



IMPLEMENT CUSTOM LAYERS WITH INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

Priyanka Bagade

May 2019

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.



Legal Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

This document contains information on products, services, and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications, and roadmaps.

Any forecasts of goods and services needed for Intel's operations are provided for discussion purposes only. Intel will have no liability to make any purchase in connection with forecasts published in this document.

Arduino* 101 and the Arduino infinity logo are trademarks or registered trademarks of Arduino, LLC.

Altera, Arria, the Arria logo, Intel, the Intel logo, Intel Atom, Intel Core, Intel Nervana, Intel Xeon Phi, Movidius, Saffron, and Xeon are trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Copyright © 2018, Intel Corporation. All rights reserved.



Legal Notices and Disclaimers

This document contains information on products, services, and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications, and roadmaps. Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/performance.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described 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.

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon, Movidius, Saffron, and others are trademarks of Intel Corporation in the United States and other countries.

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

Copyright © 2018, Intel Corporation. All rights reserved.



Custom Layer Workflow

Model Optimizer

Custom Layer Attribute Extraction
(Python*)

Custom Layer Implementation (Python*)



Inference Engine

Custom Layer Implementation (CPU)

Language: C++

Final Product: Compiled binary file (.dll or .so)

Custom Layer Implementation (GPU)

Language: OpenCL™(C-based)

Final Product: .cl & .xml files

Custom Layer Implementation (FPGA)

Language: OpenCL™(C-based)

Final Product: bitstream(.aocx file)

Extension Generator Tool

- Python* based tool
(extgen.py)
 - Generates template/stub files
with core functions for
creating custom layers.
- ∴ Extension == Custom Layer == Kernel

Extension Generator Tool

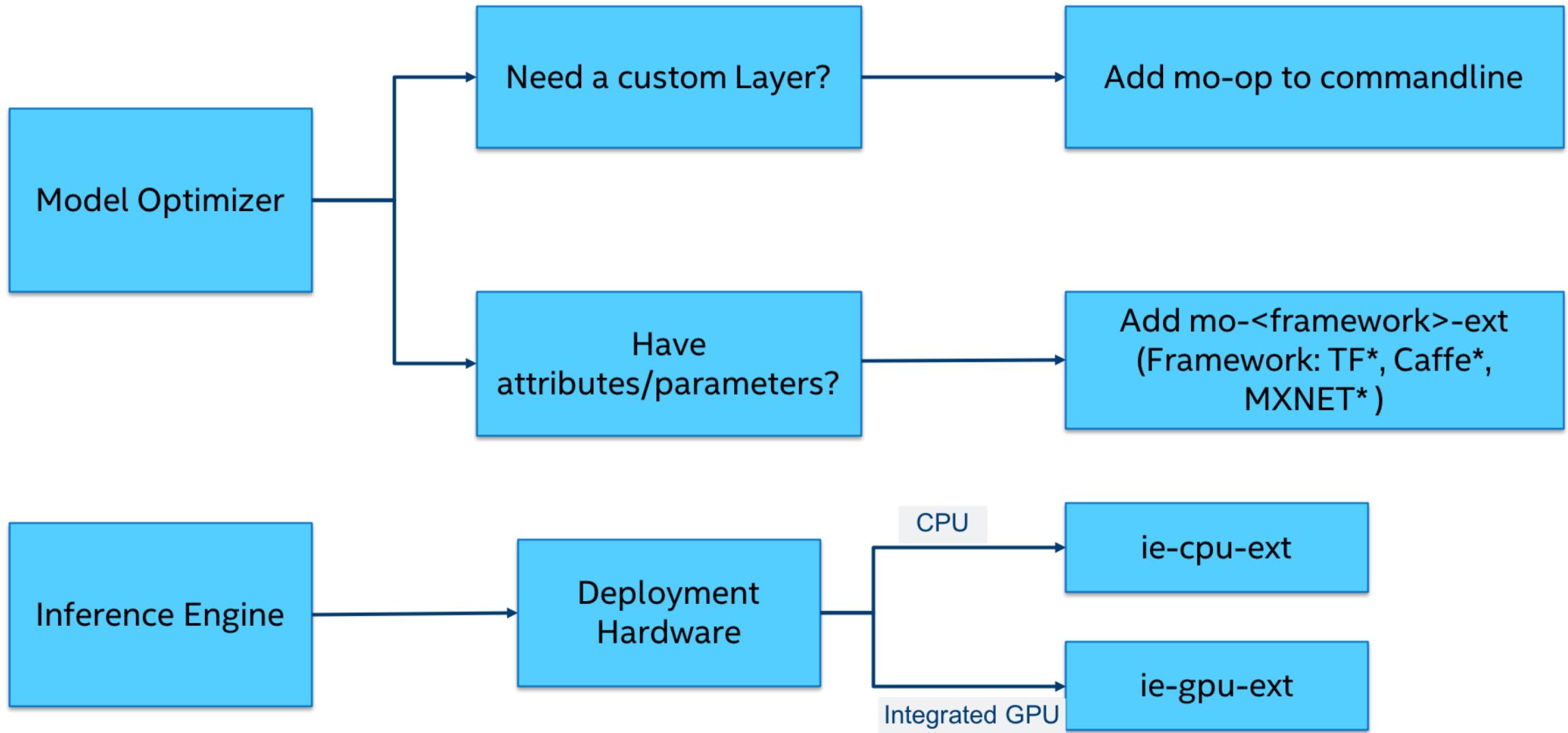
To run the tool in the interactive mode, specify the following parameters:

- mo-op - To generate a Model Optimizer operation
- mo-caffe-ext - To generate a Model Optimizer Caffe* extractor
- mo-mxnet-ext - To generates a Model Optimizer MXNet* extractor
- mo-tf-ext - To generate a Model Optimizer TensorFlow* extractor
- ie-cpu-ext - To generate an Inference Engine CPU extension
- ie-gpu-ext - To generate an Inference Engine GPU extension
- output_dir - To set an output directory. If not specified, the current directory is used by default.

Example

```
python3 extgen.py new mo-op mo-tf-ext ie-cpu-ext
```

Extension Generator Flow Chart



Extending Custom Layer to Model Optimizer

Directory Structure for Model Optimizer and Inference Engine extension files

- `/user_mo_extensions`
 - `__init__.py`
- `/front`
 - `caffe`
 - `__init__.py`
 - `mxnet`
 - `__init__.py`
 - `tf`
 - `__init__.py`
- `/ops`
 - `__init__.py`
- `/user_ie_extensions`
 - `cpu`
 - `gpu`



Extraction of Model's attributes for the targeted framework



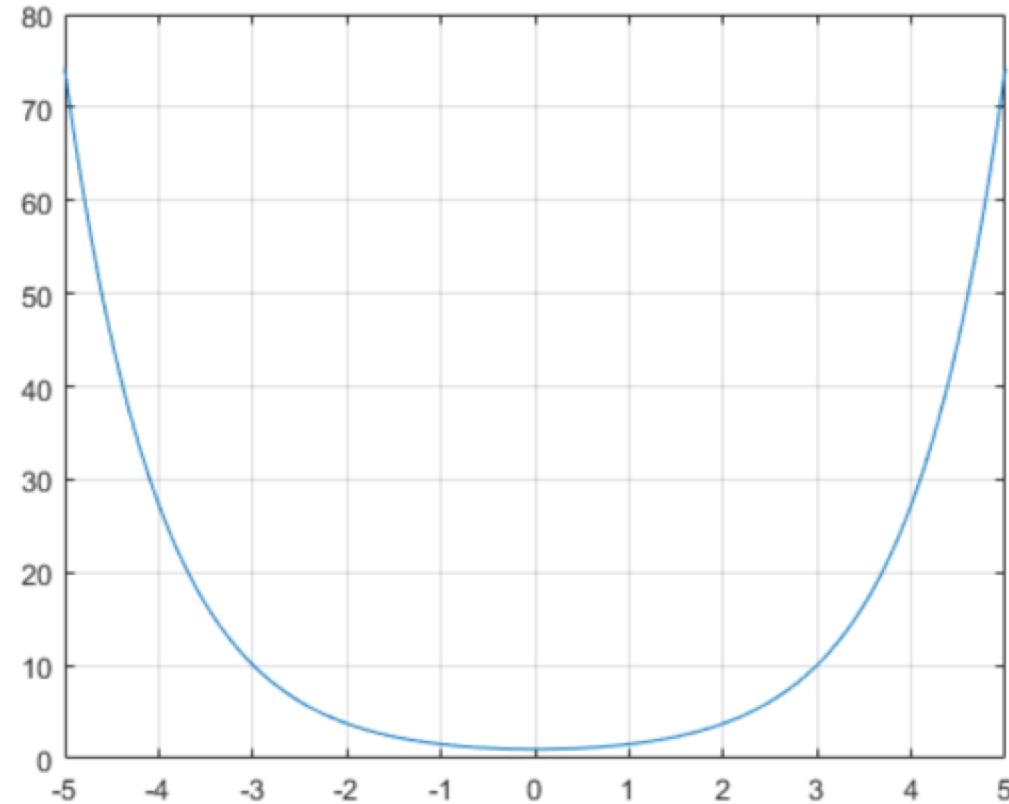
Custom Layer implementation to calculate output shapes and attributes in IR



Inference Implementation of Custom Layer

Custom Layer Example: Cosh – Hyperbolic cosine

$$\cosh x = \frac{e^x + e^{-x}}{2}$$



Ops Folder-cos.py

```
17 from mo.front.common.partial_infer.elemental import copy_shape_infer
18 from mo.ops.op import Op
19 from mo.graph.graph import Node
20
21 class Cosh(Op):
22     op = 'Cosh'
23     enabled = True
24
25     def __init__(self, graph, attrs: dict):
26         super().__init__(graph, {
27             'type': __class__.op,
28             'op': __class__.op,
29             'infer': Cosh.infer,
30         }, attrs)
31
32     @staticmethod
33     def infer(node: Node):
34         # we just set the same shape to the output
35         copy_shape_infer(node)
```

Represents if the operation should be used by Model Optimizer or excluded

Constructor is automatically generated

Sets the Output shape to the Input shape

Front/<Framework>: cosh_ext.py

.. Framework: TensorFlow*, Caffe*, MXNet*, ONNX*

```
from mo.front.extractor import FrontExtractorOp
from mo.ops.cosh import Cosh

class CoshFrontExtractor(FrontExtractorOp):
    op = 'Cosh'
    enabled = True

    @staticmethod
    def extract(node):
        Cosh.update_node_stat(node)
        return __class__.enabled
```



Constructor is automatically generated

Generate IR with Custom Layer files using Model Optimizer

- The --extension arguments are used by Model Optimizer to parse the files in the directory provided on the command line as shown below.

```
python3 mo.py --input_model <model> --extension <path to directory with files>
```

```
python3 mo.py --input_model <model> --extension .../extension_generator/
```

Extending Custom Layer to Inference Engine

Inference Engine CPU extension (.cpp)

```
namespace InferenceEngine {
namespace Extensions {
namespace Cpu {

class CoshImpl: public ExtLayerBase {
public:
    explicit CoshImpl(const CNNLayer* layer) {
        try {
            if (layer->insData.size() != 1 || layer->outData.empty())
                THROW_IE_EXCEPTION << "Incorrect number of input/output edges!";

            addConfig(layer, {{ConfLayout::PLN, false, 0}}, {{ConfLayout::PLN, false, 0}});
        } catch (InferenceEngine::details::InferenceEngineException &ex) {
            errorMsg = ex.what();
        }
    }
}
}
```



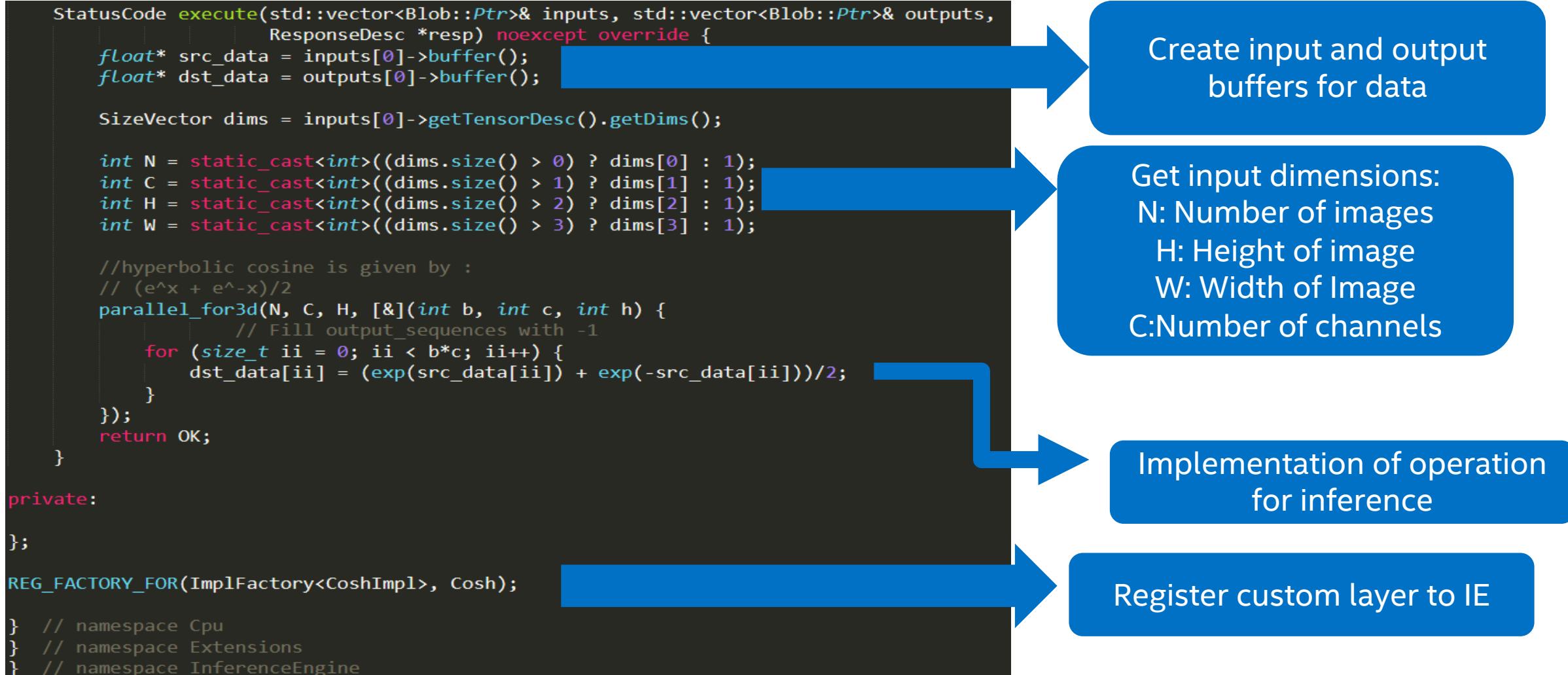
AddConfig

- `addConfig(layer, {{ConfLayout::PLN, false, 0}}, {{ConfLayout::PLN, false, 0}});`

`addConfig(layerpointer, DataConfig Vector(input), DataConfig Vector(Output))`

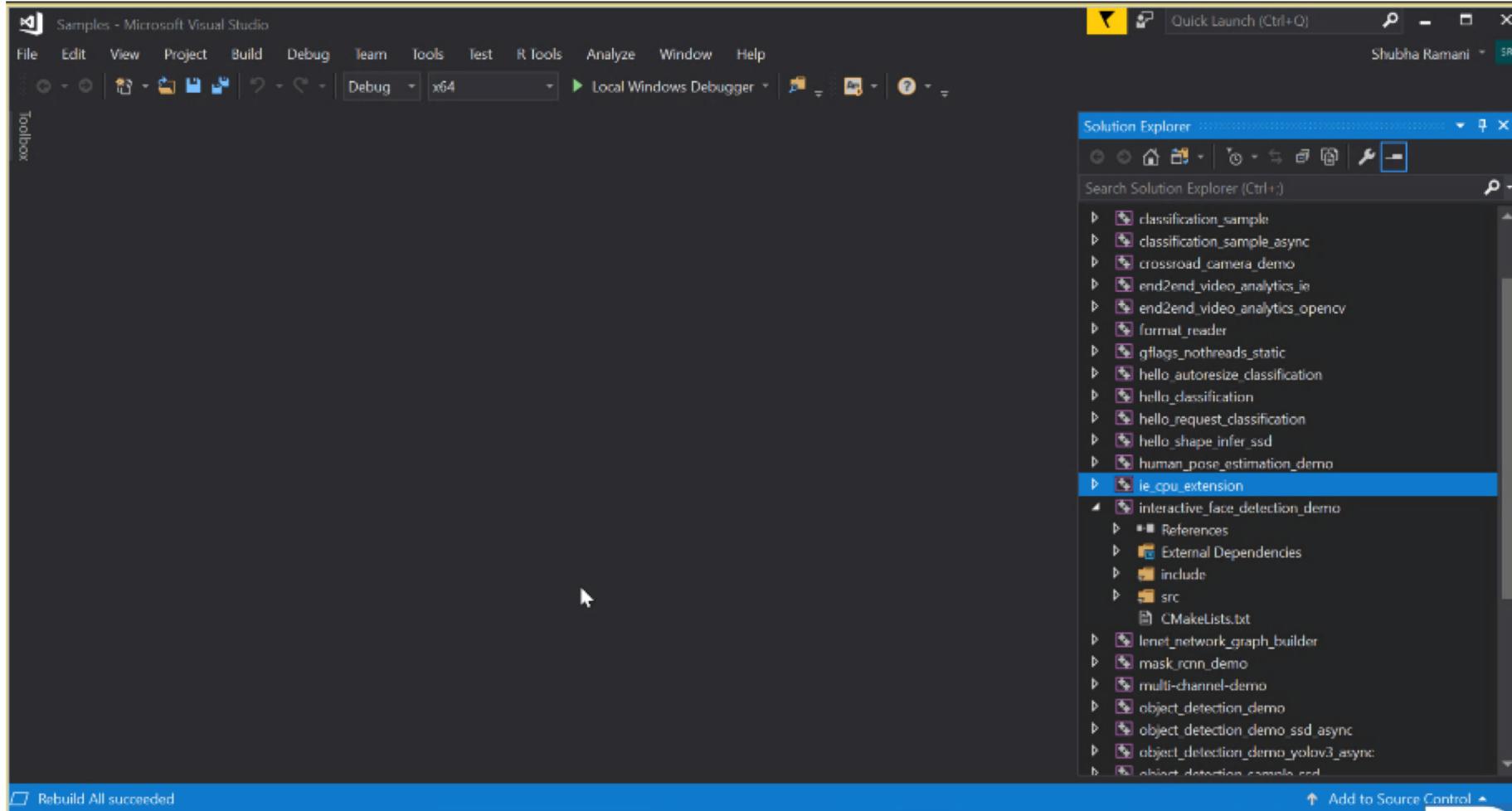
- Layer pointer
- ConfLayout is the memory layout for the input and output data which can be PLN(planar), ANY(finds the best suitable option), etc. for more information look at `IE_Common.h`
- False indicates that the flag for determination of the constant memory is disabled
- 0 indicates the index of in-place memory. If -1 memory cannot be in-place.

Inference Engine CPU extension Cont. (.cpp)



Compiling CPU custom Layer

- Recompile the cpu_extension project
- The cpu_extension.so or .dll gets updated



Inference Engine GPU custom layer (.cl) . ∴ Custom Layer == Kernel

```
15 #pragma OPENCL EXTENSION cl_khr_fp16 : enable  
16  
17 __kernel void cosh(const __global INPUT0_TYPE* input,  
18                      __global OUTPUT0_TYPE* output)  
19 {  
20     const int dims = sizeof(INPUT0_DIMS) / sizeof(INPUT0_DIMS[0]);  
21     int T_ = INPUT0_DIMS[0];  
22     int N_ = INPUT0_DIMS[1];  
23     int C_ = INPUT0_DIMS[2];  
24  
25     // Fill output_sequences with -1  
26     for (int ii = 0; ii < T_*N_; ii++)  
27     {  
28         output[ii] = (OUTPUT0_TYPE)(exp(input[ii]) + exp(-input[ii]))/2;  
29     }  
30 }
```

Directive enables half floating point support

Get input dimensions:
The format of the tensor, BFYX, BYXF, YXFB ,
FYXB or ANY
B(T_): # of images
F(N_):number of channels/features
X:width
Y(C_): height

Implementation of operation for inference

Inference Engine GPU custom layer Cont. (.xml)

- Registration of Custom Layer to Inference Engine

```
<CustomLayer name="Cosh" type="SimpleGPU" version="1">
  <Kernel entry="cosh">
    <Source filename="cosh.cl"/>
  </Kernel>
  <Buffers>
    <Tensor arg-index="0" type="input" port-index="0" format="BFYX"/>
    <Tensor arg-index="1" type="output" port-index="1" format="BFYX"/>
  </Buffers>

  <CompilerOptions options="-cl-mad-enable"/>
  <WorkSizes global="B,F"/>
</CustomLayer>
```

Configuration of
Custom Layer:

- name - name of the layer type should be identical to the name in IR.
- type and version must always be these values shown.

Inference Engine GPU custom layer Cont. (.xml)

- Registration of Custom Layer to Inference Engine

```
<CustomLayer name="Cosh" type="SimpleGPU" version="1">
  <Kernel entry="cosh">
    <Source filename="cosh.cl"/>
  </Kernel>
  <Buffers>
    <Tensor arg-index="0" type="input" port-index="0" format="BFYX"/>
    <Tensor arg-index="1" type="output" port-index="1" format="BFYX"/>
  </Buffers>

  <CompilerOptions options="-cl-mad-enable"/>
  <WorkSizes global="B,F"/>
</CustomLayer>
```

filename – Name of the file containing OpenCL™ source code. Path is relative to your executable.

Running Samples with Custom Layers Libraries

```
<sample.exe> -i <path to image> -m <path to model> -l <path to custom layer library> -d CPU
```

```
<sample.exe> -i <path to image> -m <path to model> -c <path_to_xml_config_file> -d GPU
```

