Unlocking the AI Power of Intel® Arc™ GPU for the Edge: A Deep Dive into Hardware and Software Enablement

White Paper

Authors:

Ramesh Perumal, Software Enabling and Optimization Engineer
Oluwaseyi Ogebule, GPU Product Manager
Rohit D'Souza, AI Product Manager

Contributor:

Ryan Loney, Senior Product Manager for OpenVINO™

April 2024

Document Number: 817734-1.0

Contents

Figures

Tables

Revision History

Date	Revision	Description
April 2024	1.0	Initial release.

1.0 Abstract

This white paper explores the artificial intelligence (AI) capabilities of Intel® Arc™ GPUs for the edge, delving into hardware architecture, technical specifications, and software enablement.

The paper begins by outlining hardware capabilities and introducing the innovations in X^{e}-core architecture. It then explores the software ecosystem and illustrates the capability of OpenVINO™ toolkit in optimizing and deploying PyTorch* models.

A section on benchmarks highlights the inference throughput of an Arc GPU across a few AI models, targeting classification, detection, segmentation, pose estimation, and natural language processing.

Lastly, the paper concludes with Intel Arc GPU AI use cases at the edge referencing image classification and object detection.

The case study delves into the implementation details including software requirements, showcasing performance gains achieved through Arc GPUs. By combining powerful hardware with a robust and mature software environment, Intel Arc GPUs offer a competitive solution for various demanding AI workloads.

2.0 Introduction

Artificial Intelligence (AI) at the Edge is the deployment of AI applications on or close to the device where data is generated or located, as opposed to centralized cloud computing facilities. The edge can be any device outside a central data center, such as network video recorders, security cameras, industrial robots, ultrasound machines, on-prem servers, etc. Instead of sending data to the cloud for processing, AI algorithms are run on the edge device. The key benefits of processing and analyzing data at the edge include (but are not limited to):

- **Reduced Latency**: Low latency is critical for applications requiring real-time responses. Eliminating the need for data to travel back and forth to the cloud significantly reduces latency.
- **Improved Bandwidth Efficiency**: By processing data locally, edge AI eliminates the need to send large amounts of data to the cloud, which can consume significant bandwidth.
- **Better Privacy and Security**: With edge AI, sensitive data can stay on device or on premises.

AI at the edge is exploding with new use cases and workloads being developed daily. These AI workloads often require a high degree of parallel processing and memory bandwidth for peak performance, dedicated hardware, optimized architecture for compute efficiency, and reduced latency with faster results for real-time processing. A discrete Graphics Processing Unit (GPU) may be the ideal solution for edge AI use cases requiring high performance and complex model support.

3.0 Introducing Intel® Arc™ GPU for the Edge

The Intel® Arc™ GPUs for the Edge are based on the highly scalable Intel X^e-core architecture and designed to enable innovation for AI, visual computing, and media processing. The Intel X^e-core architecture scales from integrated GPUs to discrete GPUs. With support for a more open, standards-based software stack, customers can run high-performance AI applications and solutions using the OpenVINO™ toolkit. The OpenVINO tool suite provides a streamlined development workflow to deploy inference workloads. It enables developers to create AI models once and deploy on any Intel hardware platforms. See OpenVINO Docs to learn more.

The Intel Arc GPUs represent a leap forward for graphics technology at the device edge combining advanced AI, superior graphics, and efficient media processing in a single GPU. Intel Arc GPUs pair seamlessly with select Intel® Core™ CPU Processors for a complete solution. Built on Intel's advanced X^e graphics architecture, the Intel Arc GPU delivers scalable performance across various computing environments, from integrated graphics to high-performance discrete graphics. Intel X^e HPG architecture provides purpose-built acceleration for key edge usages and workloads, including the Intel® X^e Matrix Extensions (Intel® XMX) AI Engine to speed up inference and the X^e Media Engine for faster transcode and other media-processing tasks. Intel Arc GPUs target the edge specifically with five-year long-life availability and support, diverse edge-focused form factors and support for edge-constrained usage conditions.

3.1 Hardware Features of Intel® Arc™ GPU for AI

Intel X^e-core architecture represents a significant evolution in GPU design, emphasizing versatility and performance across a spectrum of computing needs from Cloud to Edge. It incorporates advanced features such as scalable data parallelism, efficient AI acceleration and support for rich graphical rendering techniques. This architecture enables Intel® Arc™ GPUs to deliver high performance for AI applications, providing a flexible foundation for future innovations in graphics and computing technologies.

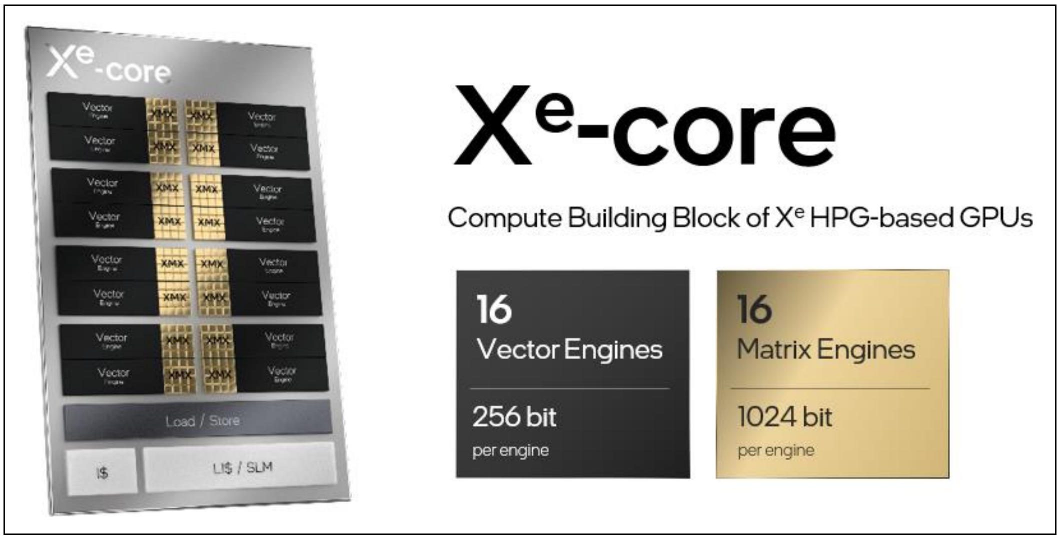

Figure 1: Compute building block of Intel Arc GPUs

Intel Arc GPUs are engineered to tackle the demands of Edge AI workloads. The Intel® XMX AI engine embedded within represents a sophisticated computational framework optimized for executing neural network operations. It leverages tensor processing capabilities to accelerate the throughput of AI inferencing tasks. This engine is specifically engineered to enhance the parallel processing of AI algorithms, thereby substantially increasing the efficiency and speed of AI-related computations on the Intel Arc GPU compared to the host CPU or integrated GPU.

By utilizing advanced processing architectures within its AI engines, Intel Arc GPUs can enhance AI and machine learning workloads. The built-in systolic array allows for high-throughput, energy-efficient computation, ideal for tasks requiring matrix multiplication in common machine learning algorithms. This architecture improves the efficiency of AI operations, enabling faster and more power-efficient processing of AI tasks directly on the GPU, thereby enabling overall performance for AI-driven applications and workloads.

Intel Arc GPU provides robust software support with seamless integration to the OpenVINO™ toolkit, enhancing Edge AI workloads. OpenVINO facilitates the deployment of AI models across a range of hardware solutions, ensuring that users can leverage the full potential of Intel Arc GPUs for a wide range of applications, from computer vision to deep learning inference. Eliminate vendor lock-in with a more open, standards-based software stack to build high-performance AI applications and solutions. Code once and run across GPUs, CPUs, and other hardware accelerators, with the open-sourced OpenVINO toolkit.

Developers are choosing OpenVINO to -

- Gain efficiencies by maximizing existing investments and reducing the need for specialized hardware-specific skill sets.
- Scale fluidly to extend applications to CPUs, GPUs, and other heterogeneous computing.
- Deploy Large Language Models (LLMs) at the edge with GPU-specific optimizations and reduced memory footprint.
- Handle diverse challenges at the edge with choices among the most popular AI models and frameworks, including TensorFlow* and PyTorch*.
- Enhance interoperability and compatibility across heterogeneous systems and reduce costs with low or no licensing fees.

3.2 Technical Specifications

As shown in Table 1, Intel® Arc™ GPUs offer a range of power-optimized and performance-optimized solutions with a common GPU architecture based on X^e-cores. All SKUs support long-life and product availability as well as extended software support.

	A310E	A350E	A370E	A380E	A580E	A750E
TDP	75W	25-35W	35-50W	75W	185W	225W
Graphics Clock (MHz)	2000	1150	1550	2000	1700	2050
X^e –cores	6	6	8	8	24	28
Execution Units	96	96	128	128	384	448
INT8 TOPS[1]	49	28	38	66	167	235
FP16 TFLOPS[1]	24.6	14.1	25.4	32.8	83.6	117.6
FP32 TFLOPS[1]	3.1	1.8	3.2	4.1	10.4	14.7
Memory (GDDR6)	4 GB	4 GB	4 GB	6 GB	In Planning	
Memory Bandwidth (up to)	112 GB/s	112 GB/s	112 GB/s	186 GB/s	In Planning	
Operating System	Windows 10/11 Client; Windows 10 LTSC Linux – Ubuntu[4]					
Use Conditions[2]	Embedded	Embedded	Embedded	PC Client	PC Client	PC Client
Launch Date	April 2024				2H 2024	
Product Availability[3]	5 years					

Table 1: Technical Specifications for Intel® Arc™ GPU[1,2,3,4]

[1] Precisions supported: INT8, INT16, INT32, FP16, FP32 | (with XMX): INT2/4, INT8, BF16, FP16, FP32

[2] Embedded use conditions of up to 5 years, up to 80 percent active, PC Client use conditions of up to 5 years, up to 20 percent active

[3] Intel does not commit or guarantee product Availability or Technical Support by way of roadmap guidance. Intel reserves the right to change roadmaps or discontinue products, software, and software support services through standard EOL/PDN processes. Please contact your Intel account rep for additional information.

[4] For latest supported OS: Linux-Ubuntu releases - Intel® Arc™ Graphics Driver - Ubuntu; Windows releases - Intel® Arc™ & Iris® Xe Graphics - Windows

The Intel Arc A7xxE GPU offers high performance for heavy AI workloads and expansive use cases such as facial recognition, gesture identification, generative conversational speech, and many more.

The Intel Arc 5xxE GPU offers an unparalleled blend of immersive visual experiences and enhanced AI inferencing capabilities.

The Intel Arc 3xxE GPU series offers up to 6 GB of GDDR6 memory and TDP ranging from 25W to 75W, perfect for low power and small form factor designs targeting AI at the Edge.

Intel Arc GPUs are built to be paired alongside and complement Intel Core processors. This maximizes the value of existing investments and the flexibility of implementations. For higher levels of performance and efficiency, pair Intel Arc GPUs with select Intel Core processors. Supported host processors include:

- 10th Gen Intel® Core™ Processors
- 11th Gen Intel® Core™ Processors
- 12th Gen Intel® Core™ Processors
- 13th Gen Intel® Core™ Processors
- Intel® Core™ Processors (14th Gen)
- Intel® Xeon® W-3400 and W-2400 Processors

4.0 AI Applications

The Intel® Arc™ GPUs enable various AI use cases in several segments, including machine vision and natural language processing. Figure 2 illustrates a few examples of use cases in retail, healthcare, smart cities, and industrial robotics, where Intel Arc GPUs play a key role.

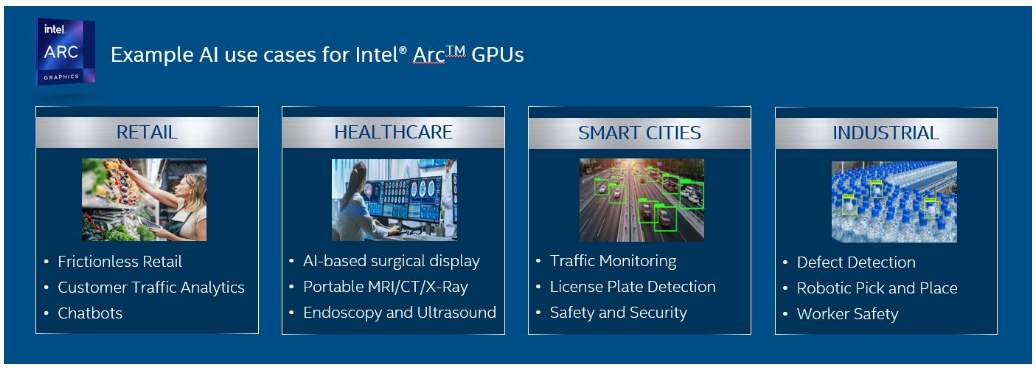

Figure 2: Examples of AI use cases for the Intel® Arc™ GPUs

These use cases demonstrate Intel Arc GPU's multifaceted role in AI deployment, showcasing its ability to address unique challenges and fulfill specific needs of diverse industries. In retail, Intel Arc GPUs process vast datasets for customer behavior analytics and conversational AI for personalized shopping experiences. The Healthcare segment leverages Intel Arc GPUs for high-resolution medical imaging and real-time diagnostic in multiple applications. In Smart Cities, the GPU enhances the management of continuous data flow from traffic and surveillance sensors. Furthermore, within industrial environments, visual inspection tasks and coordinating complex robotic movements are prime workloads for AI. Beyond the segments and use cases mentioned, countless other applications will leverage Intel Arc GPUs for AI to solve unique challenges and enhance operational efficiency.

5.0 Software Enablement

Figure 3 is a representative software stack to enable AI inference on the Intel® Arc™ GPU using the OpenVINO™ toolkit[5]. OpenVINO is a versatile solution to bridge the gap between model development and efficient deployment. By providing tools to optimize trained neural networks for various hardware architectures, OpenVINO enables developers to achieve improved inference performance and reduced latency without significantly compromising accuracy. The toolkit's model optimization techniques, including quantization, distillation, pruning, and layer fusion, allow for streamlined inference execution, making it an asset in applications leveraging object detection, image classification, segmentation, real-time generation of images, and sophisticated large language model processing.

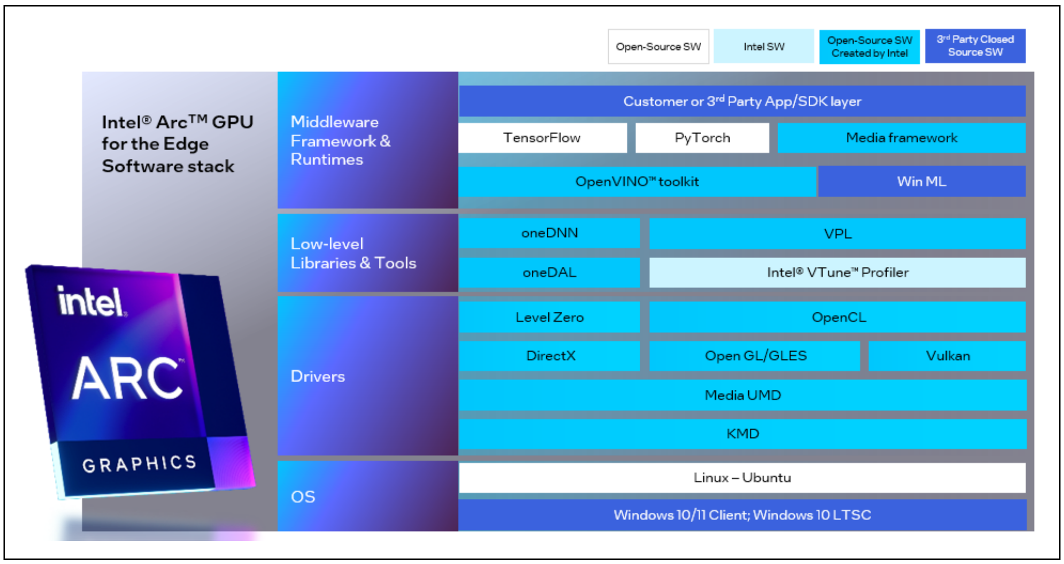

Figure 3: Representative Software Stack for Intel® Arc™ GPU

[5] For latest supported OS: Linux-Ubuntu releases - Intel® Arc™ Graphics Driver - Ubuntu; Windows releases - Intel® Arc™ & Iris® Xe Graphics - Windows

5.1 Optimizing and Deploying PyTorch* Models with OpenVINO™ [6,7,8]

OpenVINO™ has made it easy to import PyTorch*, TensorFlow*, TensorFlow Lite* (TFLite), ONNX*, and PaddlePaddle* models and integrate them into the inference pipeline. With recent updates to OpenVINO, developers can load models and deploy them using their native format. Figure 4 and Figure 5 show an example of how to import PyTorch models into OpenVINO or use OpenVINO as a backend in PyTorch via torch.compile. Use convert_model() API to convert the model to OpenVINO intermediate representation (IR) for optimal performance, shorter model loading time, and lighter runtime package. Alternatively, use torch.compile() API to use OpenVINO in PyTorch-native applications or for quick testing after model training/fine-tuning.

```
import openvino as ov
import torch
model = torch.load("model.pt")
model.eval()
ov_model = ov.convert_model(model) # Convert model loaded from PyTorch file
core = ov.Core()
compiled_model = core.compile_model(ov_model) # Compile model from memory
```

Figure 4: PyTorch* models can be directly converted within OpenVINO™

```
import openvino.torch
import torch
# Compile PyTorch model
opts = {"device" : "CPU", "config" : {"PERFORMANCE_HINT" : "LATENCY"}}
compiled_model = torch.compile(model, backend="openvino", options=opts)
```

Figure 5: OpenVINO™ backend to torch.compile

To use torch.compile, the user needs to add an import statement and define the backend as "openvino". With this backend, Torch FX subgraphs are directly converted to OpenVINO representation without any additional PyTorch based tracing.

[6] PyTorch Deployment via "torch.compile" — OpenVINO™ Documentation Version (2023.3)

[7] Importing PyTorch and TensorFlow Models Into OpenVINO | Medium

[8] OpenVINO Get Started Guide

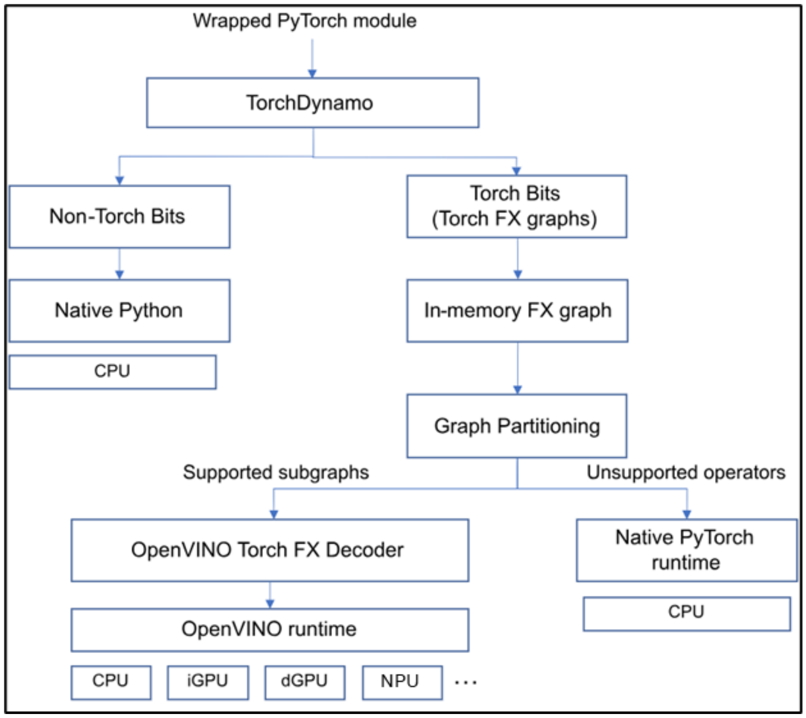

Figure 6: Flow using PyTorch* and torch.compile with OpenVINO™

PyTorch 2.0 introduced torch.compile, a feature that uses TorchDynamo to improve model performance. TorchDynamo dynamically modifies the code right before it is executed. It traces the PyTorch module and extracts sequences of operations into an FX Graph, which OpenVINO can then optimize.

Supported operations are grouped into OpenVINO submodules, converted to the OpenVINO graph using OpenVINO's PyTorch decoder, and executed efficiently on the target hardware with OpenVINO runtime. The native PyTorch runtime on CPU handles the unsupported operations. Target hardware can be the CPU, integrated GPU (iGPU), discrete GPU (dGPU), or NPU.

6.0 Intel® Arc™ A370E Benchmarks[9]

Table 2 shows the inference performance of the Intel® Arc™ A370M as a proxy for Intel® Arc™ A370E across various models targeting classification, detection, segmentation, and natural language processing. The inference performance is evaluated with Python*-based benchmark_app utility in Intel OpenVINO™ toolkit using two metrics - throughput and latency. Throughput is the amount of data an inferencing pipeline can process at once, measured in frames per second (fps). In applications where large amounts of data must be inferenced simultaneously (such as multi-camera video streams), high throughput is needed. Latency is the time it takes to process a single inference request. In applications where data needs to be inferenced and acted on as quickly as possible (such as autonomous driving), low latency is desirable.

While the performance data has been computed using Ubuntu* OS, the Intel Arc GPUs support Windows* as well.

#	Model Category	Model	Precision	Batch Size	Performance[10]
1	Image Classification	resnet-50-tf	FP16	32	906 fps Throughput
2	Image Classification	vgg19	FP16	32	273 fps Throughput
3	Object Detection	yolo-v3-tiny-tf	FP16	32	916 fps Throughput
4	Object Detection	ssd_mobilenet_v1	FP16	32	1381 fps Throughput
5	Instance Segmentation	yolov5s-seg	FP16	32	126 fps Throughput
6	Instance Segmentation	yolov5m-seg	FP16	32	75 fps Throughput
7	Pose Estimation	human-pose-estimation-3d-0001	FP16	32	380 fps Throughput
8	Question Answering	bert-large-uncased-whole-word-masking-squad-0001	FP16	1	45 ms Latency

[9] Performance measured on Intel® Arc™ 370M as proxy for Intel® Arc™ 370E. Results may vary.

[10] Average throughput and latency over three trials using OpenVINO benchmark_app in throughput mode and latency mode respectively.

#	Model Category	Model	Precision	Batch Size	Performance[10]
9	Speech Recognition	wav2vec2-base	FP16	1	15 ms Latency
10	Text Prediction	gpt-2	FP16	1	210 ms Latency

Table 2: Inference performance of Intel® Arc™ A370M GPU on various AI models[11]

The following step-by-step instructions enable users to reproduce the results in Table 2 and easily adopt Intel Arc GPUs for AI inference workloads.

6.1 Installation of Python*-based OpenVINO™ Development Tool

1. Create and activate a Python virtual environment.

Linux:

```
# create python venv (Linux)
$ python3 -m venv openvino_env
# activate venv (Linux)
$ source openvino_env/bin/activate
```

2. Install OpenVINO™ and dependencies for benchmarking (next section)

```
$ python -m pip install pip --upgrade
$ pip install openvino-dev[tensorflow2,pytorch,onnx]==2023.2.0
```

[11] **System Configuration:** Processor: Intel® Core™ i3-12100E Processor (4 Cores); Integrated GPU (iGPU): Intel® UHD Graphics 730; Discrete GPU (dGPU): Intel® Arc™ A370M Graphics; GPU Driver: 23.35.27191.42; OS: Ubuntu 22.04.3 LTS, Kernel: 6.5.0-17-generic; Memory: 16 GB; Python: 3.10.12; OpenVINO: 2023.2.0

6.2 Preparing the Models

Below is the list of commands used to download, optimize, and evaluate the models:

1. Image Classification: resnet-50-tf

```
$ omz_downloader --name resnet-50-tf

$ mo --framework=tf --output_dir=public/resnet-50-tf/FP16 --
model_name=resnet-50-tf --
input=map/TensorArrayStack/TensorArrayGatherV3 '--
mean_values=[123.68,116.78,103.94]' --output=softmax_tensor --
input_model=public/resnet-50-tf/resnet_v1-50.pb --
reverse_input_channels '--
layout=map/TensorArrayStack/TensorArrayGatherV3(NHWC->NCHW)' '--
input_shape=[1, 224, 224, 3]' --compress_to_fp16=True

$ benchmark_app -m public/resnet-50-tf/FP16/resnet-50-tf.xml -d GPU.1
-b 32 -hint throughput
```

2. Image Classification: vgg19

```
$ omz_downloader --name vgg19

$ omz_converter --name vgg19 --precisions FP16

$ benchmark_app -m public/vgg19/FP16/vgg19.xml -d GPU.1 -b 32 -hint
throughput
```

3. Object Detection: yolo-v3-tiny-tf

```
$ omz_downloader --name yolo-v3-tiny-tf

$ omz_converter --name yolo-v3-tiny-tf --precisions FP16

$ benchmark_app -m public/yolo-v3-tiny-tf/FP16/yolo-v3-tiny-tf.xml -d
GPU.1 -b 32 -hint throughput
```

4. Object Detection: ssd_mobilenet_v1

```
$ omz_downloader --name ssd_mobilenet_v1_coco

$ omz_converter --name ssd_mobilenet_v1_coco --precisions FP16

$ benchmark_app -m
public/ssd_mobilenet_v1_coco/FP16/ssd_mobilenet_v1_coco.xml -d GPU.1
-b 32 -hint throughput
```

5. Instance Segmentation: yolov5s-seg

```
$ git clone https://github.com/ultralytics/yolov5.git -b v7.0

$ cd yolov5

$ wget
https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s-
seg.pt

$ python3 export.py --weights yolov5s-seg.pt --imgsz 640 --batch-size
32 --include ONNX

## Create a new Python script optimize_yolov5s-seg.py with the
following

import openvino as ov

MODEL_NAME = "yolov5s-seg"

onnx_path = f"{MODEL_NAME}.onnx"

fp16_path = f"yolov5s-seg_FP16_openvino_model/{MODEL_NAME}_fp16.xml"

model = ov.convert_model(onnx_path)

ov.save_model(model, fp16_path, compress_to_fp16=True)

$ python3 optimize_yolov5s-seg.py

$ benchmark_app -m yolov5s-seg_FP16_openvino_model/yolov5s-
seg_fp16.xml -d GPU.1 -b 32 -hint throughput
```

6. Instance Segmentation: yolov5m-seg

```
$ git clone https://github.com/ultralytics/yolov5.git -b v7.0

$ cd yolov5

$ wget
https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5m-
seg.pt

$ python3 export.py --weights yolov5m-seg.pt --imgsz 640 --batch-size
32 --include ONNX

## Create a new Python script optimize_yolov5m-seg.py with the
following

import openvino as ov

MODEL_NAME = "yolov5m-seg"

onnx_path = f"{MODEL_NAME}.onnx"

fp16_path = f"yolov5m-seg_FP16_openvino_model/{MODEL_NAME}_fp16.xml"

model = ov.convert_model(onnx_path)

ov.save_model(model, fp16_path, compress_to_fp16=True)

$ python3 optimize_yolov5s-seg.py

$ benchmark_app -m yolov5m-seg_FP16_openvino_model/yolov5m-
seg_fp16.xml -d GPU.1 -b 32 -hint throughput
```

7. Pose Estimation: human-pose-estimation-3d-0001

```
$ omz_downloader --name human-pose-estimation-3d-0001

$ omz_converter --name human-pose-estimation-3d-0001 --precisions
FP16

$ benchmark_app -m public/human-pose-estimation-3d-0001/FP16/human-
pose-estimation-3d-0001.xml -d GPU.1 -b 32 -hint throughput
```

8. Question-Answering: bert-large-uncased-whole-word-masking-squad-0001

```
$ omz_downloader --name bert-large-uncased-whole-word-masking-squad-
0001 --precisions FP16

$ benchmark_app -m intel/bert-large-uncased-whole-word-masking-squad-
0001/FP16/bert-large-uncased-whole-word-masking-squad-0001.xml -d
GPU.1 -hint latency
```

9. Speech Recognition: wav2vec2-base

```
$ omz_downloader --name wav2vec2-base

$ omz_converter --name wav2vec2-base --precisions FP16

$ benchmark_app -m public/wav2vec2-base/FP16/wav2vec2-base.xml -d
GPU.1 -hint latency
```

10. Text Prediction: gpt-2

```
$ omz_downloader --name gpt-2

$ omz_converter --name gpt-2 --precisions FP16

$ benchmark_app -m public/gpt-2/FP16/gpt-2.xml -d GPU.1 -data_shape
[1,1024] -hint latency
```

7.0 Case Study: Product Classification and Person Detection

7.1 Overview

This case study demonstrates the accelerated end-to-end performance of product classification and person detection use cases while offloading the inference execution to the Intel® Arc™ A370M GPU. Note that the Intel® Arc™ A370M is used as proxy for the Intel® Arc™ A370E. As shown in Figure 7, the end-to-end pipeline involves video decoding, image pre-processing (such as color conversion or image resizing), inference execution and post-processing (optional, such as writing the inference results to the video frame or database). Additionally, all code snippets and software dependencies are highlighted with step-by-step instructions to enable anyone looking to get started with running the AI workloads on Intel Arc A370M or Intel Arc A370E GPUs.

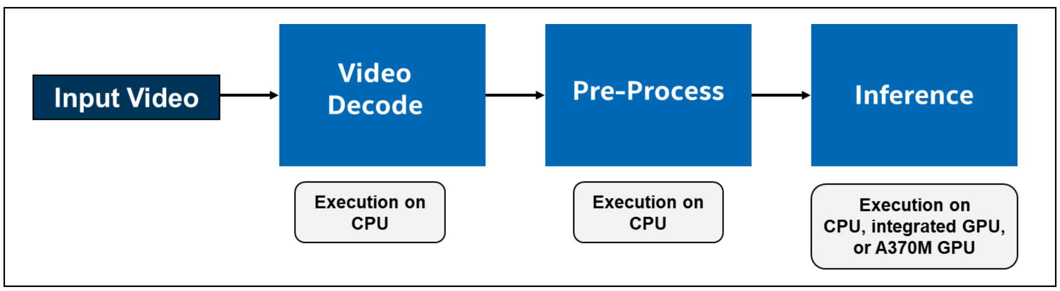

Figure 7: End-to-end workflow or pipeline for product classification/person detection use cases[12,13]

The image classification and object detection demos from Open Model Zoo repository are used with resnet-50-tf and yolo-v3-tiny-tf models to evaluate the end-to-end performance of the host CPU, integrated GPU and A370M discrete GPU. Following are the steps to run the two demos.

1. Open a new terminal and activate the Python* virtual environment (openvino_env) created in section 6.1.

```
$ source openvino_env/bin/activate
```

[12] **System Configuration**: Processor: 12th Gen Intel® Core™ i3-12100E; Integrated GPU (iGPU): Intel® UHD Graphics 730; Discrete GPU: Intel® Arc™ A370M; OS: Ubuntu 22.04 LTS (Kernel version: 6.5.0-17-generic); Intel OpenVINO Toolkit: 2023.2

[13] Performance measured on Intel® Arc™ 370M as proxy for Intel® Arc™ 370E. Results may vary.

2. Download the Open Model Zoo repository and extract the folder.

3. Download the input video and prepare the models according to section 6.2.

4. Run the image classification demo.

```
$ cd open_model_zoo\demos

$ python3 classification_demo/python/classification_demo.py -i
<fruit-and-vegetable-detection.mp4> -m <public/resnet-50-
tf/FP16/resnet-50-tf.xml> -d GPU.1 --labels
../data/dataset_classes/imagenet_2012.txt
```

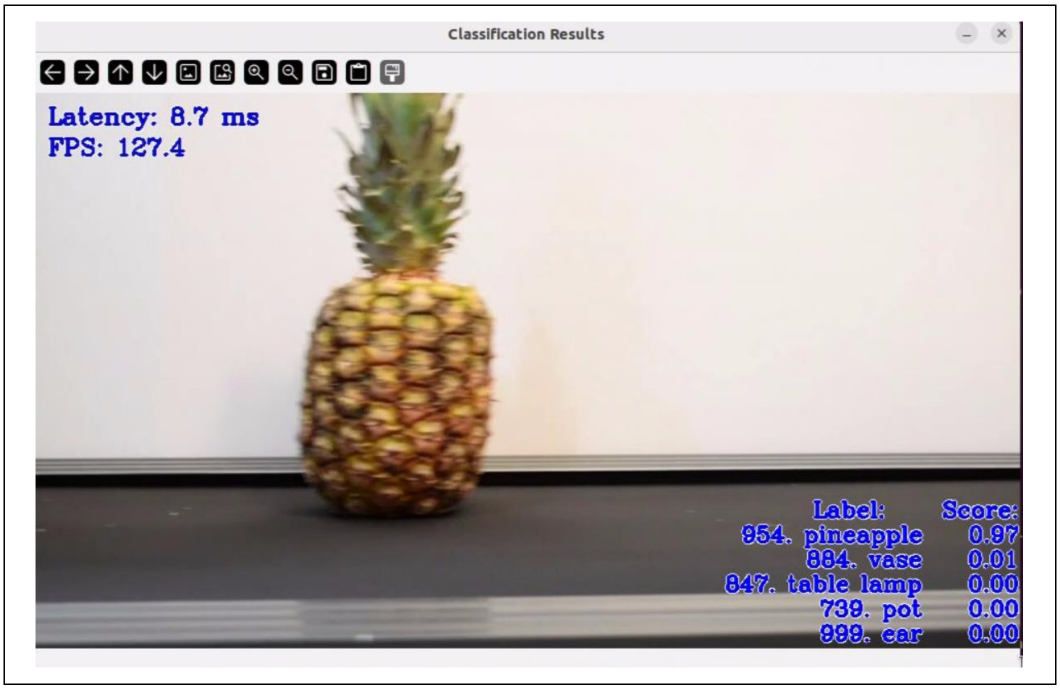

Figure 8: Sample product classification output on Intel® Arc™ A370M GPU

5. Download the input video and run the object detection demo.

```
$ python3 object_detection_demo/python/object_detection_demo.py -m
<public/yolo-v3-tiny-tf/FP16/yolo-v3-tiny-tf.xml> -i <face-
demographics-walking-and-pause.mp4> -at yolo --labels
../data/dataset_classes/coco_80cl.txt -d GPU.1
```

Note: Change the parameter *d* to *CPU* or *GPU.0* to run the demo on CPU or integrated GPU in the host.

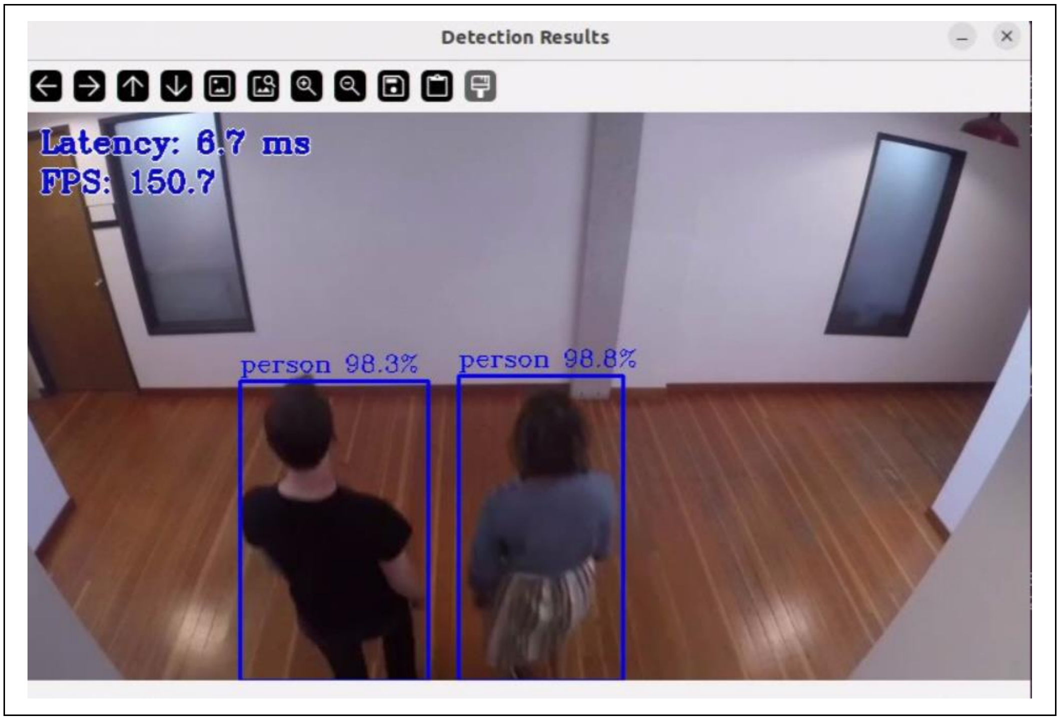

Figure 9: Sample person detection output on Intel® Arc™ A370M GPU

Person detection models play a vital role in enhancing safety, security, and efficiency across various domains. For example, retail stores can use person detection models to analyze foot traffic, customer demographics, and prevent shoplifting. Additionally, person detection models in conjunction with product classification and pose estimation models enable use cases such as frictionless checkout.

7.2 Performance

The end-to-end performance of the product classification and person detection use cases are evaluated by adding the input parameter *--no_show* to the script in steps 4 and 5 to avoid displaying the inference results. The results in Table 3 show the end-to-end throughput of the two use cases on the host CPU, integrated GPU, and A370M GPU (used as a proxy for A370E GPU). In all cases video decode is done on the CPU. These results show that performance on Intel® Arc™ A370M GPU is up to 5x higher compared to CPU and up to 4x higher compared to integrated GPU. Note that the difference in inference results between Table 2 and Table 3 is that Table 2 shows the inference performance of AI models in isolation, whereas Table 3 shows the end-to-end performance of AI models including video decoding, pre-processing, and inferencing.

Use Case	Precision	Inferencing Device	End-to-End Throughput[14] (fps)
Product classification on an end-to-end pipeline with resnet-50-tf (Input video resolution: 960x540, frame rate: 60fps)	FP16	CPU	54
		Integrated GPU	71
		Arc A370M	313
Person detection on an end-to-end pipeline with yolo-v3-tiny-tf (Input video resolution: 768x432, frame rate: 12fps)	FP16	CPU	77
		Integrated GPU	105
		Arc A370M	391

Table 3: End-to-end performance of product classification and person detection[15,16]

[14] Throughput represents the end-to-end throughput averaged over three trials; CPU: Intel Core i3-1200E, iGPU: Intel UHD Graphics 730, dGPU: Intel Arc A370M

[15] Performance may vary based on system configurations such as OS and GPU/NPU drivers for same target hardware.

[16] **System Configuration**: Processor: 12th Gen Intel® Core™ i3-12100E; Integrated GPU (iGPU): Intel® UHD Graphics 730; Discrete GPU: Intel® Arc™ A370M; OS: Ubuntu 22.04 LTS (Kernel version: 6.5.0-17-generic); Intel OpenVINO Toolkit: 2023.2

8.0 Conclusion

The Intel® Arc™ GPUs, based on innovative X^{e}-cores and leveraging the OpenVINO™ software ecosystem, offer a range of power-optimized and performance-optimized solutions for edge AI workloads. This white paper explores a case study on image classification and object detection use cases in detail. This case study demonstrated how Arc GPUs and OpenVINO empower developers to leverage hardware capabilities effectively and with ease. The versatility of Arc GPUs extends beyond object detection and image classification use cases, offering immense potential for various applications across diverse sectors such as retail, healthcare, smart cities, and robotics.

While this paper discusses a handful of benchmarks, additional performance benchmarks are in the works and will be published on the OpenVINO site (OpenVINO™ Performance Benchmarks). This white paper has explored the AI capabilities of Intel Arc GPUs, demonstrating Intel's commitment to hardware and software innovation, and pushing the boundaries of AI capabilities at the Edge.

Intel invites AI developers and enthusiasts to join the growing Intel Arc GPU community and explore the immense potential of this family of GPUs. Together, we can unlock groundbreaking possibilities across the diverse Edge AI use cases.

9.0 References

[6] PyTorch Deployment via “torch.compile” — OpenVINO™ Documentation Version (2023.3)

[7] Importing PyTorch and TensorFlow Models Into OpenVINO | Medium

[8] OpenVINO Get Started Guide

10.0 Additional Information

- Intel® Arc™ GPU for the Edge
- Intel AI in Production Success Stories
- OpenVINO™ Toolkit Overview
- Innovate in Healthcare and Life Science with Intel Graphics Solutions
- AI-boosted endoscopy solution powered by Intel
- AI in the Operating Room Starts with Intel
- Intel® Arc™ Graphics Empowers Medical Imaging AI Inference Solution
- Heterogeneous AI Powerhouse: Unveiling the Hardware and Software Foundation of Intel® Core™ Ultra Processors for the Edge