Model Optimizer

Intel® Deep Learning Deployment Toolkit

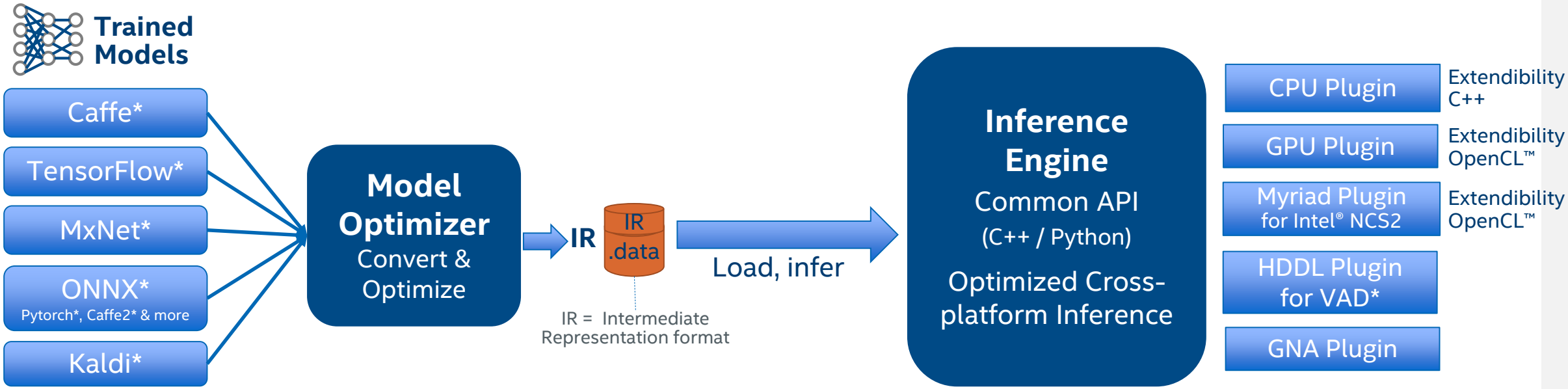
For Deep Learning Inference

Model Optimizer

- A Python* based tool to **import** trained models and **convert** them to Intermediate Representation
- **Optimizes for performance** or space with conservative topology transformations
- **Hardware-agnostic** optimizations

Inference Engine

- High-level, C/C++ and Python, inference **runtime API**
- Interface is implemented as **dynamically loaded plugins** for each hardware type
- Delivers advanced performance for each type **without requiring users to implement and maintain multiple code pathways**



GPU = Intel® CPU with integrated GPU/Intel® Processor Graphics, Intel® NCS = Intel® Neural Compute Stick (VPU)
*VAD = Intel® Vision Accelerator Design Products (HDDL-R)

Model Optimizer: **Generic Optimization**

- Model optimizer performs generic optimization
  - Node merging
  - Horizontal fusion
  - Batch normalization to scale shift
  - Fold scale shift with convolution
  - Drop unused layers (dropout)

The simplest way to convert a model is to run mo.py with a path to the input model file

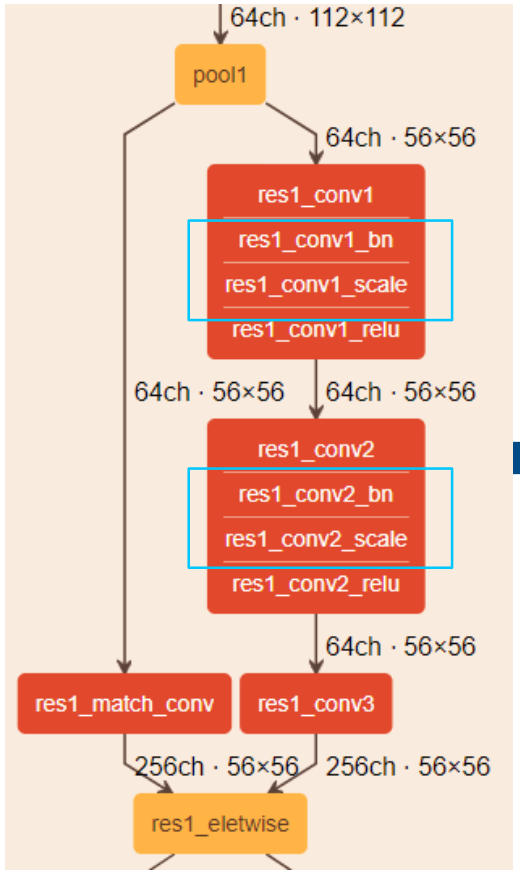
- By default, generic optimization will be automatically applied, unless manually set disable

```
python3 /opt/intel/openvino/deployment_tools/model_optimizer/mo.py \
    --input_model models/public/resnet-50/resnet-50.caffemodel \
```

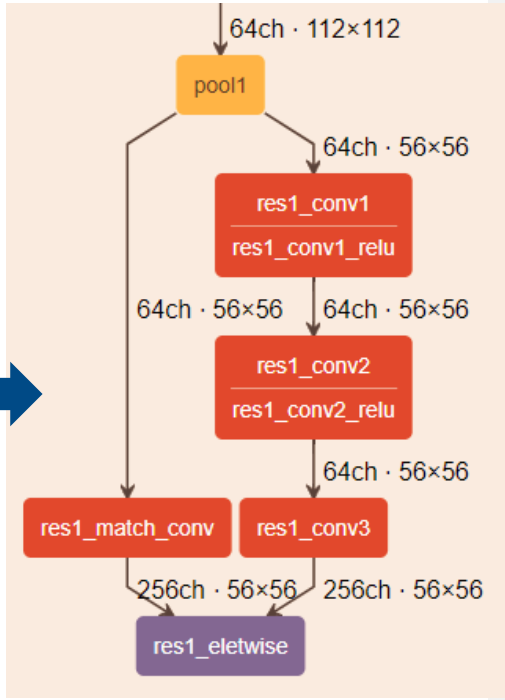
Model Optimization Techniques

- Linear Operation Fusing: 3 stages

1. **BatchNorm and ScaleShift decomposition:** *BN* layers decomposes to *Mul->Add->Mul->Add* sequence; ScaleShift layers decomposes to *Mul->Add* sequence.
2. **Linear operations merge:** Merges sequences of Mul and Add operations to the **single** Mul->Add instance.
3. **Linear operations fusion:** Fuses Mul and Add operations to Convolution or FullyConnected layers.



Caffe* Resnet269 block (from Netscope)

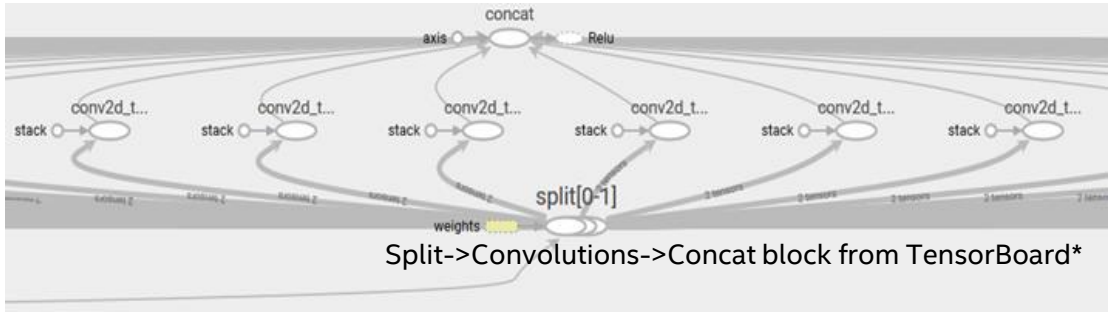


Merged Caffe* Resnet269 block (from Netscope*)

Model Optimizer: Framework or topology specific optimization

Grouped Convolutions Fusing

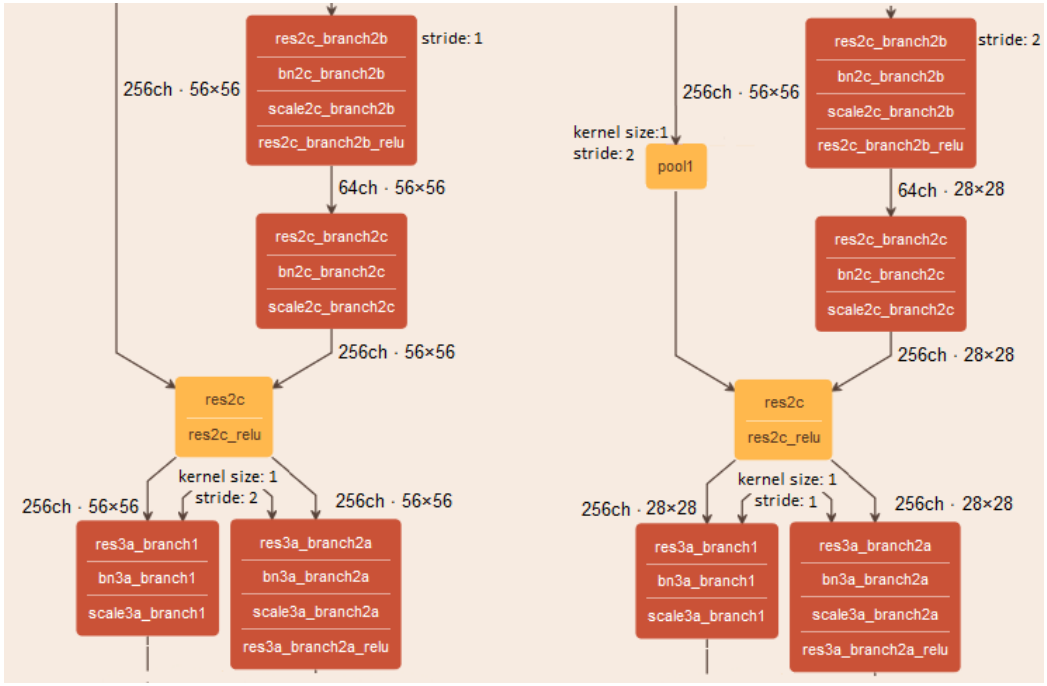
- Grouped convolution fusing is a specific optimization that applies for TensorFlow* topologies. The main idea of this optimization is to combine convolutions results for the Split outputs and then recombine them using **Concat** operation in the same order as they were out from **Split**.



Split->Convolutions->Concat block from TensorBoard*

ResNet* optimization (stride optimization)

- This optimization is to move the stride that is greater than 1 from Convolution layers with the kernel size = 1 to upper Convolution layers. In addition, the Model Optimizer adds a Pooling layer to align the input shape for a Eltwise layer, if it was changed during the optimization.



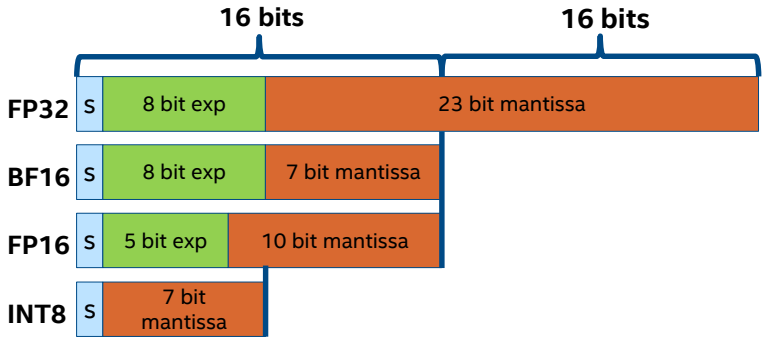
Model Optimizer: Quantization

--data_type {FP16,FP32,half,float}

- Data type for all intermediate tensors and weights.
- If original model is in FP32 and --data_type=FP16 is specified, all model weights and biases are quantized to FP16.

```
python3 /opt/intel/openvino/deployment_tools/model_optimizer/mo.py \
      --input_model models/public/resnet-50/resnet-50.caffemodel \
      --data_type FP16 \
      --model_name resnet-50-fp16 \
      --output_dir irfiles/
```

PLUGIN	FP32	FP16	INT8
CPU plugin	Supported and preferred	Supported	Supported
GPU plugin	Supported	Supported and preferred	Supported*
VPU plugins	Not supported	Supported	Not supported
GNA plugin	Supported	Supported	Not supported
FPGA plugin	Supported	Supported	Not supported



Note:
1. To create INT8 models, you will need DL Workbench or Post Training Optimization Tool
2. FPGA also support FP11, convert happens on FPGA

Model Optimizer: Other Common Parameters

- --scale, --scale_values, --mean_values, --mean_file
  - Usually neural network models are trained with the normalized input data. This means that the input data values are converted to be in a specific range, for example, [0, 1] or [-1, 1]. Sometimes the mean values (mean images) are subtracted from the input data values as part of the pre-processing
- --input_shape
  - when the input data shape for the model is not fixed, like for the fully-convolutional neural networks. In this case, for example, TensorFlow* models contain -1 values in the shape attribute of the Placeholder operation. Inference Engine does not support input layers with undefined size, so if the input shapes are not defined in the model, the Model Optimizer fails to convert the model.
- --reverse_input_channels
  - Inference Engine samples load input images in the BGR channels order. However, the model may be trained on images loaded with the opposite order