White Paper

Visual Technologies
Multi-device Enabling

Unlocking the Power of Intel® Deep Link

Part One: Client Artificial Intelligence (AI) Using Intel® GPUs

This paper is part one in a series of white papers designed to provide details regarding openly available development tools that can be used to take full advantage of Intel® Deep Link Technology.

Together, these papers will introduce and demonstrate some of the tools and processes that can be used to leverage Deep Link and allow developers to build better performing and more efficient applications. Included will be use cases that showcase Deep Link's current and future potential.

This first paper focuses on applying Deep Link concepts for Client AI using the OpenVINO™ Toolkit, and presents two use cases: Gigapixel AI by Topaz Labs, and an AI-powered video effects process which utilizes multiple neural network models.

Authors

Roman Borisov
Senior Software Application Engineer

Max Domeika
Principal Engineer

Ajith Illendula
Senior Software Application Engineer

Nick Yang
Senior Software Application Engineer

Table of Contents

Introduction

With the release of the Intel® 11th Generation mobile processor and the Intel® Iris® Xᵉ and Intel® Iris® Xᵉ MAX graphics architecture, Deep Link was introduced to the world and a new era of innovation was born.

Developers now have the ability to strategically apply computing power that was previously unavailable, and to assign tasks to parts of the machine which would otherwise just lie dormant. Imagine having the ability to significantly boost the performance of your application using not much more than a strategic approach and some lines of code.

That is the power of Deep Link.

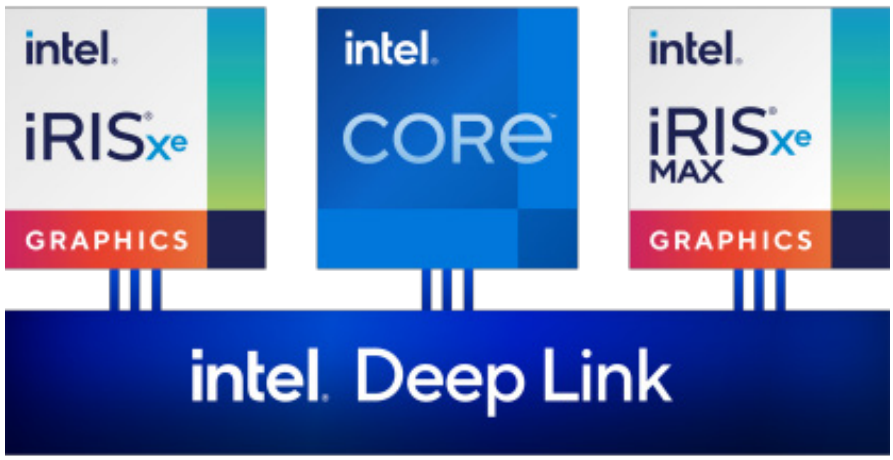


Figure 1. Intel® Deep Link combines an 11th Generation processor with an Iris® Xᵉ integrated GPU and an Iris® Xᵉ MAX discrete GPU, and manages the use of multiple GPUs simultaneously.

Deep Link Technology

At its core, Intel® Deep Link Technology is a reimagining and reshaping of the way that a machine's Central Processing Unit (CPU) and Graphics Processing Units (GPUs) interact. Already available on a number of devices, Deep Link offers the ability to combine the computing power of a discrete GPU with that of a powerful integrated GPU.

Using Deep Link, complex workloads and pipelines can be constructed which use multiple computing elements with a high degree of code re-use, thereby simplifying code development and reducing overall effort. This approach offers significant gains in both performance and efficiency, boosting the functional capabilities of a given application by offering expanded computing capabilities and processing options.

Deep Link Puts Multiple GPUs to Work

By partitioning computational elements into logical segments, Deep Link GPUs can equitably share the workload - each one working independently from and concurrently with the other. The Intel® Iris® Xᵉ (also referred to as the integrated, or iGPU) and Intel® Iris® Xᵉ MAX (also referred to as the discrete, or dGPU) can be used together to equitably split tasks, often allowing a workload to be completed in roughly half the time.

Deep Link Gets the Most out of Intel® Hardware

Deep Link enables Intel® computing components to work together at a level of speed and efficiency that was not available before by combining the computing power of multiple GPUs with similar characteristics. Because the two graphics processors use the same kernel code and have similar computing power and performance characteristics, there is little additional overhead introduced when partitioning tasks between the two.

OpenVINO™ (Open Visual Inference Neural network Optimization)

Intel® has created a number of software tools, processes and methodologies to ease implementation. Using these tools, developers can take full advantage of Deep Link Technology and build a solution that efficiently utilizes all of the computing power available to the system without favoring or ignoring any components - whether integrated or discrete.

One of those tools is the OpenVINO™ Toolkit.

The OpenVINO™ Toolkit was developed by Intel® and released to the open source community in May of 2018. Two versions of the toolkit are available today; one which is fully open source and a second which is distributed and supported by Intel®.

The toolkit was designed to help developers save time, energy and resources by providing the tools to build deep learning and Artificial Intelligence (AI)-driven applications that not only perform much better, but are also easier to create. It does this in a number of different ways, including:

Heterogeneous Execution (write once - deploy anywhere)
OpenVINO™ allows developers to write a segment of code one time, and then deploy a device-specific iteration of the same code across multiple computational components (CPU, iGPU, dGPU, Vision Processing Unit, FPGA, etc.).

Intuitive Workflow
The workflow that it utilizes was built from the ground up to allow OpenVINO™ to be an efficient and comprehensive development and deployment platform.

Workload Sharing
OpenVINO™ enables the assignment of workloads across the entire computational platform, preventing the overloading and bottle-necking that plagues many applications.

Tools to Speed Time-to-Market
Intel® offers a full suite of functions and pre-optimized kernels that are compatible with OpenVINO™ and can be used to execute various tasks in any number of applications. This includes direct calls to kernels for OpenCV, OpenCL™, and more.

Accelerated Deep Learning
The efficiencies and opportunities for expansion that are available to the entire platform because of OpenVINO™ allow for faster and more accurate pattern recognition and AI inference.

Use Case: AI-powered Image Upscaling with Topaz Gigapixel AI

Combining Intel® Deep Link Technology with the OpenVINO™ toolkit opens up a whole new world of applications that are now able to rely on faster processing and easier integration of deep learning data sets. As an Intel® development partner, Topaz Labs recognized early on that this unique combination of speed and computing power would be a huge benefit to its image upscaling platform known as Gigapixel AI.

Image upscaling using AI is a new technology which allows small, grainy, or even out of focus images to be enlarged and sharpened in such a way that they can be displayed and printed without distortion - even in much larger formats.

The Benefit of Applying AI to Image Upscaling

Image upscaling has been around for a long time. For well over a hundred years the easiest and most economical way to enlarge a photograph was to simply take another photo of the subject photograph at a higher resolution. The limitations of this approach are obvious, and there was always some loss of detail as compared to the original image. Then came the digital age which presented an opportunity to enlarge and sharpen photographs using an algorithm that would be applied to each pixel individually. The characteristics of the adjacent pixels were used to decide the proper characteristics of each new or changed pixel. The drawback to this method (and the main limiting factor to its effectiveness) was that the same stoic algorithm was applied to each pixel throughout the image.

Advances in AI and data processing, though, have opened new doors into the application of deep learning logic as it relates each individual pixel in an image. No longer is there a necessity for a single pixel's characteristics to be determined by its immediate neighbors; each pixel can now be optimized to fulfill its ideal role in the overall image.

Using this approach, not only can you enlarge an image (up to 600%!) without losing detail, but enlargement, sharpening and enhancement of the image can be applied at the same time.

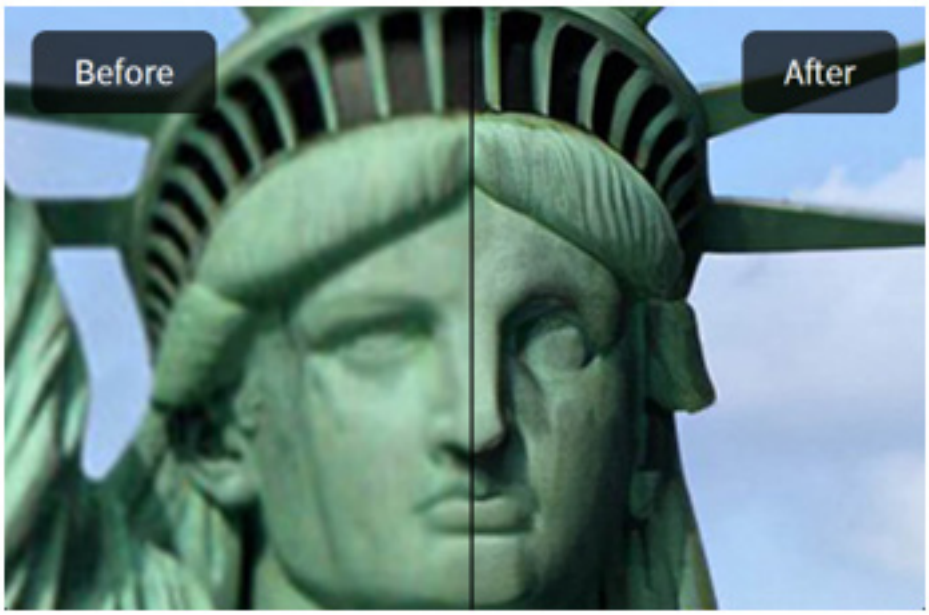


Figure 2. Photo of the Statue of Liberty before and after being processed using Gigapixel AI.

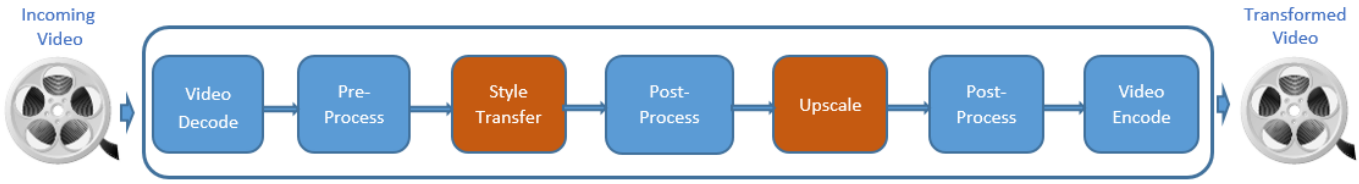
The deep learning aspect of Gigapixel AI is applied as the software is exposed to multiple images of a given subject. For instance, with every image that the software analyzes that includes a human face, it learns more about how the average human face is shaped, how it is colored, how it reflects light, and many other defining characteristics. It also learns what human hair looks like as it lays across the skin, as well as what it looks like when it is curly, or straight, or in a ponytail. After analyzing a series of photographs containing any subject matter, the AI can apply the information that it has collected to enlarge, enhance and sharpen photos with similar subjects.

Use Case: AI-powered Video Effects Using Multiple CNN Models

Processing video content has always had a way of pushing the limits of computing capabilities. By its very nature, video processing requires a high degree of computing power due to the large number of frames that have to be processed (24 - 60 frames / second of HD video content, for example). Using the increased computing power made available by Deep Link and the ease of incorporation afforded by OpenVINO™, multiple Convolutional Neural Networks (CNNs) can be added to a single transformation process. The end result is a significant reduction in the time required to fully transform video content in processes that require the use of multiple CNNs - especially for large-scale video processing projects.

In building a video transformation pipeline, one or more CNN models can be utilized (along with other pre- and post-process modification stages) to process the content in a single pass using multiple GPUs, which can significantly shorten the time required to reach a completed output.

In the example shown below, multiple CNNs (Style Transfer and Upscale) are incorporated into a typical video transformation process, along with pre- and post-processing and CODEC stages. This pipeline shows a common process involving style transfer of the original video clip, followed by an upscaling operation used to improve the image quality of the newly stylized clip. This process is typical of one which might be found in a video editing application.



This pipeline can be implemented using OpenVINO™ to run the CNN models and OpenCV to perform the video decoding/encoding in addition to the pre- and post-processing operations (resizing, cropping, padding, etc.).

Using OpenVINO™, it is possible to select which of the available compute devices is best used to run inference with each CNN. One device - CPU, iGPU or dGPU - can be used to run both CNN models, or the more efficient path may be to run the Style Transfer model on one device and channel the Upscale model through another.

A test implementation of the pipeline (with the pre- and post-processing and CODEC stages being performed by the CPU, upscaling performed by the iGPU, and style transfer performed by the dGPU) resulted in the following performance and device utilization figures:

Device Utilization	FPS	iGPU Utilization	dGPU Utilization
Both CNNs Run on iGPU	8.7	90%	n/a
Both CNNs Run on dGPU	6.9	n/a	100%
Upscale Run on iGPU and Style Transfer Run on GPU	10.4*	67%	36%

Table 1. Device utilization statistics for multi-device test implementation.

When processing video content through two separate GPUs, it is important ensure that the two devices are able to process their inputs concurrently. There are two important considerations to making sure that this is possible:

1. Ensure that the writing of the upstream stage output does not block the reading of the downstream stage input. This is done by making sure that there are sufficient buffers/frames allocated to storing the intermediate results.
2. Inference should be running asynchronously for both models. There are two ways to make sure that this is the case, using either the OpenVINO™ Synchronous or Asynchronous Application Programming Interface (API):
   a. Use synchronous API to start inference but in separate threads for the different models, or
   b. Perform all functions in one thread, but use the asynchronous API.

* - The theoretical maximum Frames-per-Second (FPS) performance for Deep Link in this configuration would be the sum of the performance of each individual GPU (in this case 8.7 + 6.9 = 15.6 FPS), but it is clear from the table that when two GPUs are engaged simultaneously both GPUs are under-utilized.

Changing the workload partitioning and using the GPU(s) to perform all of the pre- and post-processing (using CV::UMat abstraction) would increase efficiency and increase the FPS rate substantially. In addition, decoding directly to video memory and encoding directly from video memory to avoid unnecessary data transfer would increase efficiency, but this is not currently supported by OpenCV. Intel™ Media SDK (which will be the subject of a future paper in this series) can be used for this purpose.

What Problems Does OpenVINO™ Solve?

There seems to be an ever-expanding number of options for deep learning frameworks used in our industry, each with its own data and information modeling structure used for inference. Beyond that, Intel® offers a wide range of devices and computing platforms that each has its own specific architecture, instruction set and programming model. For an individual developer - or even a team of developers - to take the time to learn all of the various platforms and programming variables necessary to match each inference model to each device would be a tremendous undertaking and require significant effort and coordination.

Luckily, OpenVINO™ does it for you.

Which Models Does OpenVINO™ Support?

Using a common API, OpenVINO can import a trained model from any of the more common deep learning frameworks (TensorFlow, Caffe, MXNet, Kaldi and ONNX are currently supported, with more frameworks to follow) and implement inference across any supported device, eliminating the need to create specific instances of code for each device type.

How Does OpenVINO™ Work?

OpenVINO™ works by importing a trained model from a deep learning framework and processing it first through its Model Optimizer, then through its Inference Engine.

Model Optimizer

The output (presentation files) from the trained model from one deep learning platform to the next can vary significantly. The OpenVINO™ Model Optimizer analyzes these output files and transforms them by creating the files needed by the inference engine. It does this in two steps:

1. Conversion
   The files that are output from the trained model must be converted to a format which can be used by the inference engine. No matter what file types were input, the output format of the converted files will be:
   a. One .xml file (describes the topologies, layers, connectivity, parameters, etc.)
   b. One .bin file (contains all weights and biases)
2. Optimization
   Once the files have been converted, they are further modified by the Model Optimizer in ways that will be useful to the Inference Engine. This is achieved by sorting and combining like data and tasks, which in turn reduces the overall computational and memory space requirements.

Inference Engine

Once passed from the Model Optimizer, the data housed in the two files is customized to suit each specific device and then implemented using the Inference Engine. The Inference Engine is a unified API which is integrated within the application. It is designed to ease implementation and allow for high performance with minimal input from the developer

Once a target device is specified, the Inference Engine utilizes plugins and access to various existing CNN libraries to build an implementation of each operation specific to that device.

Although it does support a wide range of devices, the Inference Engine is optimized for 11th Gen Intel® processors (including the integrated graphics processor) and Xe-based graphics processors.

Getting Started

Intel® Deep Link Technology and OpenVINO™ provide a solid foundation for building a great application, but the work really only begins there - getting maximum results requires some planning and execution.

Following are the general steps that should be taken early in the development process to ensure that you get as much out of Deep Link as possible in your specific implementation. For specific instructions regarding installation on Windows-based or Linux-based systems, please visit the OpenVINO™ site (link provided in the Online Resources section).

OpenVINO™ Installation and Validation

Installation Prerequisites

In order to ensure proper function, there are a number of prerequisites which must be accomplished or verified prior to installing and using OpenVINO™:

- *Download and install the latest Intel® Iris® Xe MAX GPU drivers (link provided in Online Resources section)*
- *Visual Studio 2019 (Community Edition will work, but requires all of the C++ modules which are not included with standard Unity VS install package)*
- *CMake (latest version will work)*
- *Python™ (any version between 3.5 - 3.8; 3.9 is not supported)*
  - *Must select 'Add Python to the PATH' when prompted*

Installation

Download and run the latest version of the OpenVino™ installer (link provided in Online Resources section).

When you start the OpenVINO™ installer, it should provide a link to the install documentation. If not, a link is provided in the Online Resources section.

As stated in the documentation, be sure to run *setupvars* at the command prompt to set the appropriate environment variables.

When configuring Model Optimizer (for all platforms), the latest version of Python™ installer (pip) is not compatible. Line 81 of the *install_prerequisites* file must be changed to:

pip3 install --user --use-deprecated=legacy-resolver -r .\requirements%postfix%.txt

Validation

After installation is complete, run *demo_squeezenet_download_convert_run* from the installation directory. If there are no errors, you'll know your OpenVINO™ install is valid.

NOTE: The *squezenet* test demo runs the inference on the CPU. To target the Intel® Iris® Xe MAX, change line 7 of *demo_squeezenet_download_convert_run* to:

set TARGET=GPU.1

Testing OpenVINO™ with Deep Link

After OpenVINO™ installation has been validated, you can progress to leveraging Deep Link with one of the freely available optimized deep learning models available online.

Download the *pedestrian_tracker_demo* from Model Zoo using the following link:

https://github.com/openvinotoolkit/open_model_zoo/tree/master/demos

Additionally, download the *person-detection-retail-0013* and *person-reidentification-retail-0031 inference models* from the models section on Model Zoo using the following link:

https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/intel

Assuming that the system has been configured correctly, a demonstration which utilizes both GPUs simultaneously can now be executed by running the pedestrian tracker demo. Set separate GPU targets for each of the two inference models and execute by running the following command (assumes Windows install):

C:\Program Files (x86)\Intel\openvino_2021\deployment_tools\demo\pedestrian_tracker_demo -i <path_video_file> -m_det <path_to_model>\person-detection-retail-0013.xml -m_reid <path_to_model>\person-reidentification-retail-0031.xml -d_det GPU.1 -d_reid GPU.0

Implementation: Putting Intel® GPUs to Work for Your Application

Step One: Identify the GPUs in Your System

The process required for performing this step will differ depending on the platform and operating system; shown below are typical methods that could be used to collect information regarding installed GPUs:

Function:	Intel oneAPI/DPC++	OpenCL	Intel C-for-media	DirectX
Identify all system GPUs	sycl::platform::get_platforms() platform.get_devices()	See clinfo utility clGetPlatformInfo() clGetDeviceInfo()	CreateDXGIFactory1() IDXGIFactory1::EnumAdapters1()	CreateDXGIFactory1() IDXGIFactory1::EnumAdapters1()
Get GPU general attributes	See get_info() and sycl::info::platform	See cl_device_info	IDXGIAdapter1::GetDesc1()	IDXGIAdapter1::GetDesc1()
Get GPU vendor specific attributes	See extensions in get_info() and sycl::info::platform	See cl_device_info and CL_DEVICE_EXTENSIONS	GetCmSupportedAdapters() QueryCmAdapterInfo() CmDev::GetCaps()	
Obtain/Create device	See sycl::device	See clGetDeviceIDs()	CreateCmDeviceFromAdapter()	D3D11CreateDevice()

Table 2. API-specific functions used to collect information regarding GPU classifications and characteristics.

Following these steps will return information that will be useful to the developer in assigning tasks and partitioning data, such as:

- GPU Description, Device ID, Subsystem ID and Revision
- Maximum Number of Threads, Number of Slices/Subslices, Total Number of Execution Units
- Available Memory, Dedicated Memory and Shared Memory
- Min/Max Frequency

Step Two: Partition the Workload

Depending on the application, workload elements can be partitioned (segregated or separated to be assigned to a particular system component) in any number of ways, including:

Partitioning by Data Type

- Image and Video Processing
- Deep Learning Inferencing
- Encoding/Decoding

Partitioning by Order/Operation

- Sequential: One GPU performs the initial operation, then passes the result to a second GPU which performs the second operation.
- Parallel: Workloads with multiple independent operations (like facial detection/recognition and gender determination/emotion detection) can be carried out simultaneously by utilizing multiple GPUs concurrently.

Partitioning by Batch Size

- Processing of a single-frame image (batch size = 1): one GPU processes the top half of the image as the second GPU processes the bottom half of the image simultaneously (the image could also be partitioned into smaller segments, such as part of a line, a full line, or multiple lines, with data being assigned to each GPU in a round-robin fashion).
- Processing of a multiple-frame data set (batch size > 1): any number of images can be processed in parallel, with the workload being partitioned based on predetermined factors (i.e., one GPU works odd-numbered frames while the other processes even-numbered frames, etc.).

Step Three: Allocate System Resources

One of the most important considerations when building an application that takes advantage of all that Deep Link and OpenVINO™ have to offer is proper allocation of the available resources. Along with the additional processing paths and increased efficiency in the use of multiple devices comes the requirement that a number of important questions be asked and answered. These questions might include:

- Is the effort required to build a multi-device processing solution even worth exploring in the first place?
- Is simply splitting the workload evenly among components sufficient, or will strategically partitioning the workload provide better efficiency?
- Which algorithms are best performed by the CPU vs. the iGPU? vs. the dGPU?
- Where are the processing hotspots in the application?
- Are there opportunities to process tasks in parallel?
- What levels of increased power consumption will I have to deal with by running multiple processors?

The answers to these questions will be different for each application, but researching and determining the answer can reap large benefits in the end. Understanding the hardware and the underlying development platform is an important part of building an effective strategy for processing and storing the data used by your application.

Step Four: Execute!

Platform Considerations

When plotting out a path for handling the workload requirements for your application there are many factors to consider. There is no way around adding some processing overhead when splitting a workload between two (or more) processors, but it can be reduced to a negligible level with careful planning, distribution and management of the workload. Keep all of the following factors in mind when laying out the computational structure of your application.

The CPU must handle its own workload in addition to coordinating the GPU workload.
The CPU is the host and directs work to/from the GPUs. If the CPU is saturated with compute work it may not be able to adequately perform its host functions, causing the GPU processes to bog down. Find the right balance of CPUs dedicated to compute work vs. acting as host.

Other client processes may be employing the GPU for processing.
GPUs are employed primarily to handle images and visuals on the screen, but can also be employed for other actions such as background memory scan (like Intel® Threat Detection Technology). Understanding when and how the system engages the GPU for other client processes prior to dispatching workloads is important.

There may be performance differences when running on battery versus AC.
By default, most systems will lower performance when operating under battery power. As a result, a workload distribution strategy that works well under AC power may not work as well under battery power. Consider this difference and factor it into your workload distribution plan as necessary.

GPUs can provide compute performance.
In a typical system the CPU will provide better latency and general-purpose compute capability, but the GPUs will often outperform the CPU on high throughput workloads. In addition, CPUs can become oversubscribed, and offloading some functions to the iGPU or dGPU can provide higher overall performance (completing more work in the same amount of time).

GPU data transfer times can impact performance.
Some GPUs contain memory that is distinct from the CPU and may require large data transfers. This could diminish (or even eliminate) the performance gains made by taking advantage of the compute performance of the GPU. Factor in the size of the computation task vs. the cost of the time required to perform data transfers to/from GPU memory.

Copying data between system and video memory can adversely effect performance.
When allocating resources, keep in mind that buffers and surfaces are allocated in the video memory for best memory I/O performance. Copying data between system memory and video memory or between iGPU video memory and dGPU video memory can be slow, and frequent copying of such data between two GPUs should be avoided.

GPUs have memory limits.
Some GPUs have memory that is distinct from CPU and system memory. Some GPUs share memory with the CPU, but there are typically performance costs associated with managing the memory. Understanding the application data structure needs and comparing them to GPU and CPU memory limits is an important part of the overall process.

Software libraries are not all created equal.
The selection of software libraries to be utilized and the ways that they are implemented will have an impact on application performance. Intel® distributions of several of the more commonly-used libraries are designed specifically to work with Intel® hardware and will show improved performance when compared to libraries implemented over hardware from other manufacturers.

There may be performance differences between accelerators.
Distributing the work can result in higher overall performance, but performance may be limited by the slowest compute. It may be advantageous to reduce the workload given to some accelerators, as slower compute resources in the system can become a bottleneck. It is important to find the right balance in distributing the workload using accelerators.

Online Resources

Link to download updated GPU drivers:
https://downloadcenter.intel.com/download/30078/Intel-Iris-Xe-MAX-Dedicated-Graphics-Drivers

Link to download OpenVINO™ Installer:
https://software.seek.intel.com/openvino-toolkit

Link to download OpenVINO™ Installation guides:
https://docs.openvinotoolkit.org/2021.2/installation_guides.html

OpenVINO™: How It Works
https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/usage.html

Conclusion

Used separately, Intel® Deep Link Technology and the OpenVINO® Toolkit are both very powerful tools in their own right. Used together, these two development tools open up a whole new world of possibilities. Whether you are building an application from scratch or attempting to build additional functionality or efficiencies into an existing application, the development process will be smoother and more effective when these two tools are utilized together.

It is our sincere hope that you have found the information in this white paper helpful and informative. Future papers in this series promise to bring to light other tools that can be paired with Deep Link to provide exceptional processing speed and performance for your new and existing applications. From dynamic power sharing to accelerated video rendering and much more, the opportunities for performance improvement are nearly endless.