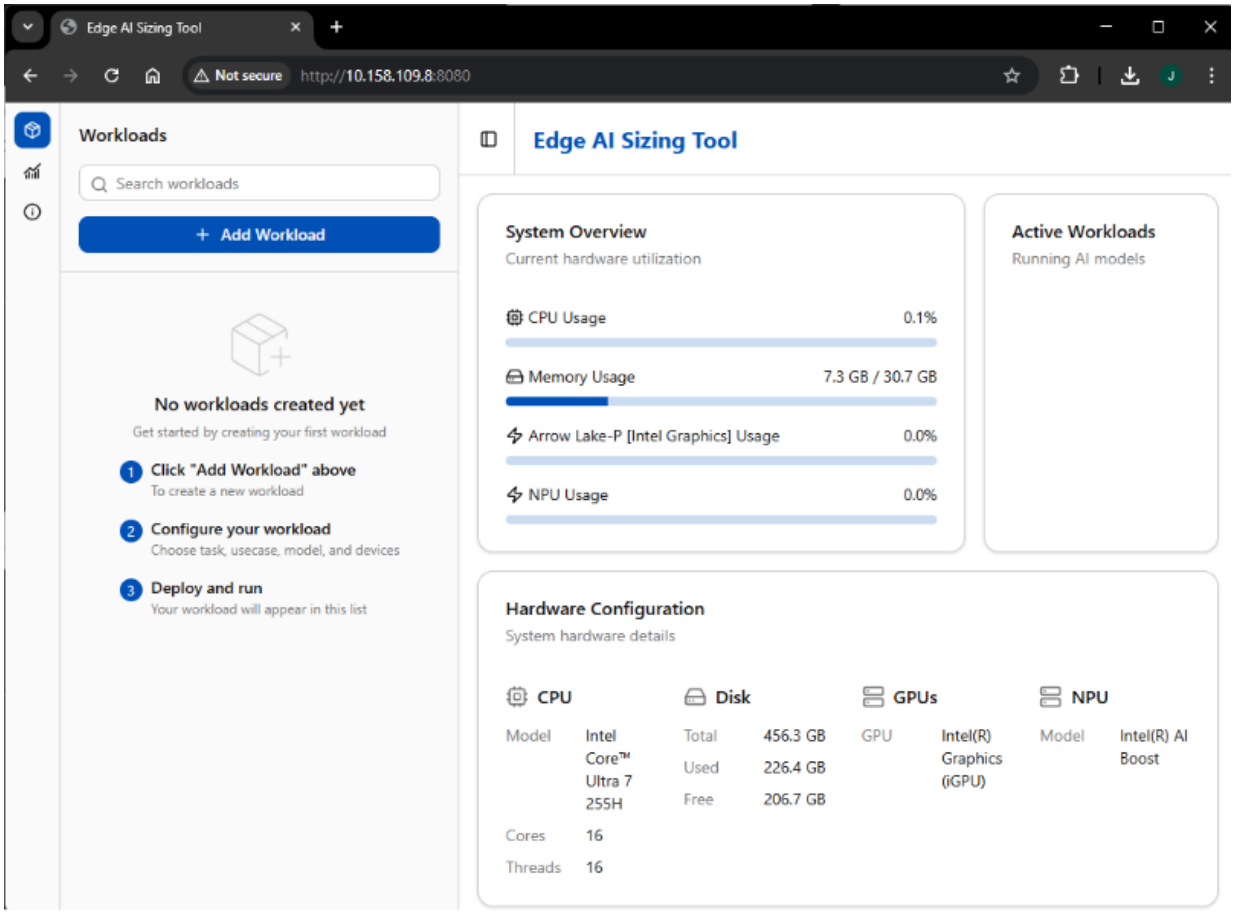


Figure 1: User interface of Edge AI Sizing Tool landing page.

Step 3: Configure Your Workload

NOTE: Refer to Appendix A to see an alternate approach to completing Steps 2 and 3.

Next, let's "build" a workload that closely mimics your application use-case with the options below.

3a. Video Processing

Perform the steps in this section if your application utilizes video processing and analytics in a single pipeline as used for object detection, face recognition, or motion tracking for industries like retail, safety & security, education, or healthcare.

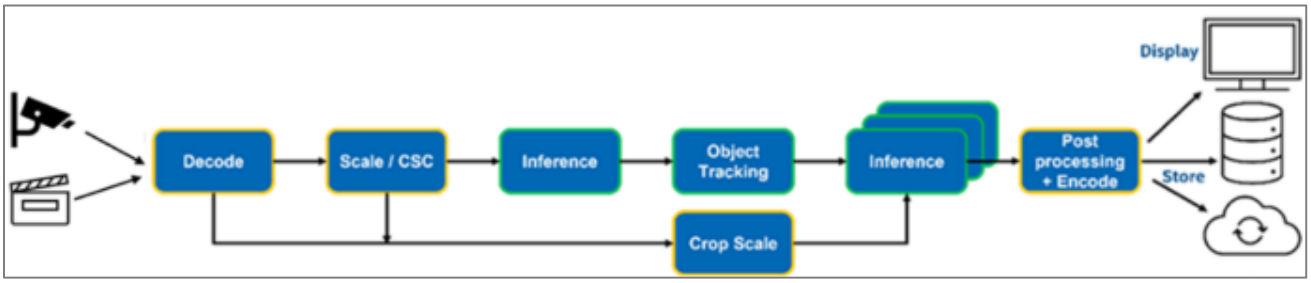
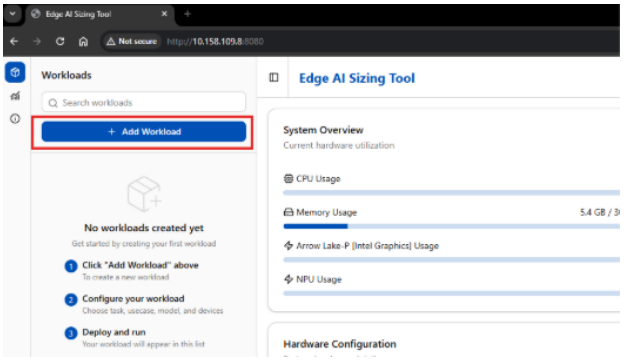
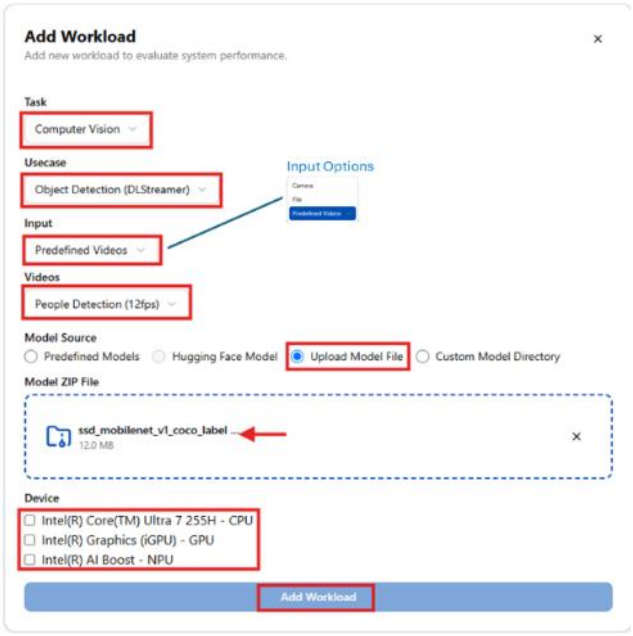
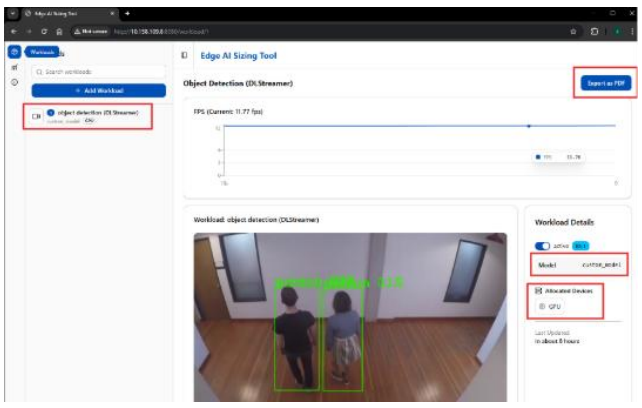


Figure 2: Example of a Video Processing and Analytics Pipeline.

Follow the steps below to orchestrate and run the workload that mimics your use-case

• **Add Workload** 	a. Click + Add Workload to Start the Workload Configuration.
• **Configure Workload** 	a. Set Task to "**Computer Vision**" . b. Set Use-case to "**Object Detection (DLStreamer)**". c. For Input – Consider the option that best suits your application *(Camera, File or Predefined Videos that comes with this application)*. d. Select the test video that works well with your AI Model. e. Upload your optimized custom model (.zip) file that should contain .bin and .xml files. Please refer to the "Custom Model Guide" section below for packaging details. f. Select target device that will load the model. Only one device per workload may be selected for model execution. g. Click "**Add Workload**" to start workload pipeline.
• **Verify Workload and Generate Report** 	a. Upon workload activation, the added workload will appear in the left pane of the workload interface and you will then be able to observe its status and workload behavior on the right pane as illustrated in the image. b. In order to generate a report, select "**Export as PDF**". By default, the report will be downloaded and saved in the default browser's Downloads folder.

Custom Model Guide

- Package your custom model into a ZIP file with the file structure below.

```
my-model.zip
├── model.xml
├── model.bin
└── labels.txt   # (optional)
```

- Only OpenVINO model formats are supported. ZIP must contain all necessary model files in OpenVINO IR format (e.g., .xml, .bin, and optionally labels.txt for object detection).

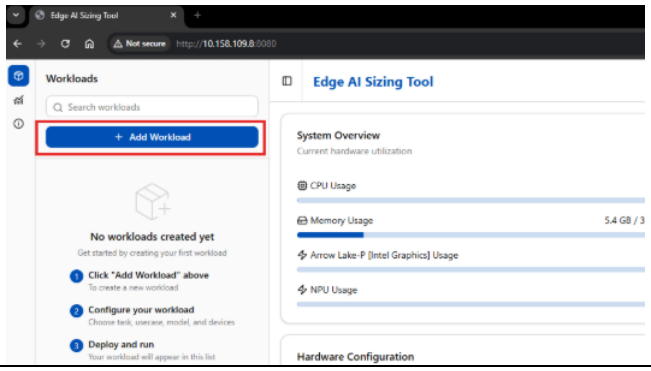
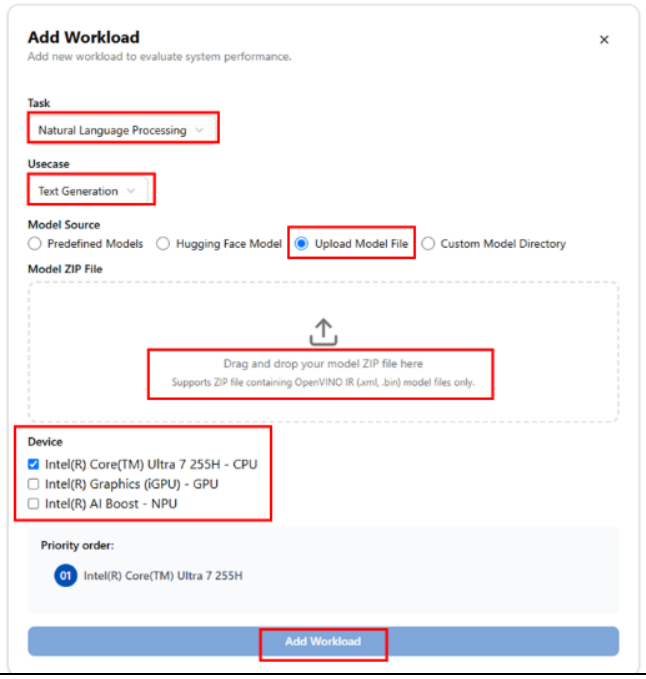
- To learn more, go to *https://github.com/intel/edge-ai-sizing-tool/blob/main/custom_models/README.md#custom-model-directory*

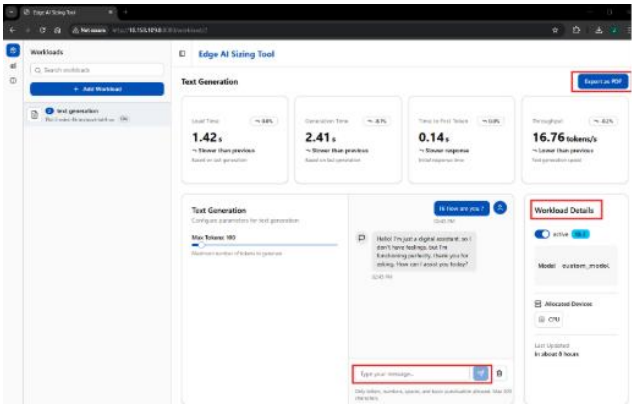
Note: Custom models loaded into testbed systems will be deleted after each user session.

3b. NLP or Text Generation

Perform the steps in this section if your application utilizes Natural Language Processing (NLP) or Text Generation efficiently in edge environments, ensuring fast responses without relying on cloud connectivity

Follow the steps below to orchestrate and run the workload that mimics your use-case.

• **Add Workload** 	a. Click "+ Add Workload" To Start Workload Configuration.
• **Configure Workload** 	a. Set Task to "**Natural Language Processing**". b. Set use-case to "**Text Generation**" c. Upload your optimized custom model (.zip) file that should contain .bin and .xml files). d. Select target device that will load the model. e. Click "**Add Workload**" to start workload pipeline.

• **Verify Workload and Generate Report** 	a. Verify workload listed on workload pane on the left and workload details as shown in this image. b. Enter a prompt in the “Type your message” field and hit the button to generate text. c. In order to generate a report, select “Export as PDF”. By default, the report will be downloaded and saved in the default browser’s Downloads folder.

Step 4: Submit Your Application Ready Verification Form

Please fill out the online Application Ready Verification Form where you will provide the following:

General Information

- Name
- Company Name
- Email
- Project Name

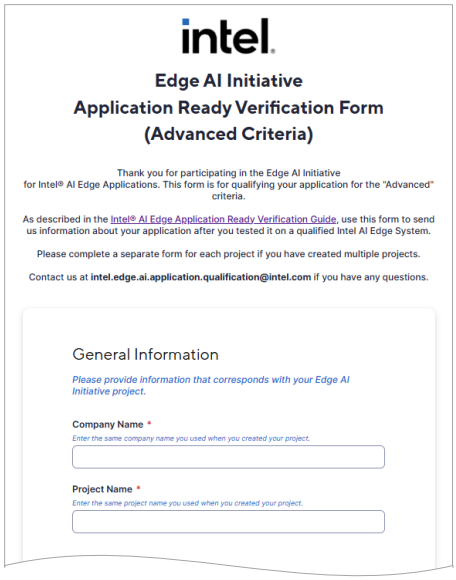
Details

- System Manufacturer
- System Model
- System CPU
- System GPU
- System Source
- Pipeline Description

Uploads

- A picture of your system.
- A block diagram of your application pipeline.
- The report you generated after running the Edge AI Sizing Tool.

Below is the URL location where you can find the form.
https://app.smartsheet.com/b/form/019a3bfa6a0b7623876994c2950fc14f



Examples: Pipeline Description and Block Diagram

Below is an example to the question where we ask you to provide your application pipeline description and a block diagram.

EXAMPLES

Pipeline Description:

Our software application uses real-time video processing and analytics software designed to detect, track, and analyze objects or behaviors from live or recorded video feeds. Beyond video, we also run Generative AI and Large Language Model (LLM) features — like answering questions, summarizing insights, or generating reports based on what's seen in the video. These powerful features require components like DL Streamer, Hugging Face and other needful extensions to make our workload efficient on Intel Architecture.

Block Diagram:

Step 5: Application Review, Promotion and Benefits Activation

Intel will review your submitted information and may contact you with any follow-up questions. Additionally, Intel may request to schedule an application review meeting where our representatives can meet with your team for an informal discussion about your application details. This session will help us better understand your solution, explore potential benefits and investigate future engagement opportunities.

Support

Our team is ready to help ensure a smooth and successful integration into the Edge AI Applications Program. For any support or assistance required during the onboarding process, please reach out to us at:

intel.edge.ai.application.qualification@intel.com

Appendix

Appendix A: An Alternative Verification Method *(replaces Steps 2 and 3)*

Note: This alternative method replaces Steps 2 and 3 in the "How Does it Work" section.

The method outlined below enables software vendors to test the entire workload while providing the appropriate output to achieve the desired verification of the software on Intel hardware.

All steps include commands for both Linux and Windows. Be sure to capture all the outputs that can be sent to Intel for verification.

A1: Aquire the Hardware

Gain access to hardware as outlined in Step 1 "Acquire an Intel AI Edge System & Software" on page 7.

For steps 2 and 3, where applicable, execute the commands in the Userspace, VM, or Container within which the application runs.

A2: Capture Hardware Info

a. Use the following commands to capture information about the hardware:

Linux	Windows
dmidecode	msinfo32

b. Install the full application on to the system

c. Execute the application and obtain the list of runtime libraries in use

Linux	Windows
lsof or /proc/<pid>/maps	tasklist

A3: Configure the load

a. Configure a light input load (e.g. # streams or # requests)

b. Run the application and collect measurements plus screenshots and record the utilization of the CPU, GPU, and/or NPU.

	Linux	Windows
CPU	top or mpstat	Task Manager (Performance view)
GPU	intel_gpu_top (from intel-gpu-tools) or qmassa (see *https://github.com/ulissesf/qmassa*)	Task Manager (Performance view)
NPU	Intel-npu-top (see *https://pypi.org/project/intel-npu-top/*) or nputop (see *https://github.com/ZoLArk173/nputop*)	Task Manager (Performance view)

c. Repeat 1 & 2 with medium input load.

d. Repeat 1 & 2 with heavy input load.

A4: Submit results and additional details

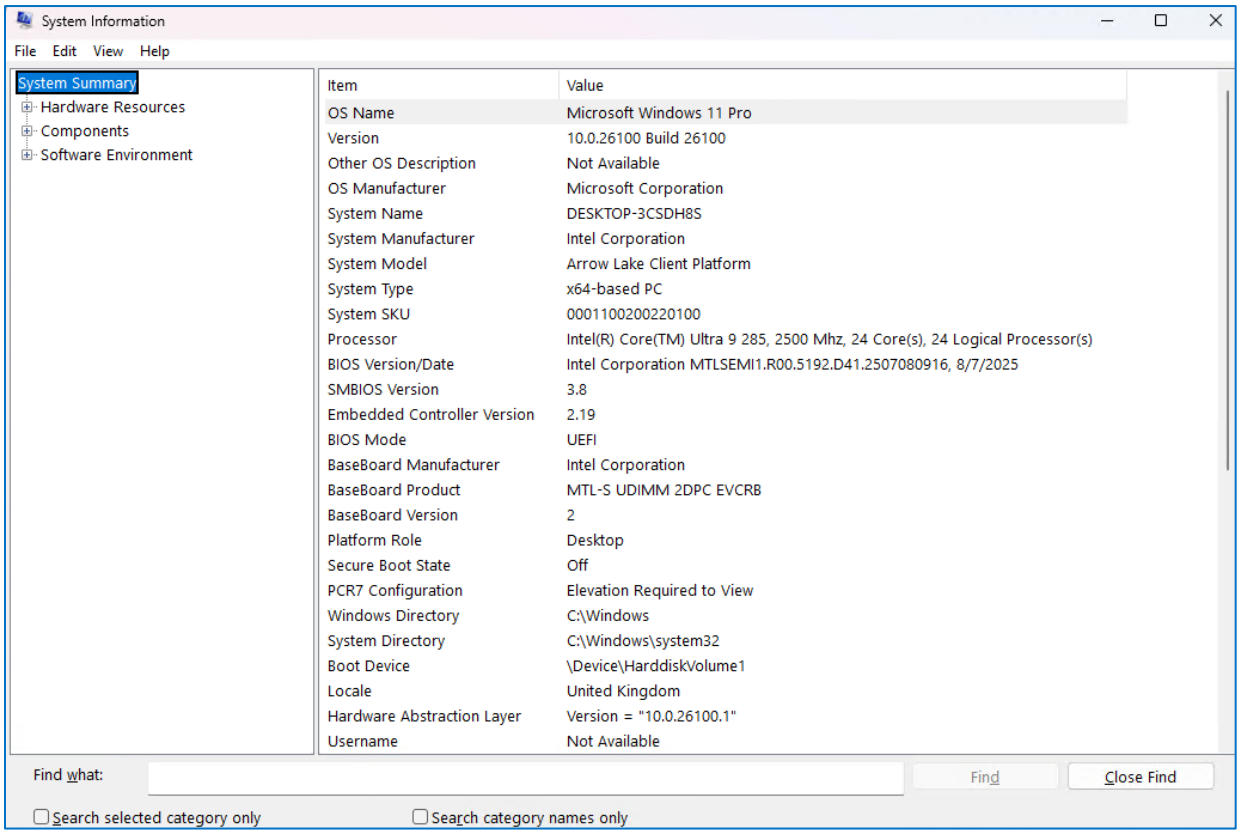
Gather the outputs and submit them as outlined in step 4 “Submit Your Application Onboarding Request” on page 17. Be sure to include a high-level diagram of the application pipeline and a picture of the system if you are not using the Intel Test Bed.

Example #1: Full Application Verification on Windows

In this example, we walk through the steps to verify an application called “classify loop” on an Intel AI Edge System running **Windows**.

Capture System Configuration

1. Open a command window
2. Enter: “msinfo32”
3. Save the results to a file, e.g. system.nfo



List Runtime Libraries Used by the Application

1. Install and run the application on the system
2. Open a cmd window with admin privilege
    - tasklist /fi "imagename eq [app name]" /m
    - e.g. tasklist /fi “imagename eq classify_loop.exe” /m
    - alternatively, may use “listdlls” (see https://learn.microsoft.com/en-us/sysinternals/downloads/listdlls) may be used instead
3. Save the output to a text file, say “app_runtimes.txt”

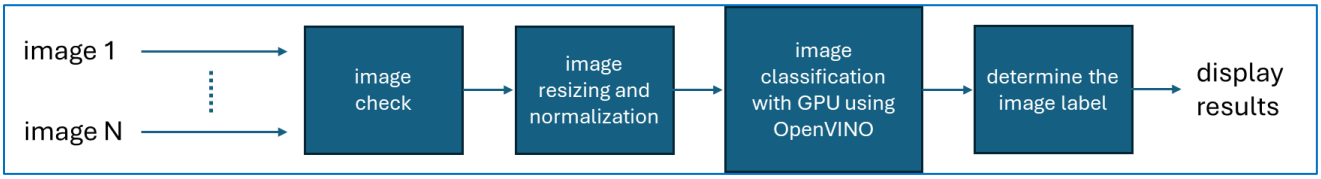
```
c:\>tasklist -fi "imagename eq classify_loop.exe" /m

Image Name                     PID Modules
========================= ======== ============================================
classify_loop.exe             7520 ntdll.dll, KERNEL32.DLL, KERNELBASE.dll,
                                   USER32.dll, win32u.dll, GDI32.dll,
                                   gdi32full.dll, msvcp_win.dll, ucrtbase.dll,
                                   ADVAPI32.dll, msvcrt.dll, sechost.dll,
                                   RPCRT4.dll, IMM32.DLL, python313.dll,
                                   WS2_32.dll, VERSION.dll, bcrypt.dll,
                                   VCRUNTIME140.dll, bcryptprimitives.dll,
                                   python3.DLL, _ctypes.pyd, ole32.dll,
                                   combase.dll, libffi-8.dll, OLEAUT32.dll,
                                   _bz2.pyd, _lzma.pyd, _queue.pyd, _wmi.pyd,
                                   PROPSYS.dll, VCRUNTIME140_1.dll,
                                   _hashlib.pyd, libcrypto-3.dll, _socket.pyd,
                                   IPHLPAPI.DLL, select.pyd, _ssl.pyd,
                                   CRYPT32.dll, libssl-3.dll,
                                   _multiarray_umath.cp313-win_amd64.pyd,
                                   libscipy_openblas64_-13e2df515630b4a41f92893
                                   938845698.dll,
                                   msvcp140-263139962577ecda4cd9469ca360a746.dl
                                   l, _umath_linalg.cp313-win_amd64.pyd,
                                   cv2.pyd, COMDLG32.dll, shcore.dll,
                                   SHLWAPI.dll, WSOCK32.dll, SHELL32.dll,
                                   wintypes.dll, COMCTL32.dll, MFPlat.DLL,
                                   MF.dll, MFReadWrite.dll, dxgi.dll,
                                   d3d11.dll, cfgmgr32.dll, powrprof.dll,
                                   MFCORE.DLL, RTWorkQ.DLL, UMPDC.dll,
                                   _pyopenvino.cp313-win_amd64.pyd,
                                   MSVCP140.dll, openvino.dll, tbb12.dll,
                                   tbbmalloc.dll, _uuid.pyd,
                                   py_tensorflow_frontend.cp313-win_amd64.pyd,
                                   kernel.appcore.dll, uxtheme.dll,
                                   clbcatq.dll, amsi.dll, USERENV.dll,
                                   profapi.dll, MpOav.dll, MSASN1.dll,
                                   openvino_ir_frontend.dll, unicodedata.pyd,
                                   mswsock.dll, DNSAPI.dll, NSI.dll,
                                   rasadhlp.dll, fwpuclnt.dll,
                                   openvino_auto_batch_plugin.dll,
                                   openvino_intel_gpu_plugin.dll,
                                   SETUPAPI.dll, OpenCL.dll, dxcore.dll,
                                   windows.staterepositorycore.dll,
                                   directxdatabasehelper.dll, igdrcl64.dll,
                                   WINMM.dll, igdgmm64.dll,
                                   windows.storage.dll, igdfcl64.dll,
                                   igc64.dll, DEVOBJ.dll, WINTRUST.dll,
                                   tbbbind_2_5.dll
```

Determine Resource Utilizations by the Application at Different Loads

1. Open the Task Manager
2. Show the “Performance” view
3. Run the application with light load
4. Record the average utilizations of CPU, GPU (Compute), and NPU (if applicable)
5. Repeat 1-4 with medium load
6. Repeat 1-4 with heavy load
7. Save the data into a txt file, say utilizations.txt

Provide a Block Diagram of the Application

Application name: Classify Loop

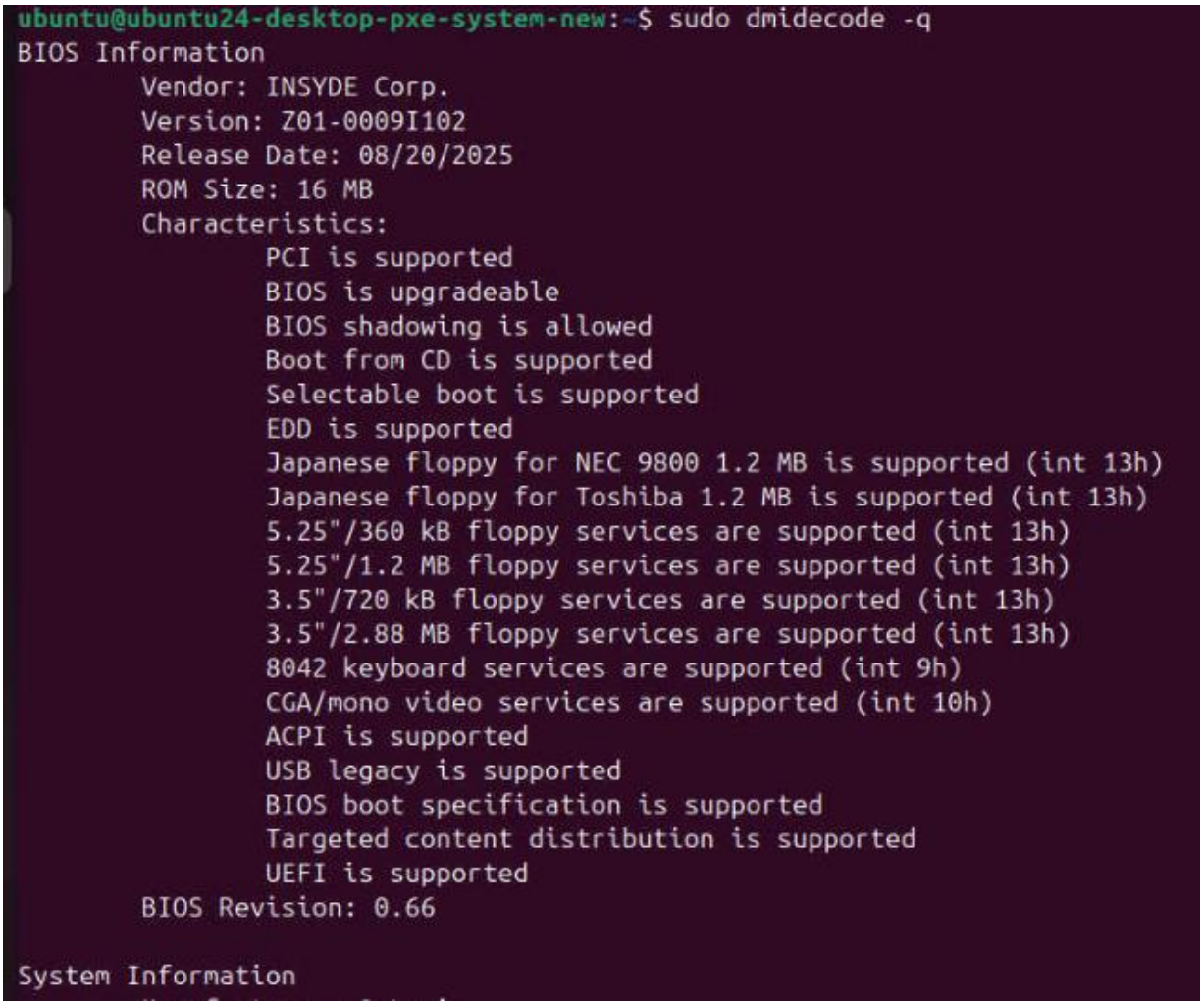


Example #2: Full Application Verification on Linux (Ubuntu)

In this example, we walk through the steps to verify an application called “classify loop” on an Intel AI Edge System running **Ubuntu**.

Capture System Configuration

1. Open a command window
2. Run: “sudo dmidecode -q”
3. Save the results to a file, e.g. system.txt



List Runtime Libraries Used by the Application

1. Install and run the application on the system
2. Open a cmd window
    - Obtain the process id of the application process
        - pgrep classify_loop.exe
    - Obtain the list of runtime libraries in use by the application process
        - lsof -p <process id>
3. Save the output to a text file, say “app_runtimes.txt”

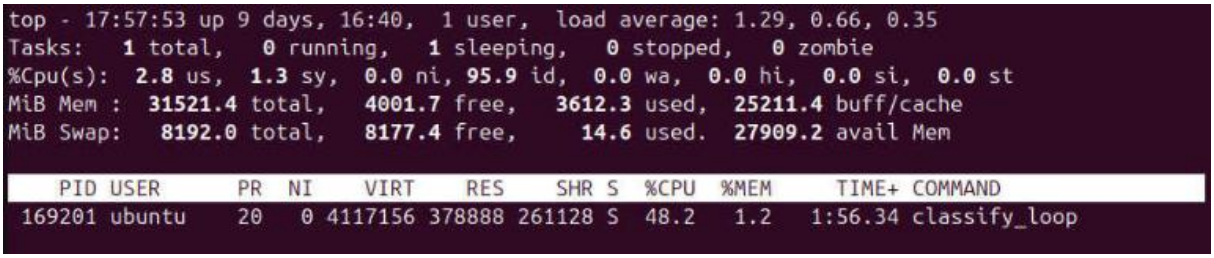
```
ubuntu@ubuntu24-desktop-pxe-system-new:~$ lsof -p 124576
COMMAND      PID   USER   FD   TYPE  DEVICE  SIZE/OFF     NODE NAME
classify_ 124576 ubuntu  cwd    DIR   252,0      4096 10505643 /home/ubuntu/EdgeAI_App_Qualifi
cation/hello_classification/dist/classify_loop
classify_ 124576 ubuntu  rtd    DIR   252,0      4096        2 /
classify_ 124576 ubuntu  txt    REG   252,0   5949088 10505644 /home/ubuntu/EdgeAI_App_Qualifi
cation/hello_classification/dist/classify_loop/classify_loop
classify_ 124576 ubuntu  mem    REG   252,0 121930472 10506779 /home/ubuntu/EdgeAI_App_Qualifi
cation/hello_classification/dist/classify_loop/alexnet.bin
classify_ 124576 ubuntu  mem    REG    0,16                69 anon_inode:i915.gem (stat: No s
uch file or directory)
classify_ 124576 ubuntu  DEL    REG     0,1             179612 /dev/zero
classify_ 124576 ubuntu  DEL    REG     0,1             179600 /dev/zero
classify_ 124576 ubuntu  mem    REG   252,0  91426808  8132970 /usr/lib/x86_64-linux-gnu/libig
c.so.2.18.5+0
classify_ 124576 ubuntu  mem    REG   252,0 116298256  8132975 /usr/lib/x86_64-linux-gnu/libop
encl-clang.so.15
classify_ 124576 ubuntu  DEL    REG     0,1             189510 /dev/zero
classify_ 124576 ubuntu  mem    REG   252,0  28012608  8407838 /usr/lib/x86_64-linux-gnu/intel
-opencl/libigdrcl.so
classify_ 124576 ubuntu  mem    REG   252,0   5155617 10505656 /home/ubuntu/EdgeAI_App_Qualifi
cation/hello_classification/dist/classify_loop/_internal/openvino/libs/libopenvino_tensorflow_
frontend.so.2530
classify_ 124576 ubuntu  mem    REG   252,0  17648665 10505649 /home/ubuntu/EdgeAI_App_Qualifi
cation/hello_classification/dist/classify_loop/_internal/openvino/libs/libopenvino.so.2530
classify_ 124576 ubuntu  mem    REG   252,0   5358433 10505703 /home/ubuntu/EdgeAI_App_Qualifi
cation/hello_classification/dist/classify_loop/_internal/openvino/_pyopenvino.cpython-312-x86_
64-linux-gnu.so
```

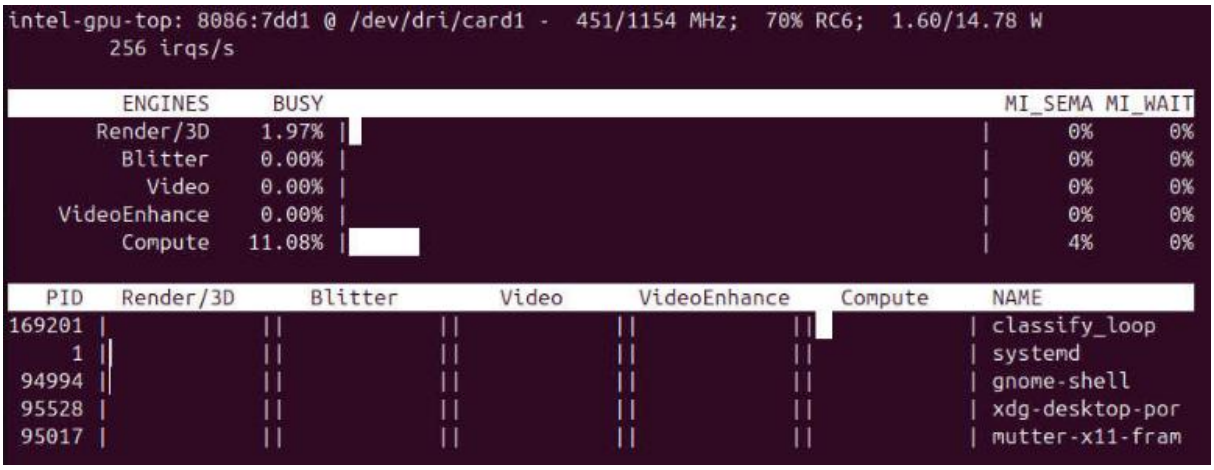
Determine Resource Utilizations by the Application at Different Loads

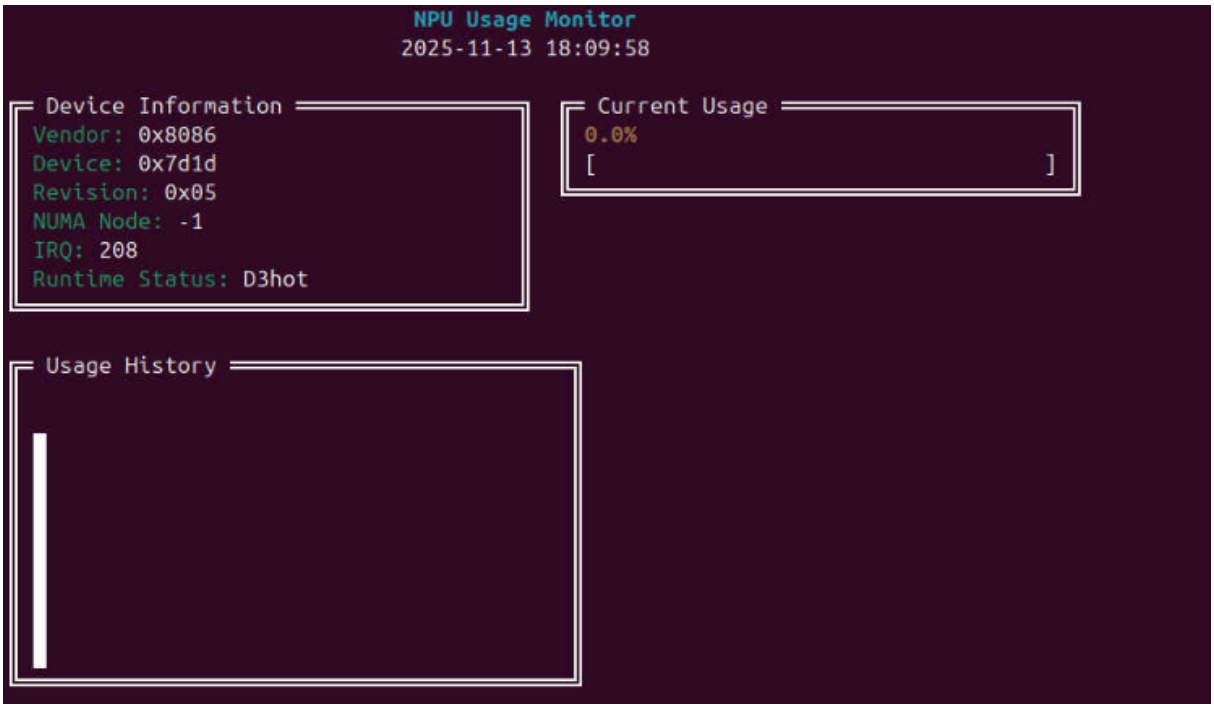
1. Perform one or all three of these commands:
    - Open a command window and start "**top -p <process id>"**
      <and/or>
    - Open a command window and start "**sudo intel_gpu_top**"
      <and/or>
    - Open a command window and start "**intel-npu-top**"

2. Run the application with light load
3. Record the average utilizations of CPU, GPU (Compute), and NPU (if applicable)
4. Repeat 1-4 with medium load
5. Repeat 1-4 with heavy load

6. Save the data into a txt file, say utilizations.txt







Provide a Block Diagram of the Application

Application name: Classify Loop