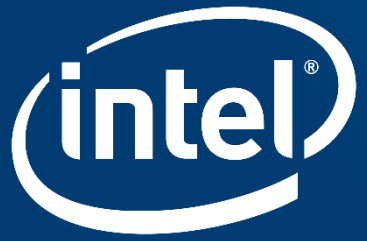


PROFILING DEEP LEARNING MODELS USING THE DEEP LEARNING WORKBENCH

LEGAL DISCLAIMER

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

This sample source code is released under the Intel Sample Source Code License Agreement.

Intel, the Intel logo, and OpenVINO are trademarks of Intel Corporation in the U.S. and/or other countries.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

*Other names and brands may be claimed as the property of others.

AGENDA

Chapter Outcome: Profile an Image Classification Model Using the DL Workbench

- Capabilities of the Deep Learning (DL) Workbench
- Installation instructions
- Profiling a trained deep learning model using the DL Workbench

WHAT IS THE DEEP LEARNING WORKBENCH?

- A web-based graphical simulation environment to visualize deep learning models on Intel® architecture (CPU, processor graphics, vision processing unit)
- Find the best deployment configurations for throughput versus latency
- Identify per layer execution time
- Compare model IR with run time graph
- Configure and measure accuracy of models
- Additionally, where possible, quantize a FP32 model to INT8 to fine-tune model performance (CPU only)

The DL Workbench is a **preview** feature in OpenVINO™ toolkit release R3 2019. Functionality is subject to change. Refer to DL Workbench documentation for up-to-date information

DL WORKBENCH—INSTALLATION

You can install the DL Workbench in one of the three ways:

- Install from the OpenVINO™ toolkit package (shown in this chapter)
  - Use the “run_openvino_workbench.sh” script available in /<path_to_installed_package>/deployment_tools/tools/workbench
  - Pass the OpenVINO installer tar.gz file as input to the script
- Install from Docker Hub*
  - Run DL Workbench by building a Docker* image locally (Linux* and Windows*)
- Install from Dockerfile
  - Build a Docker image that contains the OpenVINO toolkit package including the DL Workbench

MODEL PROFILING—PREREQUISITES

To start profiling a model, you will need:

A trained model

- Use one of the pretrained models from OpenVINO™ toolkit model zoo or provide a custom model

Dataset to run inference on

- The DL Workbench generates a dataset in ImageNet format for pretrained classification models
- You will need a sample custom dataset if profiling a custom model

Inference settings such as target device (CPU, GPU, VPU), number of runs, and number of batches

This chapter demonstrates DL Workbench capabilities with pretrained models. Steps are similar for custom models.

PROFILING A DEEP LEARNING MODEL USING THE DL WORKBENCH

1: LAUNCH THE DL WORKBENCH

Go to the Workbench Installation directory

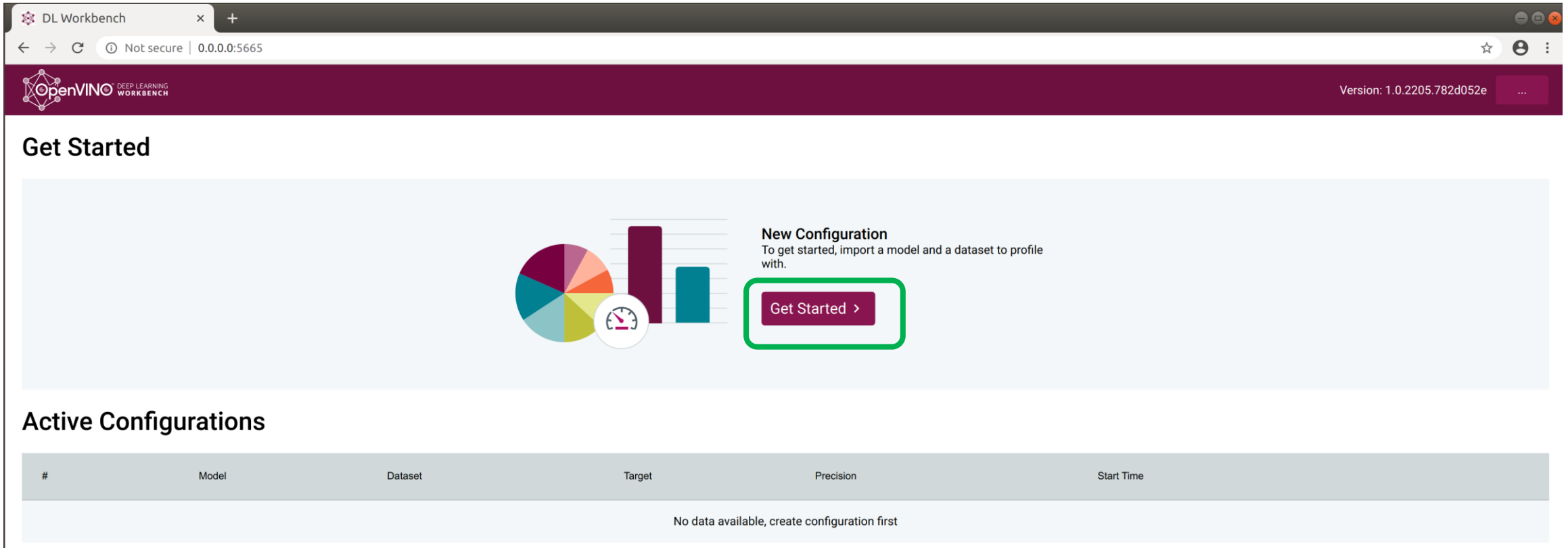
- /<path_to_installed_package>/deployment_tools/tools/workbench

Run the installer script to launch the web interface for DL Workbench

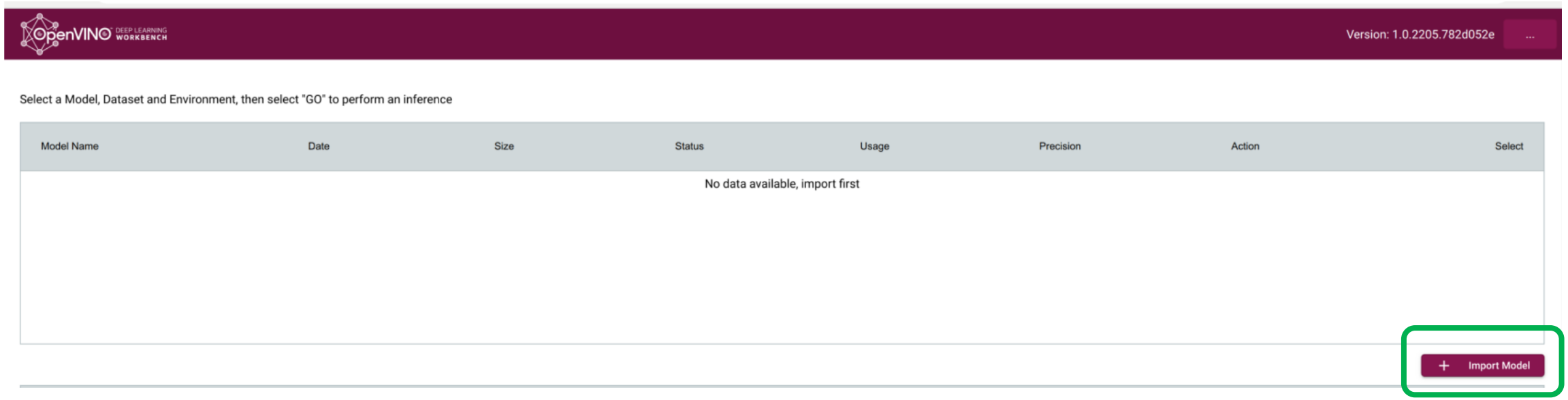
- `./run_openvino_workbench.sh -PACKAGE_PATH <path-to-archive>`

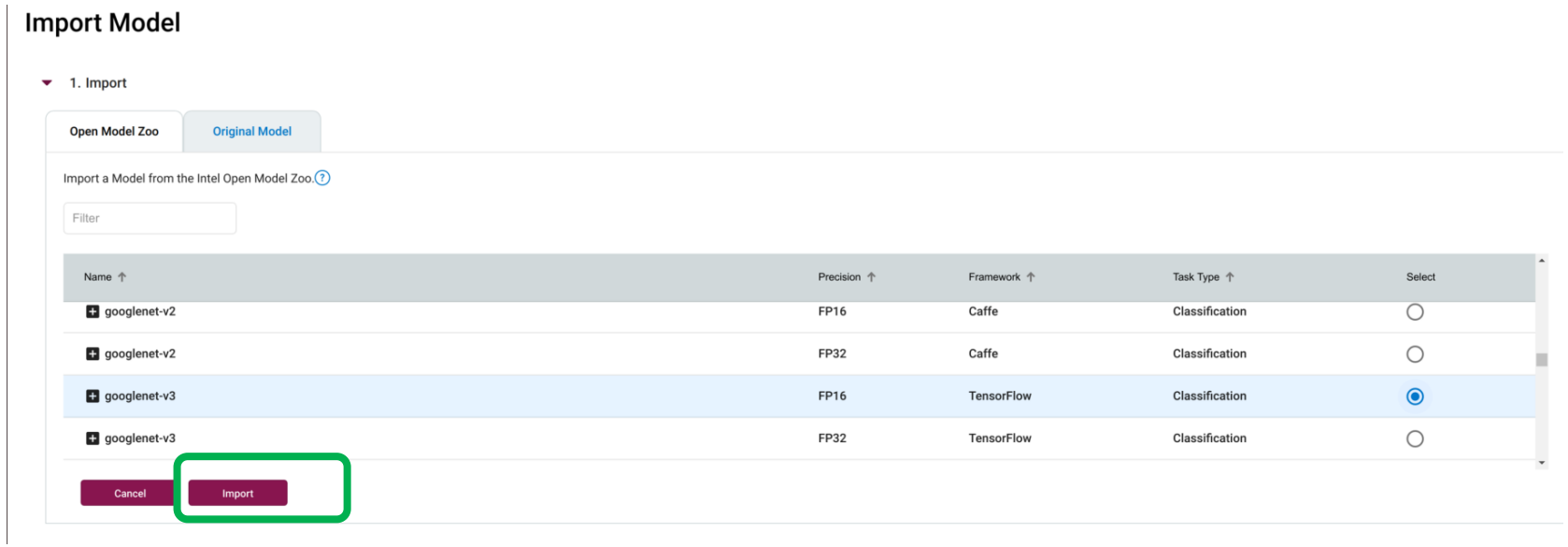
Once complete, the URL to launch the DL Workbench is provided by the script (the default URL is http://127.0.0.1:5665).

1: LAUNCH DL WORKBENCH



1: SELECT PRETRAINED MODEL

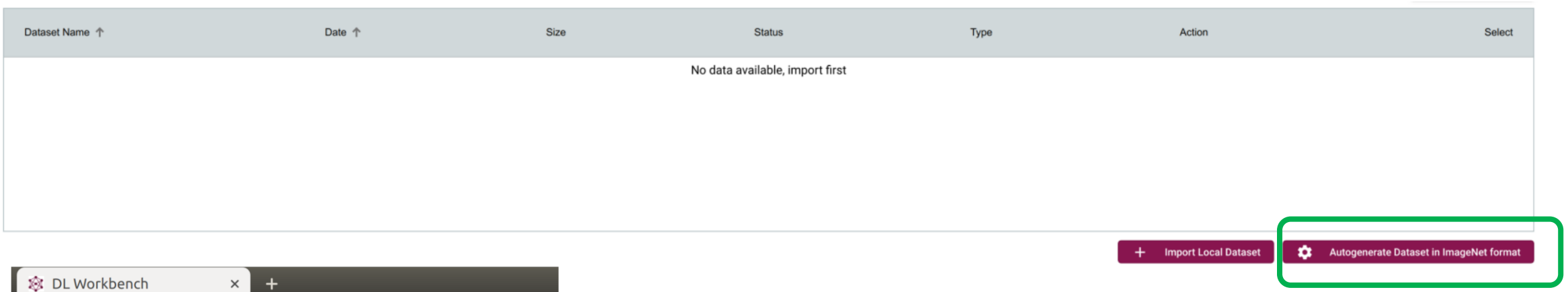


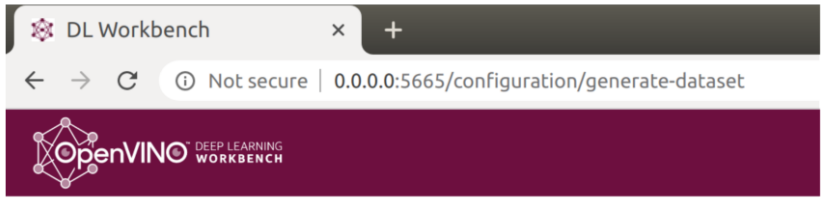


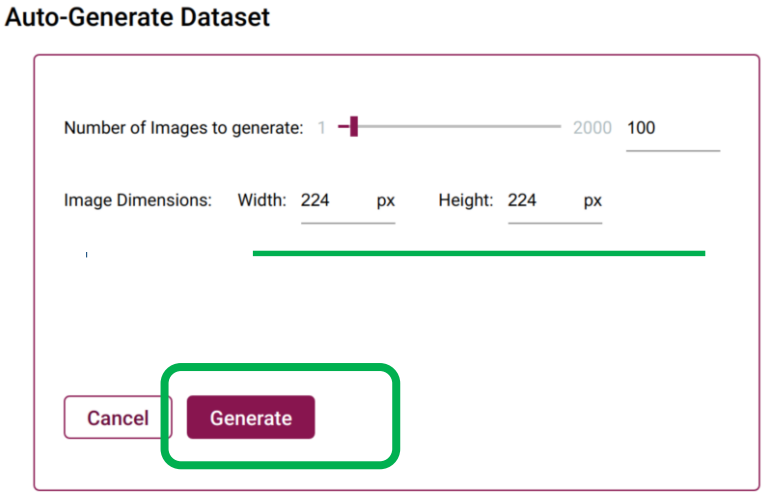
Imported frozen graph is run through the Model Optimizer (MO) to generate IR

Select FP16 precision for vision processing unit

1: SELECT OR AUTO-GENERATE DATASET

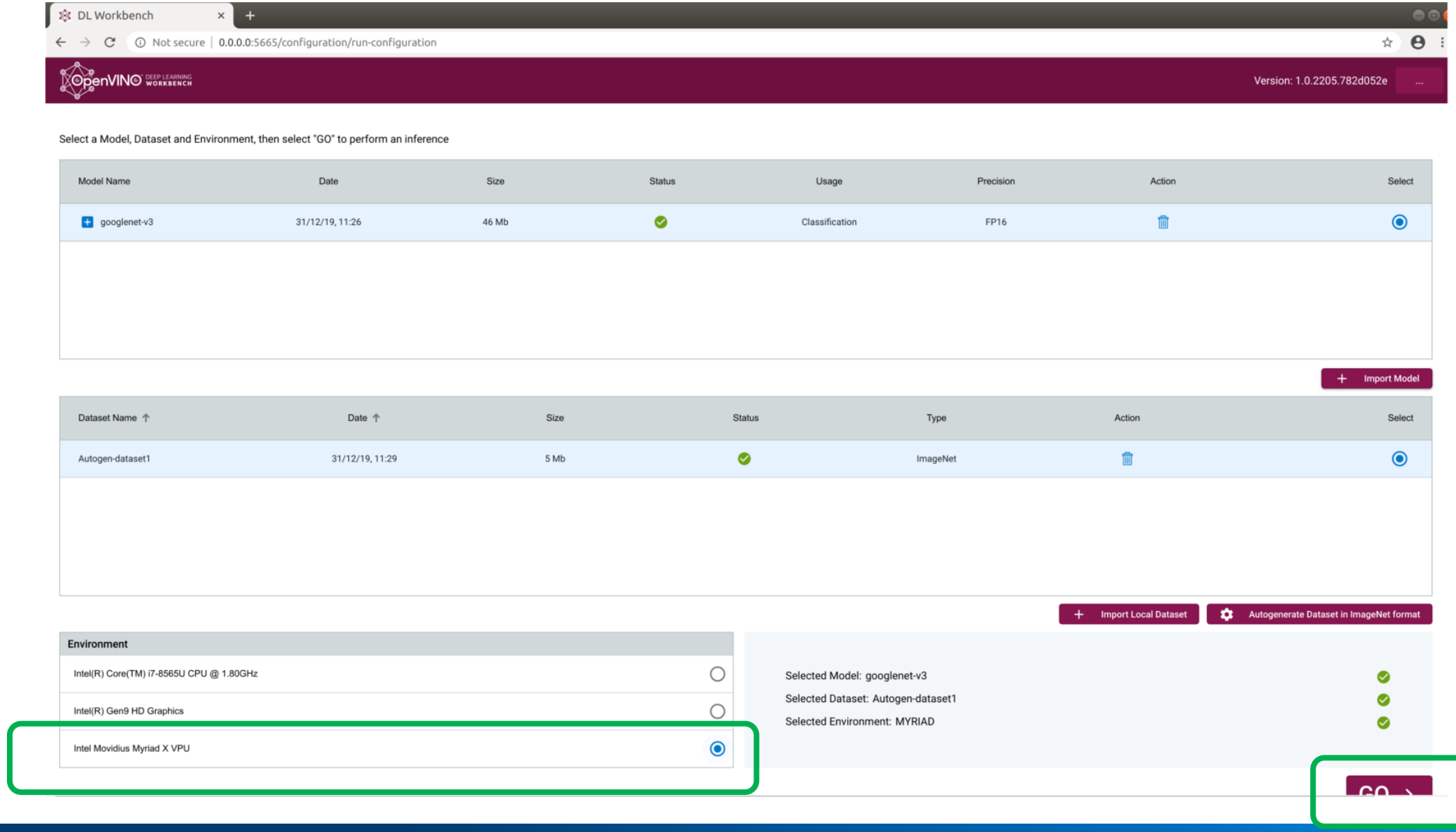






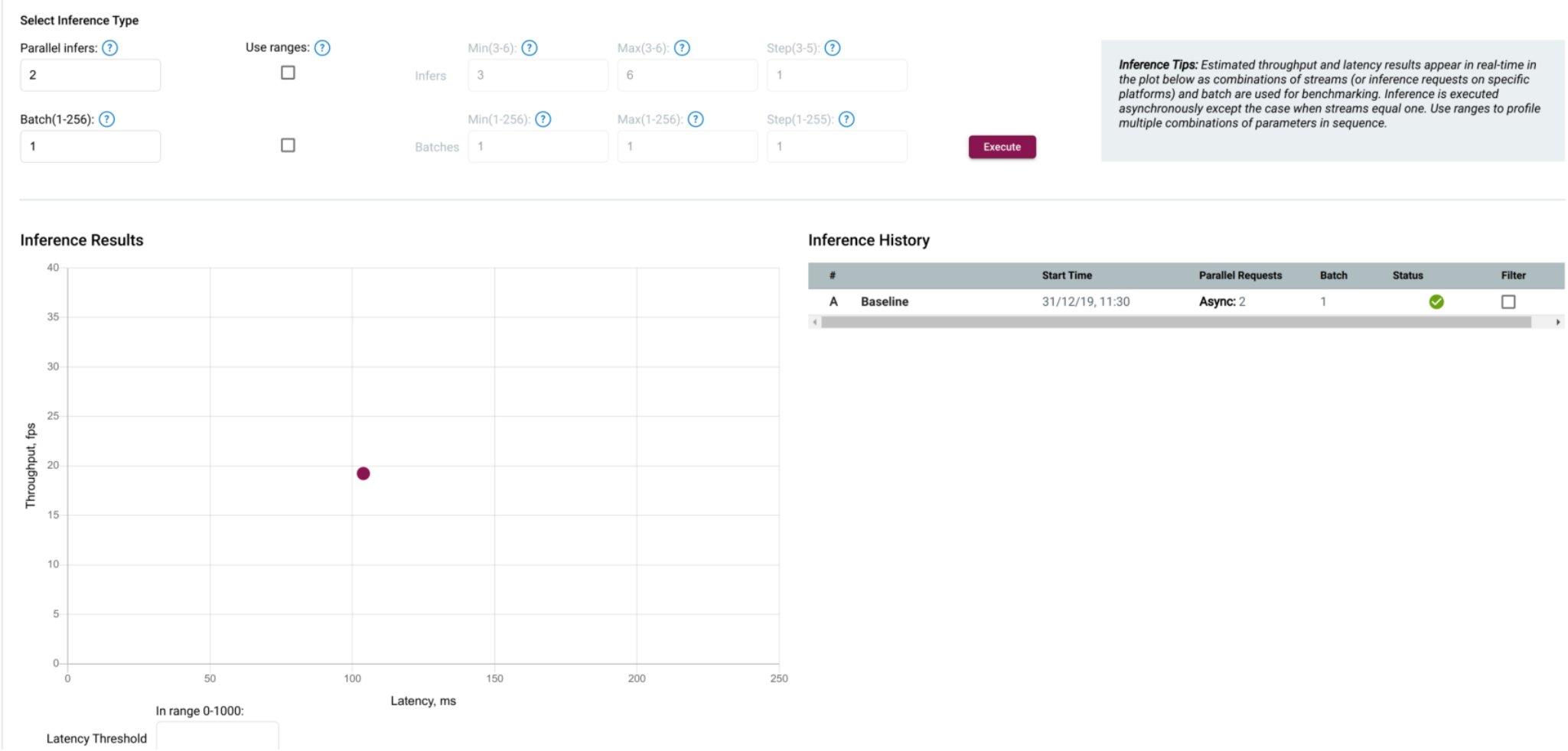
It is important to size the images as required by the topology selected. GoogLeNet requires images 224 x 224.

1: SELECT TARGET DEVICE



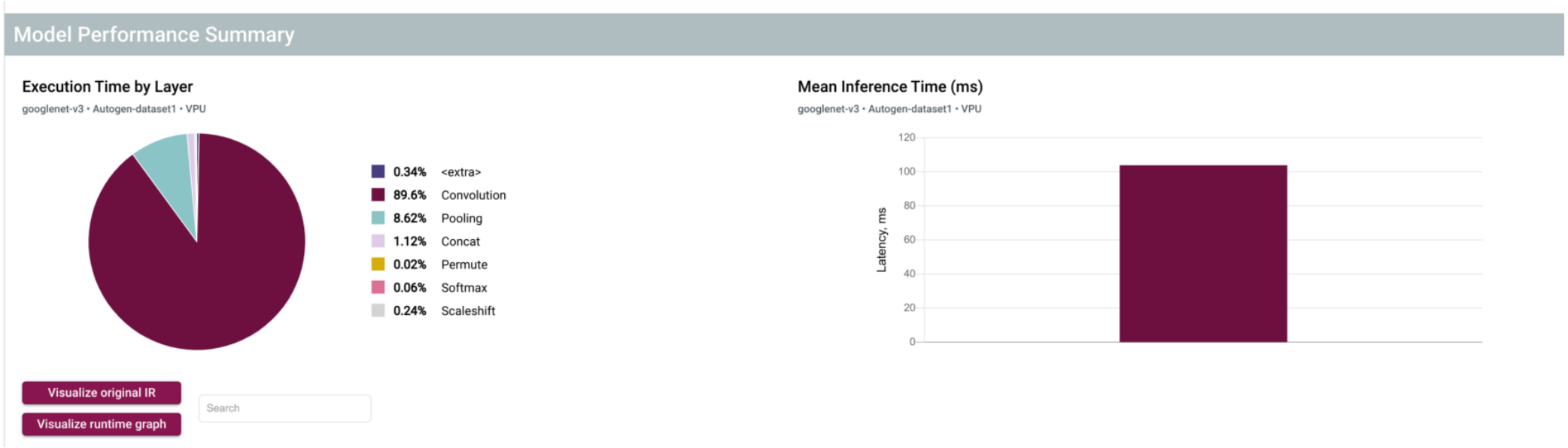
2: RUN SINGE INFERENCE

Perform a standard inference on the target device (in this case VPU) with the default settings



3: VISUALIZE THE TRAINED MODEL

Check the Model Performance Summary for statistics on the initial run of the model



Shows execution time per layer and mean inference time

4: CHECK MODEL EXECUTION TIME PER LAYER

Shows execution time for each layer in the topology

InceptionV3/InceptionV3/Mixed_5b/Branch_0/Conv2d_0a_1x1/convolution	Convolution	FP16	0.186
InceptionV3/InceptionV3/Mixed_5b/Branch_1/Conv2d_0a_1x1/convolution	Convolution	FP16	0.236
InceptionV3/InceptionV3/Mixed_5b/Branch_1/Conv2d_0b_5x5/convolution	Convolution	FP16	0.705
InceptionV3/InceptionV3/Mixed_5b/Branch_2/Conv2d_0a_1x1/convolution	Convolution	FP16	0.236
InceptionV3/InceptionV3/Mixed_5b/Branch_2/Conv2d_0b_3x3/convolution	Convolution	FP16	0.596
InceptionV3/InceptionV3/Mixed_5b/Branch_2/Conv2d_0c_3x3/convolution	Convolution	FP16	0.785

Check layer specific statistics

InceptionV3/InceptionV3/Mixed_5b/Branch_0/Conv2d_0a_1x1/convolution — Convolution — FP16 — 0.186

InceptionV3/InceptionV3/Mixed_5b/Branch_0/Conv2d_0a_1x1/convolution — Convolution

Positional data

Input 0 1, 192, 35, 35

Output 0 1, 64, 35, 35

Layers parameters

auto_pad same_upper
pads_begin 0,0
outputLayouts NCHW
dilations 1,1
pads_end 0,0
outputPrecisions FP16
group 1
strides 1,1
primitiveType MyriadXHwOp
kernel 1,1
execOrder 39
output 64
execTimeMcs 0.186693

5: COMPARE MODEL IR WITH RUNTIME GRAPH

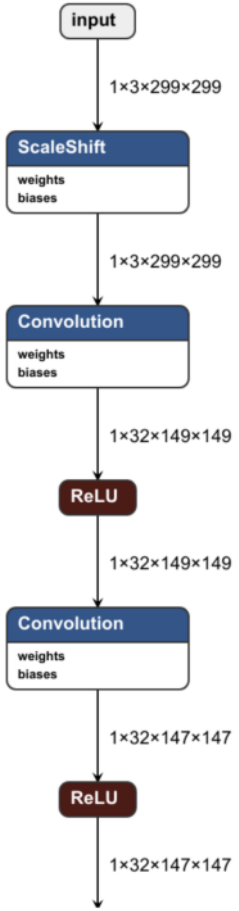
This gives a comparison of the model IR before it is run through the IE plugin and the graph when executed through the IE.

You will likely notice how some layers are different between the two. This happens because a model is changing during the inference process to achieve better performance.

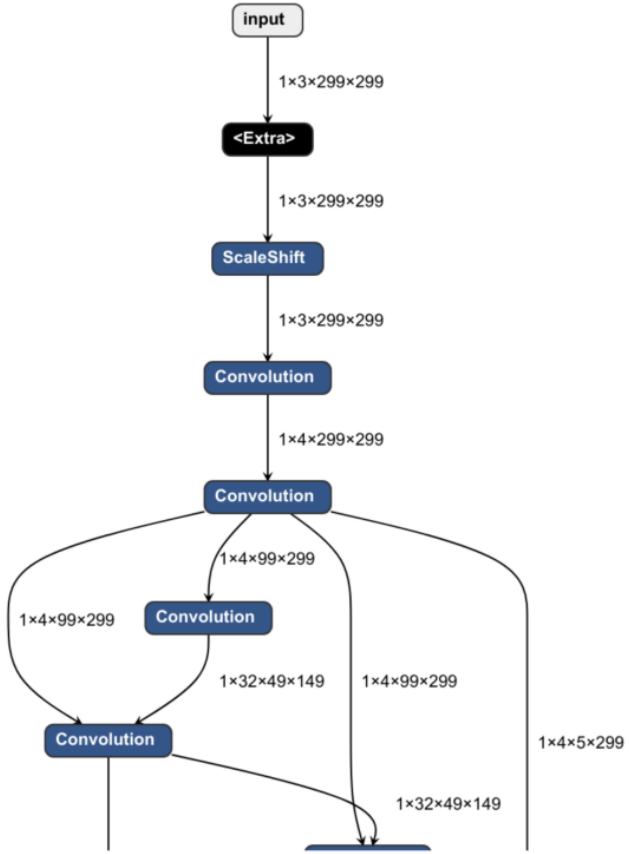
"The **Layer Name** table below shows each layer of a model. For those layers that differ between versions of a model, the *top-level* rows represent layers of a model during its runtime, while the *subrows* correspond to the layers of an original IR. *Gray areas* show combined properties of layers of versions. The layer parameter **primitiveType** shows a selected primitive type, and the **outputPrecisions** parameter corresponds to the precision of a model."

5: COMPARE MODEL IR WITH RUNTIME GRAPH

Original model IR before running through IE plugin



Runtime graph through IE



6: THROUGHPUT VERSUS LATENCY TUNING

The sweet spot for throughput and latency tradeoff can be achieved by tuning for batch sizes on different target devices and asynchronous inferences.

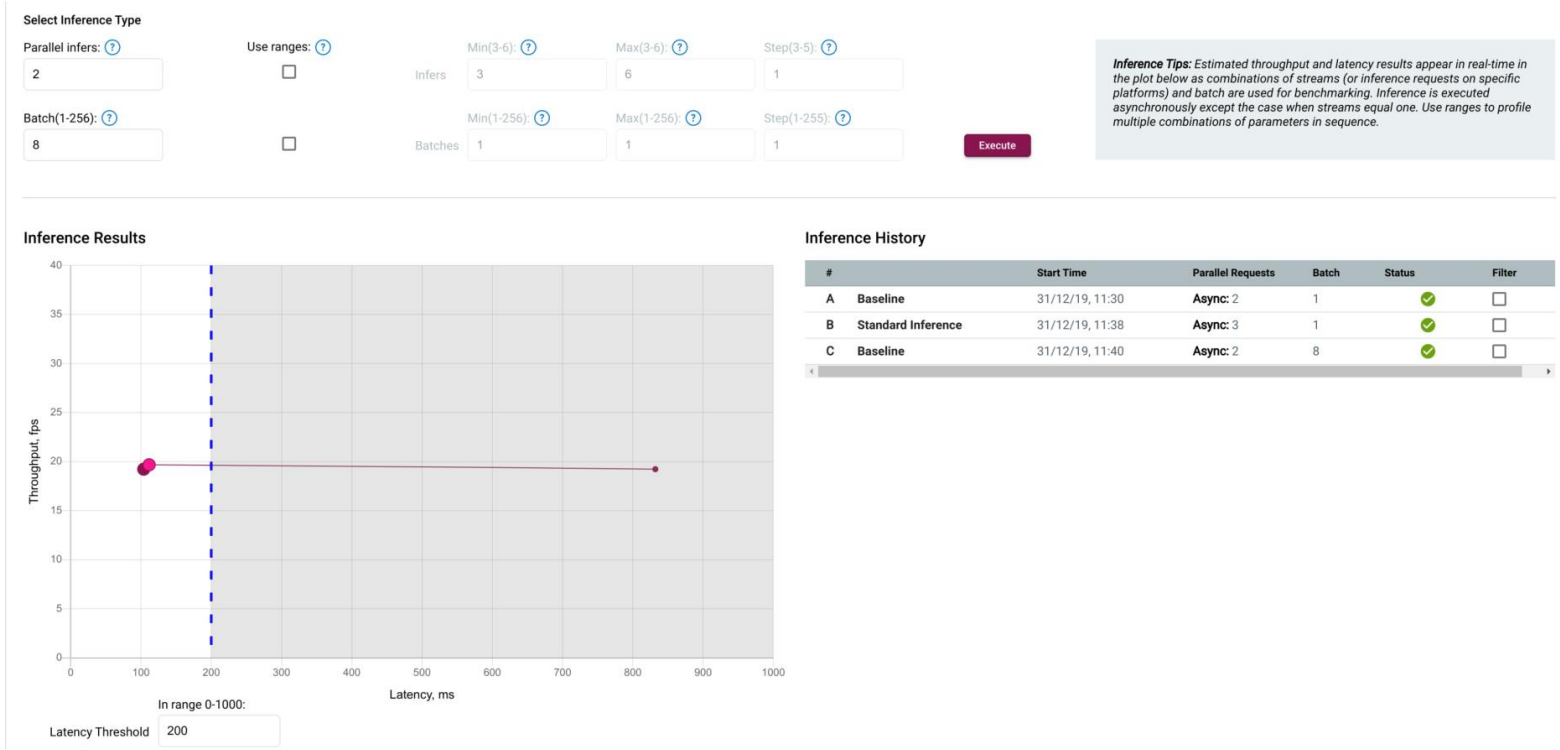
Now that we have executed a single inference on the model in Step 5, you can change the parallel inputs and batch size variables until you obtain a desired throughput with agreeable latency.

You can also set the latency threshold that you can tolerate at deployment on the target device.

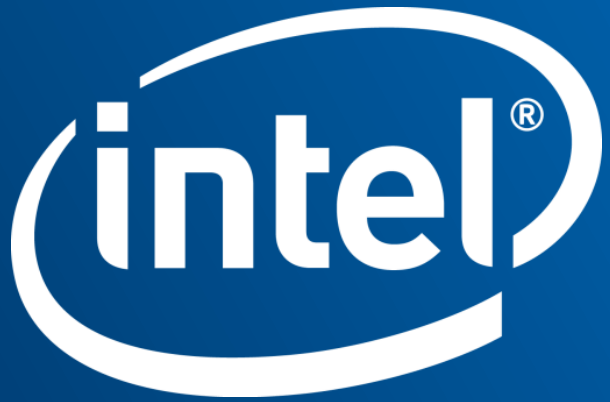
6: THROUGHPUT VERSUS LATENCY TUNING

The variation you get with changes to input variables depends on the model and target hardware.

Optimizations like INT8 and Winograd are available for the CPU.



Variable values chosen here are for demonstration purposes only.


intel