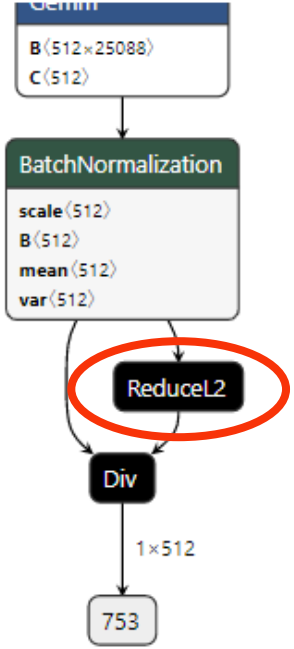
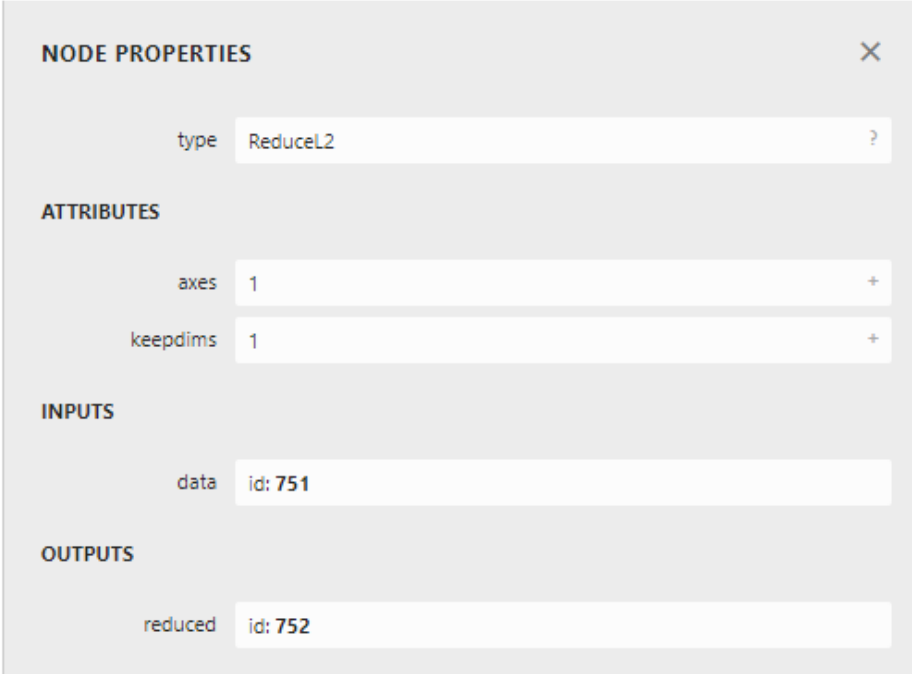
Steps on Adding Support for Unsupported Layers in The Intel® OpenVINO™ Toolkit

Setup

- OpenVINO version
  - 2020.2
- Unsupported layer
  - Operation: ReduceL2
  - Framework: ONNX
  - Convert it as:

    Custom ReduceL2





Model Optimizer

- To enable unsupported layer into MO, we need to add or patch some files to enable it.
  - IR Operation
    - The operation for IR need to follow **ORIGINAL** operation to save some attributes and **infer function for shape** and **value**.
  - Operation Extractor
    - To extract operation and parameter from Original operation and save to IR operation.
  - Register Operation Version
    - To link the ngraph operation between MO and IE.

Model Optimizer – IR Operation

ReduceOps.py dir: ${MO_ROOT}/extensions/ops/ReduceOps.py

- Following Operation and its attribute to create the IR Op.
  - Here **using existing ReduceOps to extend** if no similar operation on MO, it should create a new file such as previous OpenVINO version to extend it.
    - During create the operation, **if the data type of attribute is int, must be as int64 for IR v10** since ngraph doesn't support **int32**.
  - For previous version, you can check files which listed on below from RDC or previous version documents from OpenVINO page.
    - Steps to Add an Unsupported TensorFlow Layer into OpenVINO
      - https://cdrdv2.intel.com/v1/dl/getContent/613858
    - Steps to Add an Unsupported ONNX* Layer into OpenVINO
      - https://cdrdv2.intel.com/v1/dl/getContent/613857

Model Optimizer – IR Operation – Cont.

ReduceOps.py dir: ${MO_ROOT}/extensions/ops/ReduceOps.py

- Introduce class **ReduceOp**

```
class ReduceOp(Op):
    enabled = False
    op = None
    op_type = None

    def __init__(self, graph: Graph, attrs: dict):
        super().__init__(graph, {
            'op': self.op,
            'type': self.op_type,
            'infer': reduce_infer,
            'keep_dims': 0,
            'in_ports_count': 2,
            'out_ports_count': 1,
            'force_precision_in_ports': {
                1: 'int64' if graph.graph['cmd_params'].generate_experimental_IR_V10 else 'int32'},
        }, attrs)
        assert isinstance(self.attrs['keep_dims'], int) or isinstance(self.attrs['keep_dims'], bool)
        self.attrs['keep_dims'] = bool(self.attrs['keep_dims'])

    def supported_attrs(self):
        return [
            ('keep_dims', lambda node: str(node.keep_dims)),
        ]
```

A class for inherit, so no need to enable.

Op, type and infer function

Needs parameter, and its default value

Number of I/O node

Set INT64 for IR v10

Operation attribute for inference (IE), will save into IR.

Model Optimizer – IR Operation – Cont.

ReduceOps.py dir: ${MO_ROOT}/extensions/ops/ReduceOps.py

- Create **ReduceL2** class which **inherit from ReduceOp** for ReduceL2

```
class ReduceL2(ReduceOp):
    enabled = True
    op = 'CustomReduceL2'
    op_type = 'CustomReduceL2'
```

Enable CustomReduceL2 Op

- Enable **value infer** function for ReduceL2

```
reduce_map = {
    'ReduceSum': np.sum,
    'ReduceProd': np.prod,
    'ReduceMax': np.max,
    'ReduceMin': np.min,
    'ReduceMean': np.mean,
    'ReduceAnd': np.all,
    'ReduceLogicalAnd': np.all,
    'ReduceLogicalOr': np.any,
    'CustomReduceL2': (np.sqrt, np.sum, np.square),
}
```

Functions for value infer

Model Optimizer – IR Operation – Cont.

- Enable **value infer** function for CustomReduceL2 – Cont.

```
if in_value is not None:
    if node.op == 'CustomReduceL2':
        value = reduce_map[node.op][0](reduce_map[node.op][1](reduce_map[node.op][2](in_value.copy()), axis=tuple(axis), keepdims=node.keep_dims))
    else:
        value = reduce_map[node.op](in_value.copy(), axis=tuple(axis), keepdims=node.keep_dims)
    node.out_port(0).data.set_value(value)
```

- Enable shape infer function for CustomReduceL2
  - No need to implement, same as another Reduce function

Model Optimizer - Operation Extractor

reduce_l2_ext.py dir: ${MO_ROOT}/extensions/ front/onnx/reduce_l2_ext.py

- Create file reduce_l2_ext.py and implement extractor

```
from extensions.ops.ReduceOps import ReduceL2
from mo.front.extractor import FrontExtractorOp
from mo.front.common.partial_infer.utils import int64_array
from mo.front.onnx.extractors.utils import onnx_attr
from mo.graph.graph import Node

class ReduceL2Extractor(FrontExtractorOp):
    op = 'ReduceL2'
    enabled = True

    def extract(cls, node: Node):
        axis = onnx_attr(node, 'axes', 'ints', default=None, dst_type=lambda x: int64_array(x))
        keep_dims = onnx_attr(node, 'keepdims', 'i', default=True)
        [redacted]

        ReduceL2.update_node_stat(node, {'axis': axis, 'keep_dims': keep_dims})

        return cls.enabled
```

type	ReduceL2
ATTRIBUTES	
axes	1
keepdims	1

Import reducel2 operation

Enable extractor

Extract attributes

Create and save attribute into Reducel2 IR operation

Model Optimizer - Register Operation Version

op_versioning.py dir: ${MO_ROOT}/extensions/back/op_versioning.py

- After IR v10, each operation has its version and it defined by the MO via file, **op_versioning.py**.
  - If version unset, the MO will set it version as "**extension**"
  - Define the version sets for operations

```
class OpVersioning(BackReplacementPattern):
    enabled = False

    opset_1_types = set(map(lambda s: s.lower(), [
        "Abs",
        "Acos",
        "Add",
        "Asin",

        ...
```

  - Assign version to operations

```
    def find_and_replace_pattern(self, graph: Graph):
        for node in graph.get_op_nodes():
            node_type = node.soft_get('type').lower()
            name = node.soft_get('name', node.id)

            if node.soft_get('version', None) == 'opset1' and node_type not in self.opset_1_types:
                raise Error('Node {} has `version` attribute set to `opset1`, but it is a reserved word, '
                            'please use another'.format(name))

            if not node.has_valid('version'):
                if node_type in self.opset_1_types:
                    node['version'] = 'opset1'
```

Set Op version

Model Optimizer - Register Operation Version – Cont.

- Add **opsets** which include **CustomReduceL2** operation into class ‘OpVersioning’

```
opsets = set(map(lambda s: s.lower(), [
    "CustomReduceL2",
]))
```

- Assign version **john_custom** to **CustomReduceL2** operation which under **opsets**

```
def find_and_replace_pattern(self, graph: Graph):
    for node in graph.get_op_nodes():
        node_type = node.soft_get('type').lower()
        name = node.soft_get('name', node.id)

        if node.soft_get('version', None) == 'opset1' and node_type not in self.opset_1_types:
            raise Error('Node {} has `version` attribute set to `opset1`, but it is a reserved word, '
                        'please use another'.format(name))

        if not node.has_valid('version'):
            if node_type in self.opset_1_types:
                node['version'] = 'opset1'
            elif node_type in self.opset_1_experimental_ops:
                node['version'] = 'experimental'
            elif node_type in self.opset_2_legacy_ops:
                node['version'] = 'opset2'
            elif node_type in self.opsets:
                node['version'] = 'john_custom'
```

```
<layer id="579" name="752" type="CustomReduceL2" version="john_custom">
  <data keep_dims="True"/>
  <input>
    <port id="0">
      <dim>1</dim>
      <dim>512</dim>
    </port>
    <port id="1">
      <dim>1</dim>
    </port>
  </input>
```

IE Extension

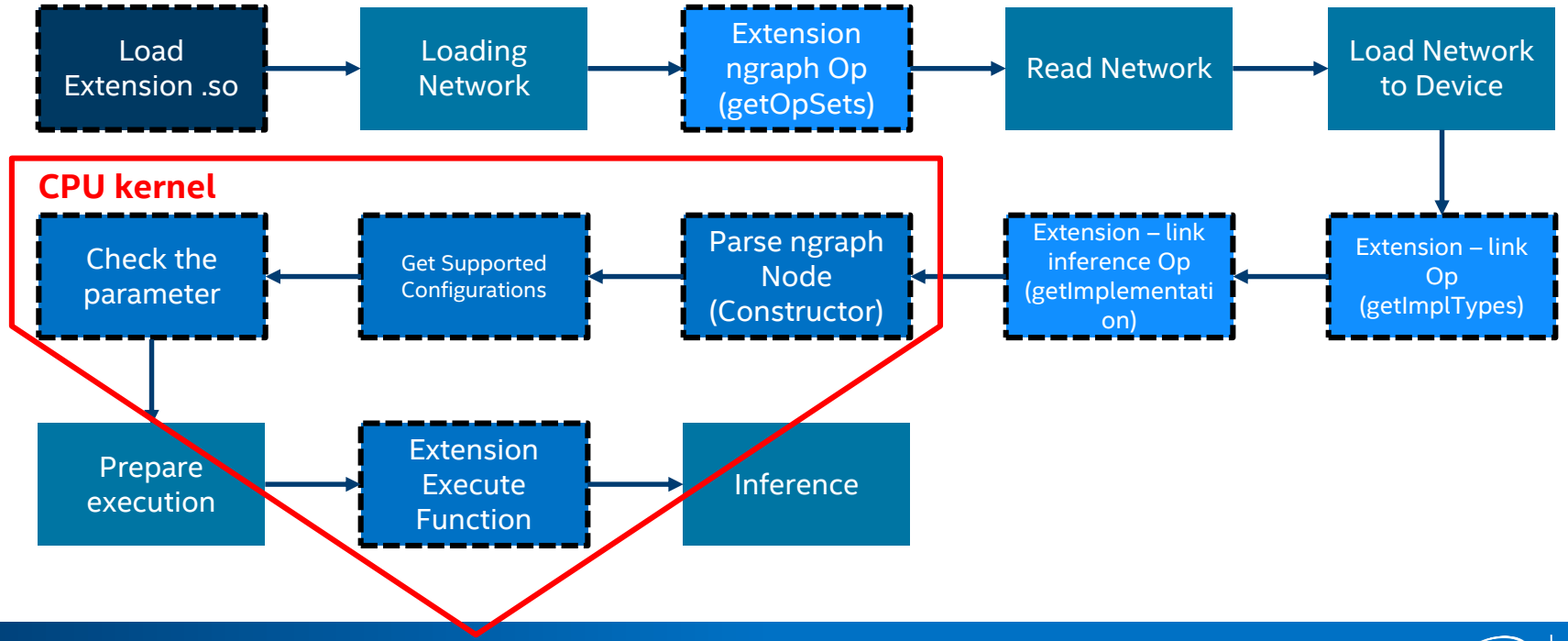
- To enable unsupported layer into IE, we need to enable 3 classes below.
  - Extension Interface
    - The interface for extension library, it can link the ngraph operation and cpu kernel.

      Reference: https://docs.openvinotoolkit.org/latest/extension.html
  - ngraph Operation
    - Create a ngraph operation with operation definition and shape inference method.

      Reference: https://docs.openvinotoolkit.org/latest/custom_operations.html
  - CPU Kernel Operation(WIP)
    - Create a CPU kernel operation with operation definition and inference function.

      Reference: https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Extensibility_DG_CPU_Kernel.html
      - **This sample will use one file to implement.**

Workflow for IE Extension



IE Extension - #include

- Create custom_reduce_l2.cpp

```
#include <map>
#include <memory>
#include <string>
#include <algorithm>
#include <vector>

#include <ie_layouts.h>
#include <ie_iextension.h>
#include <ie_blob.h>
#include <details/ie_exception.hpp>


// #include <ngraph/op/op.hpp>
#include <ngraph/axis_set.hpp>
#include <ngraph/op/util/arithmetic_reductions_keep_dims.hpp>
#include <ngraph/node.hpp>
#include <ngraph/opsets/opset.hpp>
#include <ngraph/graph_util.hpp>
#include <ngraph/op/broadcast.hpp>
```

For ngraph Operation, but here will not use this file. Will use "arithmetic_reductions_keep_dims.hpp" since it is more suitable ReduceL2.

IE Extension - Extension Interface

```
class ReduceL2Extension : public InferenceEngine::IExtension {
public:
    ReduceL2Extension() {
        impls["CustomReduceL2"] = [](const std::shared_ptr<ngraph::Node>& node) -> InferenceEngine::ILayerImpl::Ptr {
            return std::make_shared<ReduceL2Impl>(node);
        };
    }

    void GetVersion(const InferenceEngine::Version*& versionInfo) const noexcept override {}

    void Unload() noexcept override {}

    void Release() noexcept override {}

    std::vector<std::string> getImplTypes(const std::shared_ptr<ngraph::Node>& node) override {
        if (impls.find(node->description()) == impls.end())
            return {};
        return {"CPU"};
    }
```

For link IR Operation

For link Operation to run on CPU

Set the run device, CPU, for ngraph operation

IE Extension - Extension Interface – Cont.

```
InferenceEngine::ILayerImpl::Ptr getImplementation(const std::shared_ptr<ngraph::Node>& node, const std::string& implType) override {
    if (impls.find(node->description()) == impls.end() || implType != "CPU")
        return nullptr;
    return impls[node->description()](node);
}

std::map<std::string, ngraph::OpSet> getOpSets(){
    std::map<std::string, ngraph::OpSet> opsets;
    ngraph::OpSet opset;
    opset.insert<ReduceL2Op>();
    opsets["john_custom"] = opset;
    return opsets;
}

rivate:
    std::map<std::string, std::function<InferenceEngine::ILayerImpl::Ptr(const std::shared_ptr<ngraph::Node>)>> impls;
;
```

For link Operation for inference

For link ngraph Operation

Setup ngraph operation. The ngraph op class is ReduceL2, and operation version is “john_custom”.

- The ngraph op will implement later and the version is set on MO.

IE Extension - Extension Interface – Cont.

```
INFERENCE_EXTENSION_API(InferenceEngine::StatusCode) InferenceEngine::CreateExtension(InferenceEngine::IExtension *&ext,
                                                                                       InferenceEngine::ResponseDesc *resp) noexcept {
    try {
        ext = new ReduceL2Extension();
        return OK;
    } catch (std::exception &ex) {
        if (resp) {
            std::string err = ((std::string) "Couldn't create extension: ") + ex.what();
            err.copy(resp->msg, 255);
        }
        return InferenceEngine::GENERAL_ERROR;
    }
}
```

Create extension (if didn't implement this function, the extension function will not execute)

IE Extension - ngraph Operation

```cpp
#define REDUCE_L2_TYPE "CustomReduceL2"

class ReduceL2Op :  public ngraph::op::util::ArithmeticReductionKeepDims {
public:
    static constexpr ngraph::NodeTypeInfo type_info{REDUCE_L2_TYPE, 0};
    const ngraph::NodeTypeInfo& get_type_info() const override { return type_info; }

    ReduceL2Op() = default;

    explicit ReduceL2Op(const ngraph::Output<Node>& arg, const ngraph::Output<Node>& reduction_axes, bool keep_dims = false)
        : ArithmeticReductionKeepDims(arg, reduction_axes, keep_dims)
    {
        constructor_validate_and_infer_types(); // Runs shape inference on this op by invoking validate_and_infer_types()
    }

    std::shared_ptr<Node> copy_with_new_args(const ngraph::NodeVector& new_args) const override
    {
        check_new_args_count(this, new_args);
        return std::make_shared<ReduceL2Op>(new_args.at(0), new_args.at(1), get_keep_dims());
    }
```

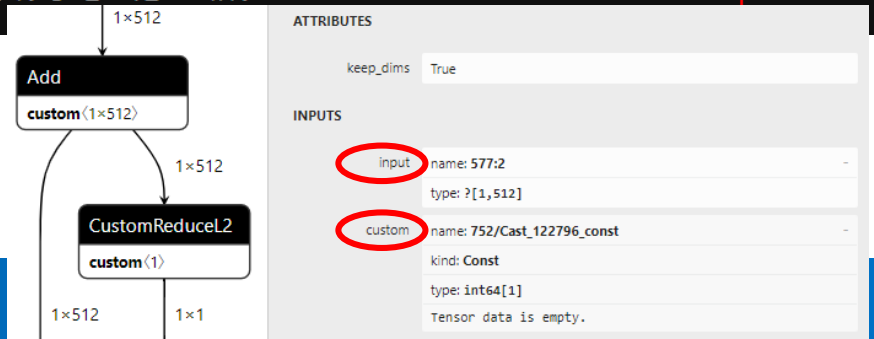
In general, will inherit the class "**Op**", but I use a similar one to replace it. So, it will have some different with general step to construct unsupported operation.

Define Operation type which same as IR name

Setup output node and attribute.

- In here, output node means the another operation output node link to this ngraph operation.

New args number is same as output node which from another operation.



IE Extension - ngraph Operation – Cont.

```python
class ReduceOp(Op):
    enabled = False
    op = None
    op_type = None

    def __init__(self, graph: Graph, attrs: dict):
        super().__init__(graph, {
            'op': self.op,
            'type': self.op_type,
            'infer': reduce_infer,
            'keep_dims': 0,
```

MO Operation

```
    std::shared_ptr<Node> get_default_value() const override
    {
        return ngraph::make_constant_from_string("0", get_element_type(), get_shape());
    }

    void validate_and_infer_types() override
    {
        ArithmeticReductionKeepDims::validate_and_infer_types();
    }

protected:
    void generate_adjoints(ngraph::autodiff::Adjoints& adjoints, const ngraph::OutputVector& deltas) override
    {
        auto delta = deltas.at(0);

        auto x = input_value(0);
        auto& x_shape = x.get_shape();

        adjoints.add_delta(x, std::make_shared<ngraph::op::Broadcast>(delta, x_shape, get_reduction_axes()));
    }
};
```

Get default value, here same as MO

Shape Inference function

Optional. Since the Reduce have 2 input, so need this function to make this operation.

IE Extension - Cpu Kernel Operation

Create class for execute and its constructor

```
constexpr ngraph::NodeTypeInfo ReduceL2Op::type_info;


class ReduceL2Impl : public InferenceEngine::ILayerExecImpl {
public:
    explicit ReduceL2Impl(const std::shared_ptr<ngraph::Node>& node) : _node(node)
    {
        auto castedNode = std::dynamic_pointer_cast<ReduceL2Op>(node);
        keep_dims = castedNode->get_keep_dims();
    }
```

Parse ngraph Node

IE Extension - Cpu Kernel Operation – Cont.

Function – getSupportedConfigurations

```
InferenceEngine::StatusCode getSupportedConfigurations(std::vector<InferenceEngine::LayerConfig> &conf,
                                                       InferenceEngine::ResponseDesc *resp) noexcept
{
    InferenceEngine::LayerConfig layerConfig;
    layerConfig.dynBatchSupport = true;

    InferenceEngine::DataConfig inData;
    InferenceEngine::DataConfig inData_const;
    InferenceEngine::DataConfig outData;

    InferenceEngine::SizeVector inOrder;
    InferenceEngine::SizeVector inOrder_const;
    InferenceEngine::SizeVector outOrder;

    size_t offset((std::numeric_limits<size_t>::max)());

    auto inShape = _node->get_input_shape(0);
    for (size_t i = 0; i < inShape.size(); i++) {
        inOrder.push_back(i);
    }

    auto inShape_const = _node->get_input_shape(1);
    for (size_t i = 0; i < inShape_const.size(); i++) {
        inOrder_const.push_back(i);
    }

    auto outShape = _node->get_output_shape(0);
    for (size_t i = 0; i < outShape.size(); i++) {
        outOrder.push_back(i);
    }
```

Returns all supported configuration formats (input and output tensor layouts)

Get input and output shape and its order. Ex: shape is 1x512x2x2, order is {0, 1, 2, 3}.

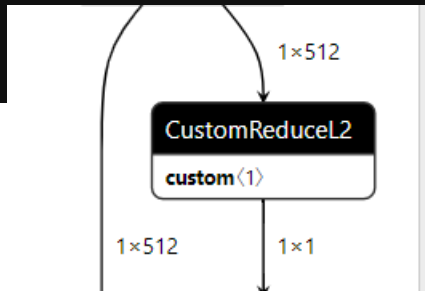
IE Extension - Cpu Kernel Operation – Cont.

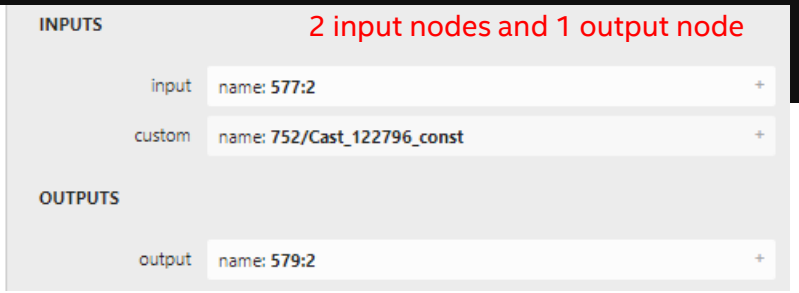
Function – getSupportedConfigurations – Cont.

```
    inData.desc = InferenceEngine::TensorDesc(InferenceEngine::Precision::FP32,
                                              inShape, {inShape, inOrder});

    inData_const.desc = InferenceEngine::TensorDesc(InferenceEngine::Precision::I32,
                                              inShape_const, {inShape_const, inOrder_const});

    outData.desc = InferenceEngine::TensorDesc(InferenceEngine::Precision::FP32,
                                              outShape, {outShape, outOrder});

    layerConfig.outConfs.push_back(outData);

    layerConfig.inConfs.resize(2);
    layerConfig.inConfs[0] = inData;
    layerConfig.inConfs[1] = inData_const;

    conf.push_back(layerConfig);
    return InferenceEngine::OK;
}
```

Set tensor descriptor, Data Precision, Shape, and Order.

Follow the layout set the layer config.





IE Extension - Cpu Kernel Operation – Cont.

Function – init

```
InferenceEngine::StatusCode init(InferenceEngine::LayerConfig &config, InferenceEngine::ResponseDesc *resp) noexcept {
    return InferenceEngine::OK;
}
```

For check the parameter

- You can check the input/output config size, dimension, etc.

IE Extension - Cpu Kernel Operation – Cont.

Function - execute

```
    InferenceEngine::StatusCode execute(std::vector<InferenceEngine::Blob::Ptr> &inputs,
                                        std::vector<InferenceEngine::Blob::Ptr> &outputs,
                                        InferenceEngine::ResponseDesc *resp) noexcept {

        int32_t *idx_data = inputs[REDUCE_INDEXES]->cbuffer().as<int32_t *>() +
                            inputs[REDUCE_INDEXES]->getTensorDesc().getBlockingDesc().getOffsetPadding();

        const float* src_data = inputs[REDUCE_DATA]->cbuffer().as<const float *>() +
                                inputs[REDUCE_DATA]->getTensorDesc().getBlockingDesc().getOffsetPadding();

        float* dst_data = outputs[0]->buffer();

        /*****************************************************************************************/
        /*                                                                                       */
        /*                                    Code for Execution                                 */
        /*                                                                                       */
        /*****************************************************************************************/

        return InferenceEngine::OK;
    }

private:
    const std::shared_ptr<ngraph::Node> _node;

    const size_t REDUCE_INDEXES = 1;
    const size_t REDUCE_DATA = 0;
    bool keep_dims;
};
```

Get input axis buffer

Get the input tensor buffer

Get the output tensor buffer

Determine the layer here, please check this file for execution code. https://github.com/openvinotoolkit/openvino/blob/95a57795dcb0ed82324af87918339225148202ca/inference-engine/src/mkldnn_plugin/nodes/reduce.cpp

Compile it as .so

- CMakelists.txt

```
set(TARGET_NAME "ReduceL2")

find_package(ngraph REQUIRED)
find_package(InferenceEngine REQUIRED)

file(GLOB_RECURSE SRC *.cpp)

add_library(${TARGET_NAME} SHARED ${SRC})

target_include_directories(${TARGET_NAME} PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})

target_compile_definitions(${TARGET_NAME} PRIVATE IMPLEMENT_INFERENCE_EXTENSION_API)
target_link_libraries(${TARGET_NAME} PRIVATE ${NGRAPH_LIBRARIES} ${InferenceEngine_LIBRARIES})
```

Compile it as .so – Cont.

- Build
  - cmake . && make
- Output

```
john-nuc@johnnuc-NUC7i7BNH:~/Desktop/bb/cc$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  libReduceL2.so  Makefile
```

Result

- ngraph operation version is incorrect (MO and IE)
  - ➢ Cannot create Operation

```
john-nuc@johnnuc-NUC7i7BNH:~/inference_engine_cpp_samples_build/intel64/Release$ ./benchmark_app -m ~/Downloads/moment_test/moment-in-time.xml -d CPU -l ~/Desktop/bb/cc/libReduceL2.so
[Step 1/11] Parsing and validating input arguments
[ INFO ] Parsing input parameters
[ WARNING ] -nstreams default value is determined automatically for a device. Although the automatic selection usually provides a reasonable performance,but it still may be non-optimal
README.

[Step 2/11] Loading Inference Engine
LeFeng getOpSets
[ INFO ] CPU (MKLDNN) extensions is loaded /home/john-nuc/Desktop/bb/cc/libReduceL2.so
[ INFO ] InferenceEngine:
        API version ............ 2.1
        Build .................. 42025
        Description ....... API
[ INFO ] Device info:
        CPU
        MKLDNNPlugin version ......... 2.1
        Build ........... 42025

[Step 3/11] Setting device configuration
[Step 4/11] Reading the Intermediate Representation network
[ INFO ] Loading network files
LeFeng getOpSets
[ ERROR ] Cannot create ReduceL2 layer 752 id:579
```

Result – Cont.

- Don’t using existing layer version

```
john-nuc@johnnuc-NUC7i7BNH:~/inference_engine_cpp_samples_build/intel64/Release$ ./benchmark_app -m ~/Desktop/moment-in-time.xml -d CPU -l /tmp/cc/cc/libReduceL2.so
[Step 1/11] Parsing and validating input arguments
[ INFO ] Parsing input parameters
[ WARNING ] -nstreams default value is determined automatically for a device. Although the automatic selection usually provides a reasonable performance,but it still
README.

[Step 2/11] Loading Inference Engine
LeFeng getOpSets
[ ERROR ] Cannot add opset with name: opset1. Opset with the same name already exists.
```

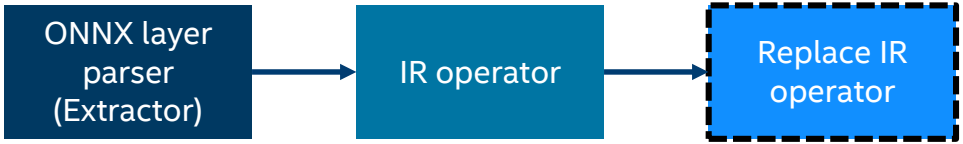
Result – Cont.

- ngraph operation version is correct and ie extension is worked(MO and IE)

```
john-nuc@johnnuc-NUC7i7BNH:~/Desktop/kkk/dldt/bin/intel64/Release$ ./benchmark_app -m ~/Desktop/custom_reducel2/moment-in-time.xml -d CPU -l ~/Desktop/custom_reduce_qq/ss/libReduceL2.so
[Step 1/11] Parsing and validating input arguments
[ INFO ] Parsing input parameters
[ WARNING ] -nstreams default value is determined automatically for a device. Although the automatic selection usually provides a reasonable performance,but it still may be non-optimal f
README.

[Step 2/11] Loading Inference Engine
LeFeng getOpSets
[ INFO ] CPU (MKLDNN) extensions is loaded /home/john-nuc/Desktop/custom_reduce_qq/ss/libReduceL2.so
[ INFO ] InferenceEngine:
        API version ............ 2.1
        Build .................. custom_2020_95a57795dcb0ed82324af87918339225148202ca
        Description ....... API
[ INFO ] Device info:
        CPU
        MKLDNNPlugin version ......... 2.1
        Build ........... custom_2020_95a57795dcb0ed82324af87918339225148202ca

[Step 3/11] Setting device configuration
[Step 4/11] Reading the Intermediate Representation network
          ...
[Step 11/11] Dumping statistics report
Count:      864 iterations
Duration:   60246.31 ms
Latency:    279.01 ms
Throughput: 14.34 FPS
```

Using existing layer replace ReduceL2

- Except for create an extension library, it also can use existing layer to replace it.



  - Add Replacement file for replace operator.

- ReduceL2

$$f(x) = \sqrt{\sum x^2}$$

  - Replace: $\sqrt{}$, $\sum$, and x^2. → **Pow**, **ReudceSum**, and **Pow**.

Model Optimizer - Operation Extractor

reduce_l2_ext.py dir: ${MO_ROOT}/extensions/ front/onnx/reduce_l2_ext.py

- Create file reduce_l2_ext.py and implement extractor

```
from extensions.ops.ReduceOps import ReduceL2
from mo.front.extractor import FrontExtractorOp
from mo.front.common.partial_infer.utils import int64_array
from mo.front.onnx.extractors.utils import onnx_attr
from mo.graph.graph import Node

class ReduceL2Extractor(FrontExtractorOp):
    op = 'ReduceL2'
    enabled = True

    def extract(cls, node: Node):
        axis = onnx_attr(node, 'axes', 'ints', default=None, dst_type=lambda x: int64_array(x))
        keep_dims = onnx_attr(node, 'keepdims', 'i', default=True)
        [redacted]

        ReduceL2.update_node_stat(node, {'axis': axis, 'keep_dims': keep_dims})

        return cls.enabled
```

type	ReduceL2
ATTRIBUTES	
axes	1
keepdims	1

Import reducel2 operation

Enable extractor

Extract attributes

Create and save attribute into Reducel2 IR operation

Model Optimizer – IR Operation

ReduceOps.py dir: ${MO_ROOT}/extensions/ops/ReduceOps.py

- Introduce class **ReduceOp**

```
class ReduceOp(Op):
    enabled = False
    op = None
    op_type = None

    def __init__(self, graph: Graph, attrs: dict):
        super().__init__(graph, {
            'op': self.op,
            'type': self.op_type,
            'infer': reduce_infer,
            'keep_dims': 0,
            'in_ports_count': 2,
            'out_ports_count': 1,
            'force_precision_in_ports': {
                1: 'int64' if graph.graph['cmd_params'].generate_experimental_IR_V10 else 'int32'},
        }, attrs)
        assert isinstance(self.attrs['keep_dims'], int) or isinstance(self.attrs['keep_dims'], bool)
        self.attrs['keep_dims'] = bool(self.attrs['keep_dims'])

    def supported_attrs(self):
        return [
            ('keep_dims', lambda node: str(node.keep_dims)),
        ]
```

A class for inherit, so no need to enable.

Op, type and infer function

Needs parameter, and its default value

Number of I/O node

Set INT64 for IR v10

Operation attribute for inference (IE), will save into IR.

Model Optimizer – IR Operation – Cont.

ReduceOps.py dir: ${MO_ROOT}/extensions/ops/ReduceOps.py

- Create **ReduceL2** class which **inherit from ReduceOp** for ReduceL2

```
class ReduceL2(ReduceOp):
    enabled = True
    op = 'ReduceL2'
    op_type = 'ReduceL2'
```


Enable ReduceL2 Op

Using existing layer replace ReduceL2 – Cont.

op_ reducel2.py dir: ${MO_ROOT}/extensions/front/onnx/reducel2.py

- Create reducel2.py for replacement

```
from extensions.ops.ReduceOps import ReduceSum
from mo.front.common.partial_infer.utils import float_array
from mo.graph.graph import Node, Graph
from mo.front.common.replacement import FrontReplacementOp
from mo.front.tf.graph_utils import create_op_node_with_second_input
from extensions.ops.elementwise import Pow

class ReduceL2Replacement(FrontReplacementOp):
    op = 'ReduceL2'
    enabled = True

    def replace_op(self, graph: Graph, node: Node):
        # Add new nodes
        pow_node_1 = create_op_node_with_second_input(graph, Pow, float_array(2), {'name': node.name + '/pow_1_'})
        pow_node_2 = create_op_node_with_second_input(graph, Pow, float_array(0.5), {'name': node.name + '/pow_2_'})
        reduce_sum = ReduceSum(graph, {'name': node.name + '/ReduceSum',
            'keep_dims': node.keep_dims,
            'axis': node.axis
            }).create_node()

        # Connect nodes
        node.in_port(0).get_connection().set_destination(pow_node_1.in_port(0))
        pow_node_1.out_port(0).connect(reduce_sum.in_port(0))
        reduce_sum.out_port(0).connect(pow_node_2.in_port(0))

        return [pow_node_2.id]
```

Will be replace Operation

Replace Operation

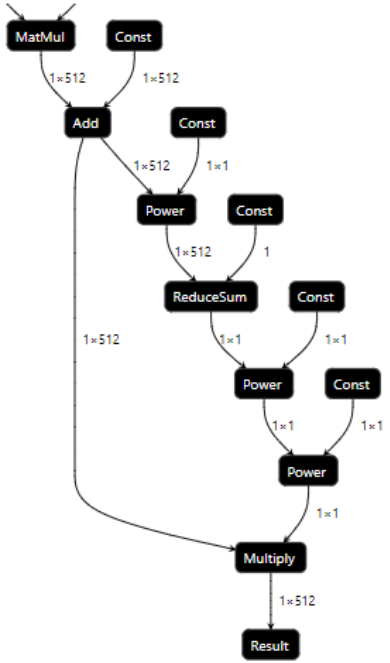
Create nodes

Link nodes

Return last node

Using existing layer replace ReduceL2 – Cont.

- Output IR



NODE PROPERTIES

type	Const
name	752/ReduceSum/Cast_121727_const

ATTRIBUTES

element_type	i64
offset	175159588
shape	1
size	8

OUTPUTS

output	id: **580:1**

Result

```
C:\Users\johnfeng\Documents\Intel\OpenVINO\inference_engine_cpp_samples_build\intel64\Release>benchmark_app.exe -m C:\Users\Public\moment-in-time.xml -d CPU
[Step 1/11] Parsing and validating input arguments
[ INFO ] Parsing input parameters
[ WARNING ] -nstreams default value is determined automatically for a device. Although the automatic selection usually provides a reasonable performance,but


[Step 2/11] Loading Inference Engine
[ INFO ] InferenceEngine:
        API version ............ 2.1
        Build .................. 37988
        Description ....... API
[ INFO ] Device info:
        CPU
        MKLDNNPlugin version ......... 2.1
        Build ........... 37988

[Step 3/11] Setting device configuration
[Step 4/11] Reading the Intermediate Representation network
[ INFO ] Loading network files
[ INFO ] Read network took 447.73 ms
[Step 5/11] Resizing network to match image sizes and given batch
[ INFO ] Network batch size: 1, precision: MIXED
[Step 6/11] Configuring input of the model
[Step 7/11] Loading the model to the device
[ INFO ] Load network took 1967.62 ms
[Step 8/11] Setting optimal runtime parameters
[Step 9/11] Creating infer requests and filling input blobs with images
[ INFO ] Network input 'input.1' precision U8, dimensions (NCHW): 1 3 112 112
[ WARNING ] No input files were given: all inputs will be filled with random values!
[ INFO ] Infer Request 0 filling
[ INFO ] Fill input 'input.1' with random values (image is expected)
[ INFO ] Infer Request 1 filling
[ INFO ] Fill input 'input.1' with random values (image is expected)
[ INFO ] Infer Request 2 filling
[ INFO ] Fill input 'input.1' with random values (image is expected)
[ INFO ] Infer Request 3 filling
[ INFO ] Fill input 'input.1' with random values (image is expected)
[Step 10/11] Measuring performance (Start inference asyncronously, 4 inference requests using 4 streams for CPU, limits: 60000 ms duration)

[Step 11/11] Dumping statistics report
Count:      1144 iterations
Duration:   60369.21 ms
Latency:    206.15 ms
Throughput: 18.95 FPS
```