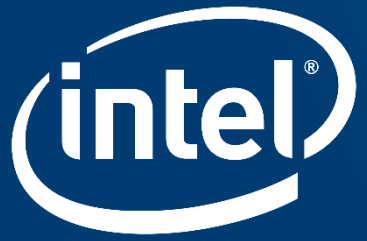


NEURAL NETWORKS OPTIMIZATIONS

IOTG/VMC/Vision Products/ICV

Novak Alexander

Agenda

- What is network optimization ?
- Tradition execution path
- Graph representations
- General optimizations
- Network compilers
- Results

WHAT IS NETWORK OPTIMIZATION ?

Problem statement

Neural Network model

+

General limitations

- Performance
- Accuracy
- Size of network
- Number and type of layers

+

HW limitations

- Memory
- Computing power
- Available layers
- HW accelerators and special units

=

Optimized Neural Network model

TRADITION EXECUTION PATH

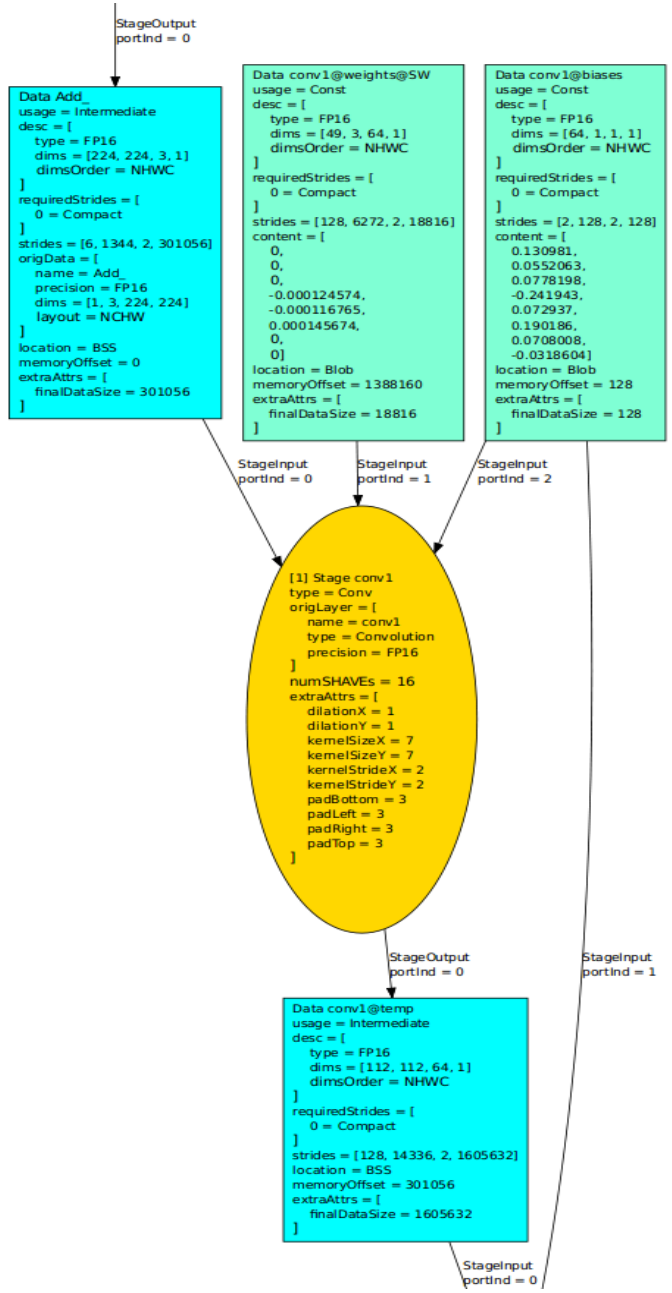
Tradition execution path:

- Graph representation
  - One of the popular CNN framework :
    TensorFlow, Keras, PyTorch, Caffe, MXNet, Caffe2, OpenVINO, etc
- Optimized backend
  - HW specific optimized libraries
    CuBlas, MKL-DNN, CLDNN
  - Full graph: transformations implemented in framework.
    Layout transform elimination, layer fusion, memory management
- New platform enablement -> Integration of layer library and framework tuning

GRAPH REPRESENTATIONS

Graph representations: General

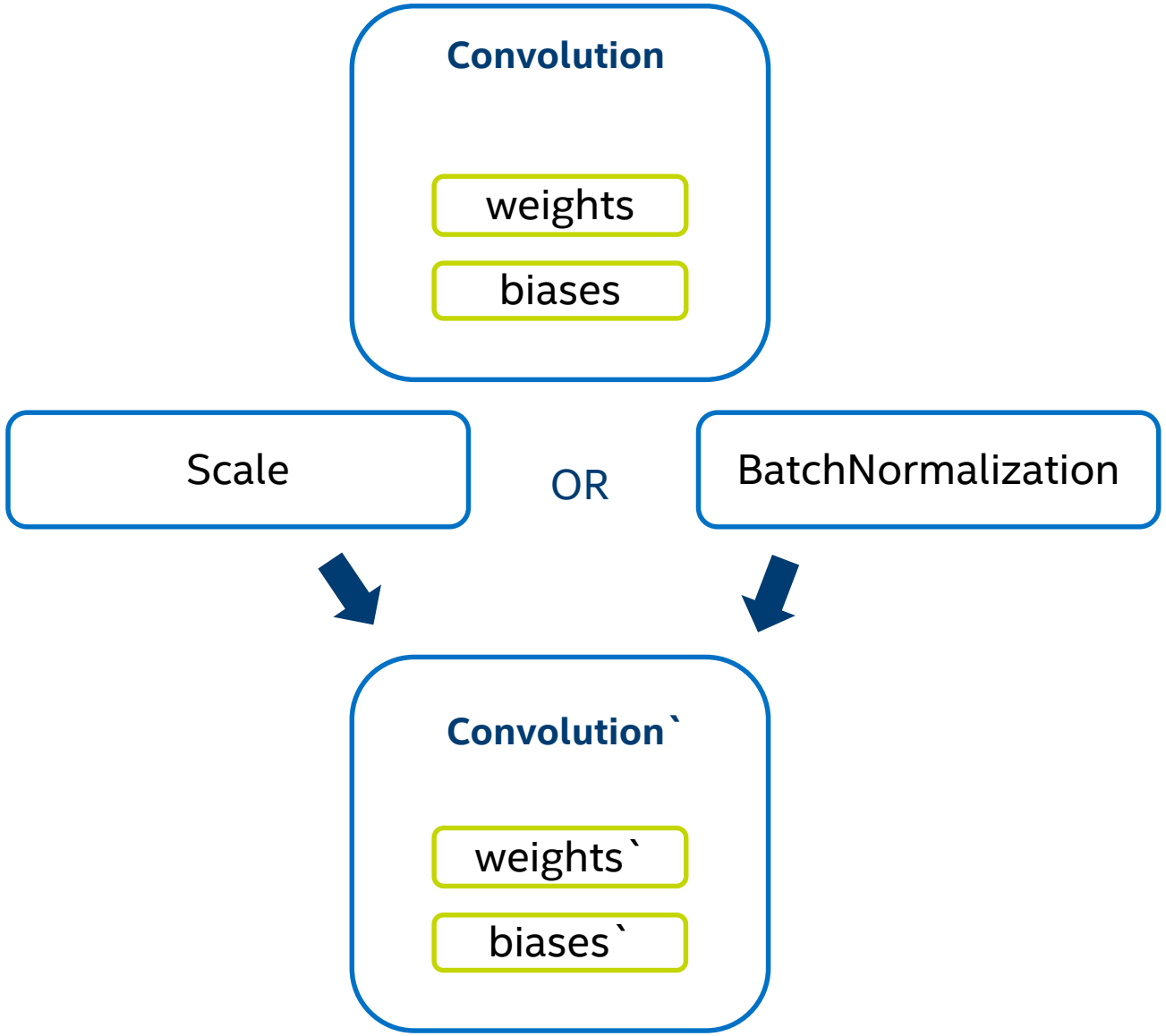
- Include different nodes and edges
- We can build some additional graphs (interference graph, tiles graph etc)
- Optimizations on graph



GENERAL OPTIMIZATIONS ON GRAPH

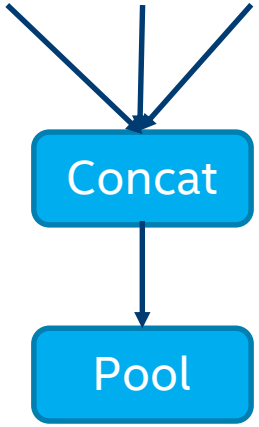
General optimizations on graph: Merge layers

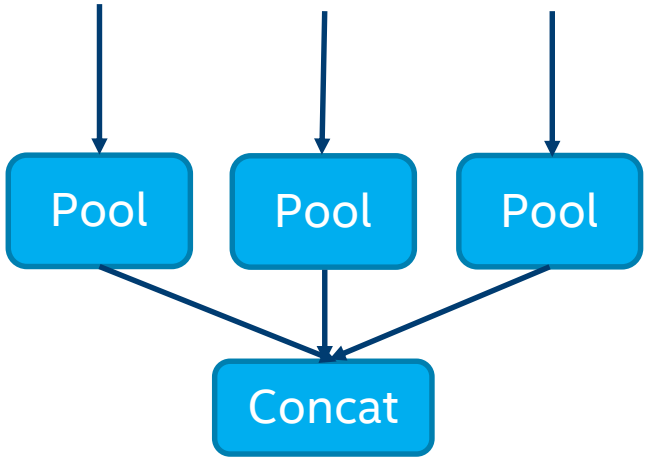
- Convolution + Scale
- Convolution + BatchNormalization
- Convolution + ReLu
- Normalization



General optimizations on graph: Replace and Change order

- Reshape
- Permutation
- Concat + Smth
- Pool and Conv





HW SPECIFIC OPTIMIZATIONS

HW specific optimizations: General

- Some operations works faster than others
- Convolution
- Merged operations and try express smth to HW op

For example:

HW specific optimizations: Memory optimizations

- Devices has different types of memory (cache, fast memory etc)
  Memory is very limited resource
- Optimal allocation is hard task
  - Tiling
  - Special allocators
  - Interference graph
  - Inplace operations

RESULTS

Networks performance (FPS)

Network	Without optimizations	Graph optimizations	HW optimizations	HW + graph optimizations
GoogleNet V1	12.01	14.48	84.11	89.08
GoogleNet V2	10.1	11.5	79.40	84.65
Yolo Tiny V1	15.34	17.73	51.01	68.35
Yolo Tiny V2	9.29	10.42	41.40	49.83
SqueezeNet 1.1	47.32	51.7	248	272.10
VGG 16	2.01	2.26	7.9	10.2
ResNet 18	13.5	16.3	93.09	93.99
ResNet 50	6.5	8.6	30.76	31.93
MobileNet SSD	20.3	20.29	50.07	56.99

FUTURE NETWORK COMPILERS

Future Network compilers: Old way problems

- Graph representation is too high level
  - It knows nothing about the backend
  - optimizations are limited
- Topologies enablement and layer fusion is limited by vendor specific library
- Problems with custom layers integration

Future Network compilers

- Compiler based approach removes meaning of the layer
- Compiler defines tensor as a native data type
- Compiler knows HW backend specifics
  - optimization choices are based on the cost model
- Assumption: convolution defines speed of the topology
  – the rest is memory bound


intel
experience
what's inside™

Links and Materials

- TVM (https://github.com/dmlc/tvm)
- PlaidML (https://github.com/plaidml/plaidml)
- DL coursera (https://www.coursera.org/learn/machine-learning)
- Yandex (https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie)