Optimization Techniques with OpenVINO™ to Enhance Performance on Your Existing Hardware

Nico Galoppo, Principal Engineer
Ryan Loney, Technical Product Manager
The Challenge
How do I Deploy My Great Neural Network?
GPU?
GPU?  Cloud?
GPU? 

Cloud?
Maybe just CPU?
“But running on CPU is slow…”
Almost Every Deep Learning Engineer

Is it?
Developer Journey

1. BUILD
2. OPTIMIZE
3. DEPLOY

© 2022 Intel
Model Optimizer

Get Your Model → Run Model Optimizer → IR

For workloads and configurations visit www.intel.com/PerformanceIndex. Results may vary.

© 2022 Intel
Performance

Throughput (FPS) - FP32

- **Intel® Core™ i7-1185G7 CPU+iGPU**
- **Intel® Core™ i7-1185G7 CPU-only**
- **Intel® Core™ i7-1185G7 GPU-only**
- **Intel® Core™ i7-8700T**
- **Intel® Core™ i5-8500**
- **Intel® Core™ i3-8100**

For workloads and configurations please scan QR code. Results may vary.

```
$ benchmark_app -m model_path -d device
```
For workloads and configurations visit www.intel.com/PerformanceIndex. Results may vary.

```sh
$ mo --input_model model.onnx --data_type FP32
```
Neural Network (any format)
Intermediate Representation (IR)
Post-Training Optimization Tool (POT)
Neural Network Compression Framework (NNCF)
Neural Network Compression Framework (NNCF)

- Quantization-Aware Training
- Mixed-Precision Quantization
- Filter Pruning
Neural Network Compression Framework (NNCF)

- Quantization-Aware Training
- Mixed-Precision Quantization
- Filter Pruning
from openvino.runtime import Core

img = load_img()

core = Core()
model = core.read_model(model="model.xml", weights="model.bin")
compiled_model = core.compile_model(model=model, device_name="CPU")

output_layer = compiled_model.outputs[0]

result = compiled_model([img])[output_layer]
supported_devices = compiled_model = core.compile_model(model=model, device_name="CPU")
Supported Devices

```python
compiled_model = core.compile_model(model=model, device_name="CPU")
```
```
compiled_model = core.compile_model(model=model, device_name="CPU")
```
compiled_model = core.compile_model(model=model, device_name="CPU")
compiled_model = core.compile_model(model=model, device_name="CPU")
```python
compiled_model = core.compile_model(model=model, device_name="CPU")
```
compiled_model = core.compile_model(model=model, device_name="CPU")
AUTO Device

CPU Workloads
- Device Selection
- Compile Network for CPU
- Compile / Load Network for GPU
- Inference on CPU
- Inference on CPU Stops on CPU

GPU Workloads
- Application Inference on Auto
- Compile / Load Network for GPU
- Inference on GPU

Application Logic
Loading NN
Inference

Time (ms)
116.7
5,800

© 2022 Intel
Input Data with Variable Shape?

“What is the weather going to be like today?”

OpenVINO running Inference time: 19.8ms
How to handle variable input shapes?

Padding

Multiple Precompiled Models

Inefficient and Cumbersome!
Native Dynamic Shapes Support in OpenVINO™

Image → Text Detection (db_mv3_slim, 1.4M) → Detection Boxes Rectify (dir_cls_mv3_slim, 0.5M) → Text Recognition (crnn_mv3_slim, 1.6M) → Output

**Dynamic in input image size**
Portable Performance Hints

- Set-and-forget optimization knob – choose **latency** or **throughput**.
- Completely portable between the devices

```
compiled_model = core.compile_model(model, "GPU", {"PERFORMANCE_HINT": "THROUGHPUT"})
```

- Throughput hint drives device-specific optimizations
- Even works for AUTO!

```
compiled_model = core.compile_model(model, "AUTO", {"PERFORMANCE_HINT": "THROUGHPUT"})
```
Automatic Batching

• How do you choose a good batch size?
• Let the OpenVINO™ runtime decide for you!
• No need to batch requests, runtime will do that too!
• Batching can improve throughput with select devices and models.

```python
# when the batch size is automatically selected by the implementation
# it is important to query/create and run the sufficient requests
compiled_model = core.compile_model(model, "GPU", {"PERFORMANCE_HINT": "THROUGHPUT"})
num_requests = compiled_model.get_property("OPTIMAL_NUMBER_OF_INFER_REQUESTS")

# leaving intact other configurations options that the device selects for the 'throughput' hint
config = {"PERFORMANCE_HINT": "THROUGHPUT",
         "ALLOW_AUTO_BATCHING": "NO"}
compiled_model = core.compile_model(model, "GPU", config)
```
Open Model Zoo

300+
Pre-Trained + Optimized Models

© 2022 Intel
Open Model Zoo

300+

Pre-Trained + Optimized Models
300+

Pre-Trained + Optimized Models

Inference time: 14.4ms
300+
Pre-Trained + Optimized Models
Open Model Zoo

300+
Pre-Trained + Optimized Models

© 2022 Intel
Open Model Zoo

300+
Pre-Trained + Optimized Models

Inference time: 24.2ms (41.3 FPS)
Open Model Zoo

300+
Pre-Trained + Optimized Models
Jupyter Notebooks

Run Live Object Detection

Run using a webcam as the video input. By default, the primary webcam is set with `source=0`. If you have multiple webcams, each one will be assigned a consecutive number starting at 0. Set `flip=True` when using a front-facing camera. Some web browsers, especially Mozilla Firefox, may cause flickering. If you experience flickering, set `use_popup=True`. Note popup mode may not work if you run this notebook on a remote computer.

```
In [0]: run_object_detection(source=0, flip=True, use_popup=False)
```

Cannot open camera 0
Source ended

Run Object Detection on a Video File

If you don’t have a webcam, you can still run this demo with a video file. Any format supported by OpenCV will work (see: https://docs.opencv.org/3.4.1/d4d43/tutorial_py_video_display.html).

```
In [*]: video_file = './2011-vision-monodepth/data/Coco Walking in Berkeley.mp4'
run_object_detection(source=video_file, flip=False, use_popup=False)
```

Inference time: 9.7ms (102.6 FPS)
Try it yourself!

Learn more and download at openvino.ai

Complete the Intel® Edge AI Certification!

• Supercharge your career
• Approx 20 hours of video + quizzes + coding
Thursday, May 19

**Intel AI Developer Expo—Let's Build Something Wonderful Together**

Session: 3:00 – 5:30 pm  
Reception: 5:30 – 7:30 pm

**Location:** Room 209/210

Afternoon snack and post-session Reception will be provided
Thank you!
### Platform Configurations for Performance Benchmarks

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Intel® Core™ i7-1185G7</th>
<th>Intel® Core™ i3-8100</th>
<th>Intel® Core™ i5-8500</th>
<th>Intel® Core™ i7-8700T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motherboard</td>
<td>Intel Corporation internal/Reference Validation Platform</td>
<td>GIGABYTE® Z390 UD</td>
<td>ASUS® PRIME Z370-A</td>
<td>GIGABYTE® Z370M DS3H-CF</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel® Core™ i7-1185G7 @ 3.00GHz</td>
<td>Intel® Core™ i3-8100 CPU @ 3.60GHz</td>
<td>Intel® Core™ i5-8500 CPU @ 3.60GHz</td>
<td>Intel® Core™ i7-8700T CPU @ 2.40GHz</td>
</tr>
<tr>
<td>Hyper Threading</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>Turbo Setting</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>Memory</td>
<td>2 x 8 GB DDR4 3200MHz</td>
<td>4 x 8 GB DDR4 2400MHz</td>
<td>2 x 16 GB DDR4 2666MHz</td>
<td>4 x 16 GB DDR4 2400MHz</td>
</tr>
<tr>
<td>Operating System</td>
<td>Ubuntu* 18.04 LTS</td>
<td>Ubuntu* 18.04 LTS</td>
<td>Ubuntu* 18.04 LTS</td>
<td>Ubuntu* 18.04 LTS</td>
</tr>
<tr>
<td>Kernel Version</td>
<td>5.8.0-05-generic</td>
<td>5.3.0-24-generic</td>
<td>5.3.0-24-generic</td>
<td>5.3.0-24-generic</td>
</tr>
<tr>
<td>BIOS Vendor</td>
<td>Intel Corporation</td>
<td>American Megatrends Inc.*</td>
<td>American Megatrends Inc.*</td>
<td>American Megatrends Inc.*</td>
</tr>
<tr>
<td>BIOS Version</td>
<td>TGLSFW1.R00.3425. A00.2010162309</td>
<td>Select optimized default settings, set OS type to “other”, save and exit</td>
<td>Select optimized default settings, save and exit</td>
<td>Select optimized default settings, set OS type to “other”, save and exit</td>
</tr>
<tr>
<td>BIOS Settings</td>
<td>Default Settings</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Batch size</td>
<td>1</td>
<td>FP32</td>
<td>FP32</td>
<td>FP32</td>
</tr>
<tr>
<td>Precision</td>
<td>FP32</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Number of concurrent inference requests</td>
<td>1</td>
<td>1</td>
<td>18-Jun-21</td>
<td>18-Jun-21</td>
</tr>
<tr>
<td>Test Date</td>
<td>18-Jun-21</td>
<td>18-Jun-21</td>
<td>18-Jun-21</td>
<td>18-Jun-21</td>
</tr>
<tr>
<td>Rated maximum TDP/socket in Watt</td>
<td>28</td>
<td>65</td>
<td>65</td>
<td>65</td>
</tr>
</tbody>
</table>

© 2022 Intel
Notices and Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.

No product or component can be absolutely secure.

Intel technologies may require enabled hardware, software or service activation. Your costs and results may vary.

Intel is committed to respecting human rights and avoiding complicity in human rights abuses. See Intel's Global Human Rights Principles. Intel's products and software are intended only to be used in applications that do not cause or contribute to a violation of an internationally recognized human right.

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