Deep Learning: Training vs. Inference

Training

- **Human**
- **Bicycle**
- **Strawberry**

Lots of Labeled Data!

Inference

- **Bicycle**

Did You Know?

Training requires a very large data set and deep neural network (many layers) to achieve the highest accuracy in most cases.

Accuracy vs. Data Set Size

- Large NN
- Medium NN
- Small NN
- Traditional Model
How do you determine the right computing for your AI needs?

**WORKLOADS**

1. What is my workload profile?
2. What are the characteristics of my use case?
3. How prevalent is AI in my environment?

**REQUIREMENTS**

- Security
- Flexibility
- Scalability
- Efficiency
- Management
- Accuracy
- Data

**DEMAND**

- Utilization
- Scale

- ASIC/FPGA/GPU
- CPU
Intel® Distribution of OpenVINO™ Toolkit

- Tool Suite for High-Performance, Deep Learning Inference
- Fast, accurate real-world results using high-performance, AI and computer vision inference deployed into production across Intel® architecture from edge to cloud

High-Performance, Deep Learning Inference
Streamlined Development, Ease of Use
Write Once, Deploy Anywhere

- Enables deep learning inference from the edge to cloud.
- Speeds time-to-market through an easy-to-use library of CV functions and pre-optimized kernels.
- Includes optimized calls for CV standards, including OpenCV* and OpenCL™.
Three steps for the Intel® Distribution of OpenVINO™ toolkit

1. **Build**
   - Trained Model
   - Open Model Zoo
     - 100+ open sourced & optimized pre-trained models available
   - Deep Learning Workbench
     - Visually analyze and fine-tune (NEW) available on Intel® DevCloud for the Edge as a Beta release

2. **Optimize**
   - Model Optimizer
     - Converts and optimizes trained model using a supported framework
   - Post-Training Optimization Tool
     - Reduces model size into low-precision without re-training
   - Intermediate Representation (xml, .bin)
   - Read, Load, Infer

3. **Deploy**
   - Inference Engine
     - Common API that abstracts low-level programming for each hardware
   - Deep Learning Streamer
     - Code Samples & Demos (e.g. Benchmark app, Accuracy Checker, Model Downloader)
   - Model Server
     - gRPC/REST Server with C++ backend
   - Deployment Manager
     - [NEW] Available on Intel® DevCloud for the Edge as a Beta release
Supported Frameworks

Breadth of supported frameworks to enable developers with flexibility

Supported Frameworks and Formats ➔ https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Introduction.html#SupportedFW
Configure the Model Optimizer for your Framework ➔ https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_Config_Model_Optimizer.html
Model Optimization

Breadth of supported frameworks to enable developers with flexibility

**Model Optimizer** loads a model into memory, reads it, builds the internal representation of the model, optimizes it, and produces the **Intermediate Representation**.

Optimization techniques available are:

- Linear operation fusing
- Stride optimizations
- Group convolutions fusing

Note: Except for ONNX (.onnx model formats), all models have to be converted to an IR format to use as input to the Inference Engine.
Model Optimizer: Linear Operation Fusing

- **Example**
  1. Remove Batch normalization stage.
  2. Recalculate the weights to ‘include’ the operation.
  3. Merge Convolution and ReLU into one optimized kernel.

---

![Diagram of Model Optimizer: Linear Operation Fusing](image)
Optimal Model Performance Using the Inference Engine

Core Inference Engine Libraries
- Create Inference Engine Core object to work with devices
- Read the network
- Manipulate network information
- Execute and pass inputs and outputs

Device-specific Plugin Libraries
- For each supported target device, Inference Engine provides a plugin — a DLL/shared library that contains complete implementation for inference on this device.

GPU = Intel CPU with integrated graphics/Intel® Processor Graphics/GEN
GNA = Gaussian mixture model and Neural Network Accelerator
Common Workflow for Using the Inference Engine API

1. Create Inference Engine Core object
   ```python
   ie = IECore()
   ```

2. Read the Intermediate Representation
   ```python
   net = ie.read_network(model=model_xml, weights=model_bin)
   ```

3. Prepare inputs and outputs format
   ```python
   input_blob = next(iter(net.inputs))
   output_blob = next(iter(net.outputs))
   ```

4. Load Network to device & Create infer request
   ```python
   exec_net = ie.load_network(network=net, device_name=device, num_requests=request_number)
   ```

5. Prepare input frame
   ```python
   n, c, h, w = net.inputs[input_blob].shape
   in_frame = cv2.resize(image, (w, h))
   in_frame = in_frame.transpose((2, 0, 1))
   in_frame = in_frame.reshape((n, c, h, w))
   ```

6. Run Inference
   ```python
   res = exec_net.infer(inputs={input_blob: in_frame})
   ```

7. Process the results
   ```python
   ```

8. Inference loop
   ```python
   ```

---

http://docs.openvinotoolkit.org/latest/docs_IE_DG_Integrate_with_customer_application_new_API.html
Inference Engine

**Synchronous vs Asynchronous Execution**

- In IE API model can be executed by **Infer Request** which can be:
  - **Synchronous** - blocks until inference is completed.
    - `exec_net.infer(inputs = {input_blob: in_frame})`
  - **Asynchronous** – checks the execution status with the wait or specify a completion callback *(recommended way).*
    - `exec_net.start_async(request_id = id, inputs={input_blob: in_frame})`
    - If `exec_net.requests[id].wait() != 0`
      - do something
Inference Engine

Throughput Mode for CPU, iGPU and VPU

- **Latency** – inference time of 1 frame (ms).
- **Throughput** – overall amount of frames inferred per 1 second (FPS)
- "Throughput" mode allows the Inference Engine to efficiently run multiple infer requests simultaneously, greatly improving the overall throughput.
- Device resources are divided into execution “streams” – parts which runs infer requests in parallel

**Device**

![Diagram showing execution streams and infer requests](image)

**CPU Example:**
```python
ie = IECore()
ie.GetConfig(CPU, KEY_CPU_THROUGHPUT_STREAMS)
```
Inference Engine

Heterogeneous Support

- You can execute different layers on different HW units
- Offload unsupported layers on fallback devices:
  - Default affinity policy
  - Setting affinity manually (CNNLayer::affinity)
- All device combinations are supported (CPU, GPU, FPGA, MYRIAD, HDDL)
- Samples/demos usage “-d HETERO:FPGA,CPU”

InferenceEngine::Core core;
auto executable_network =
core.LoadNetwork(reader.getNetwork(),
"HETERO:FPGA,CPU");
Inference Engine

Multi-device Support
Automatic load-balancing between devices (inference requests level) for full system utilization

- Any combinations of the following devices are supported (CPU, iGPU, VPU, HDDL)
- As easy as “-d MULTI:CPU,GPU” for cmd-line option of your favorite sample/demo
- C++ example (Python is similar)

```c
Core ie;
ExecutableNetwork exec = ie.LoadNetwork(network,{{"DEVICE_PRIORITIES", "CPU,GPU"}}, "MULTI")
```
Post-Training Optimization Tool

Conversion technique that reduces model size into low-precision without re-training

Reduces model size **while also improving latency, with little degradation** in model accuracy and without model re-training.

Different optimization approaches are supported: quantization algorithms, etc.
Deep Learning Workbench
Web-based UI extension tool for model analyses and graphical measurements

- **Visualizes performance data for** topologies and layers to aid in model analysis

- **Automates analysis** for optimal performance configuration (streams, batches, latency)

- **Experiment with INT8 or Winograd calibration** for optimal tuning using the Post Training Optimization Tool

- Provide **accuracy information** through accuracy checker

- **Direct access to models** from public set of Open Model Zoo

- Enables **remote profiling**, allowing the collection of performance data from multiple different machines without any additional set-up.
Pre-Trained Models and Public Models
Open-sourced repository of pre-trained models and support for public models

100+ Pre-trained Models
Common AI tasks
Object Detection
Object Recognition
Reidentification
Semantic Segmentation
Instance Segmentation
Human Pose Estimation
Image Processing
Text Detection
Text Recognition
Text Spotting
Action Recognition
Image Retrieval
Compressed Models
Question Answering

100+ Public Models
Pre-optimized external models
Classification
Segmentation
Object Detection
Human Pose Estimation
Monocular Depth Estimation
Image Inpainting
Style Transfer
Action Recognition
Colorization

Use free Pre-trained Models to speed up development and deployment

Take advantage of the Model Downloader and other automation tools to quickly get started

Iterate with the Accuracy Checker to validate the accuracy of your models
Intel® DevCloud for the Edge
Accelerate Test Cycles with the Intel® DevCloud for the Edge

A Development Sandbox for Developers, Researchers, and Startups to Test AI and Vision Workloads Remotely before Deployment.

With the Intel® DevCloud for the Edge users can:
• Prototype on the latest hardware and software to future proof the solution
• Benchmark the customized AI application
• Run AI applications from anywhere in the world
• Reduce development time and cost

[New] DL Workbench + Intel® DevCloud for the Edge

Developers can now graphically analyze models using the DL Workbench on Intel® DevCloud for the Edge (instead of local machine only) to compare, visualize and fine-tune a solution against multiple remote hardware configurations

For more information visit https://devcloud.intel.com/edge/
Accelerate Time to Production with Intel® DevCloud for the Edge

See immediate AI Model performance across Intel’s vast array of Edge Solutions

- **Instant, Global Access**
  Run AI applications from anywhere in the world

- **Prototype on the Latest Hardware and Software**
  Develop knowing you’re using the latest Intel technology

- **Benchmark your Customized AI Application**
  Immediate feedback - frames per second, performance

- **Reduce Development Time and Cost**
  Quickly find the right compute for your edge solution

[Learn more]

[Sign up now for access]
Demo

Pneumonia Classification with Class Activation Maps

https://devcloud.intel.com/edge/advanced/sample_applications/

-->Development Environment: OpenVINO 2021.3 Jupyter Notebook

Pneumonia Detection

HEALTHCARE

This example showcases a healthcare application by classifying the probability of pneumonia in X-ray images. The application uses the inference engine in the Intel® Distribution of OpenVINO™ toolkit and applies a pretrained neural network using an open source dataset. The inference results are stored in an output file.

View Code in Jupyter Notebook
Demo

Benchmark Tool

https://devcloud.intel.com/edge/advanced/advanced_tutorials/

-->Development Environment: OpenVINO 2021.3 Jupyter Notebook

Benchmark App
This tutorial demonstrates how to use the benchmark app to estimate inference performance of your deep learning model on various devices.

Try it out in Jupyter Notebook
Ready to get started?

Download directly from Intel for free

Intel® Distribution of OpenVINO™ toolkit
(Recommended)

Also available from

Intel's Edge Software Hub | Intel® DevCloud for the Edge | PIP | DockerHub | Dockerfile | Anaconda Cloud | YUM | APT

Build from source

GitHub | Gitee (for China)

Choose & Download
Questions?
## Choose between Distributions

<table>
<thead>
<tr>
<th>Tool/Component</th>
<th>Intel® Distribution of OpenVINO™ toolkit</th>
<th>OpenVINO™ toolkit (open source)</th>
<th>Open Source Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installer (including necessary drivers)</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/model_optimizer">GitHub</a></td>
</tr>
<tr>
<td>Model Optimizer</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Inference Engine - Core</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Intel CPU plug-in</td>
<td>✓ Intel® Math Kernel Library (Intel® MKL) only&lt;sup&gt;1&lt;/sup&gt;</td>
<td>✓ BLAS, Intel® MKL&lt;sup&gt;1&lt;/sup&gt;, jit (Intel MKL)</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Intel GPU (Intel® Processor Graphics) plug-in</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Heterogeneous plug-in</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Intel GNA plug-in</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Intel® FPGA plug-in</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Intel® Neural Compute Stick (1 &amp; 2) VPU plug-in</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Intel® Vision Accelerator based on Movidius plug-in</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Multi-device &amp; hetero plug-ins</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/openvino/tree/master/inference_engine">GitHub</a></td>
</tr>
<tr>
<td>Public and Pretrained Models - incl. Open Model Zoo (IR models that run in IE + open sources models)</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/open_model_zoo">GitHub</a></td>
</tr>
<tr>
<td>Samples (APIs)</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/open_model_zoo">GitHub</a></td>
</tr>
<tr>
<td>Demos</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/openvinotoolkit/open_model_zoo">GitHub</a></td>
</tr>
<tr>
<td><strong>Traditional Computer Vision</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenCV*</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/opencv/opencv">GitHub</a></td>
</tr>
<tr>
<td>Intel® Media SDK</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/Intel-Media-SDK/MediaSDK">GitHub</a></td>
</tr>
<tr>
<td>OpenCL™ Drivers &amp; Runtimes</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/Intel/compute-runtime">GitHub</a></td>
</tr>
<tr>
<td>FPGA Runtime Environment, Deep Learning Acceleration &amp; Bitstreams (Linux* only)</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/opencv/opencv">GitHub</a></td>
</tr>
</tbody>
</table>
# System Requirements

## Intel® Platforms

### CPU
- 6th-10th generation Intel® Core™ and Xeon® processors
- 1st and 2nd generation Intel® Xeon® Scalable processors
- Intel® Pentium® processor N4200/5, N3350/5, N3450/5 with Intel® HD Graphics

### Iris® Pro & Intel® HD Graphics
- 6th-10th generation Intel® Core™ processor with Intel® Iris® Pro graphics & Intel® HD Graphics
- Intel® Xeon® processor with Intel® Iris® Pro Graphics & Intel® HD Graphics (excluding E5 product family, which does not have graphics)

### FPGA
- Intel® Arria® FPGA 10 GX development kit
- Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA operating systems
- OpenCV & OpenVX* functions must be run against the CPU or Intel® Processor Graphics (GPU)

### VPU: Intel® Vision Accelerator Design Products
- Intel® Vision Accelerator Design with Intel® Arria10 FPGA
- Intel® Vision Accelerator Design with Intel® Movidius™ VPUs

### Development Platforms
- 6th-10th generation Intel® Core™ and Intel® Xeon® processors
- 1st and 2nd generation Intel® Xeon® Scalable processors

## Compatible Operating Systems

- Ubuntu* 18.04.3 LTS (64 bit)
- Microsoft Windows* 10 (64 bit)
- CentOS® 7.4 (64 bit)
- macOS* 10.13 & 10.14 (64 bit)
- Yocto Project* Poky Jethro v2.0.3 (64 bit)
- Ubuntu 18.04.3 LTS (64 bit)
- Windows 10 (64 bit)
- CentOS 7.4 (64 bit)

## Target Solution Platforms

### Intel® Platforms

#### CPU
- 6th-10th generation Intel® Core™ and Xeon® processors
- 1st and 2nd generation Intel® Xeon® Scalable processors
- Intel® Pentium® processor N4200/5, N3350/5, N3450/5 with Intel® HD Graphics

#### Iris® Pro & Intel® HD Graphics
- 6th-10th generation Intel® Core™ processor with Intel® Iris® Pro graphics & Intel® HD Graphics
- Intel® Xeon® processor with Intel® Iris® Pro Graphics & Intel® HD Graphics (excluding E5 product family, which does not have graphics)

#### FPGA
- Intel® Arria® FPGA 10 GX development kit
- Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA operating systems
- OpenCV & OpenVX* functions must be run against the CPU or Intel® Processor Graphics (GPU)

#### VPU: Intel® Vision Accelerator Design Products
- Intel® Vision Accelerator Design with Intel® Arria10 FPGA
- Intel® Vision Accelerator Design with Intel® Movidius™ VPUs

### FPGA
- Ubuntu 18.04.2 LTS (64 bit)
- CentOS 7.4 (64 bit)
- Ubuntu 18.04.3 LTS (64 bit)
- CentOS 7.4 (64 bit)
- Windows 10 (64 bit)
- macOS (64 bit)
- Raspbian (target only)
- Ubuntu 18.04.2 LTS (64 bit)
- Ubuntu 8.04.3 LTS (64 bit)
- Windows 10 (64 bit)

### Additional Software Requirements

- Linux® build environment required components
  - OpenCV 3.4 or higher
  - CMake® 2.8 or higher
- Microsoft Windows® build environment required components
  - Intel® HD Graphics Driver (latest version)
  - Intel® C++ Compiler 2017 Update 4
  - Microsoft Visual Studio® 2015

- Additional Software Requirements
  - GNU Compiler Collection (GCC) 3.4 or higher
  - Python® 3.4 or higher

## External Dependencies/Additional Software

- View Product Site, detailed System Requirements

---

Intel® Distribution of OpenVINO™ toolkit / Product Overview

2020 © Copyright, Intel Corporation
Commonly Asked Questions

Can I use the Intel® Distribution of OpenVINO™ toolkit for commercial usage? Yes, the Intel® Distribution of OpenVINO™ toolkit is licensed under Intel’s End User License Agreements and the open-sourced OpenVINO™ toolkit is licensed under Apache License 2.0. For information, review the licensing directory inside the package.

Is the Intel® Distribution of OpenVINO™ toolkit subject to export control? Yes, the ECCN is EAR99.

How often does the software get updated? Standard releases are updated 3-4 times a year, while LTS releases are updated once a year.

What is the difference between Standard and LTS releases? Standard Releases are recommended for new users and users currently prototyping. It offers new features, tools and support to stay current with deep learning advancements. LTS Releases are recommended for experienced users that are ready to take their application into production and who do not require new features and capabilities for their application.

For technical questions, visit the Model Optimizer FAQ and Performance Benchmarks FAQ. If you don’t find an answer, please visit the following community and support links.

Get Help
- Ask on the Community Forum
- Contact Intel Support
- File an Issue on GitHub*
- Get Answers on StackOverflow*

Get Involved
- Contribute to the Code Base
- Contribute to Documentation

Stay Informed
- Join the Mailing List
- Read the Documentation
- Read the Knowledge Base
- Read the Blog