PULP PLATFORM
Open Source Hardware, the way it should be!

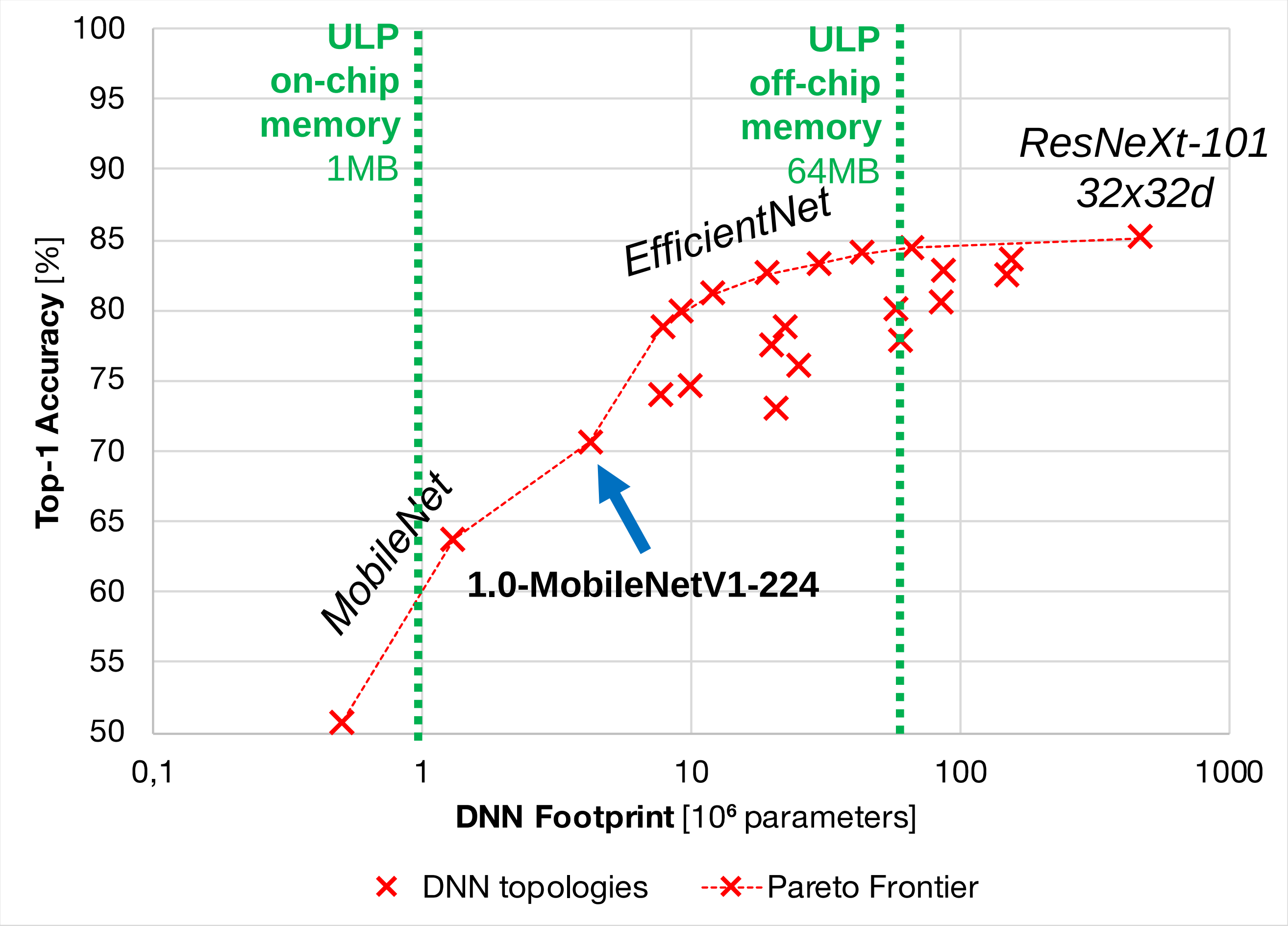
Deployment of DNN on Extreme Edge Devices (1)

Alessio Burrello <alessio.burrello@unibo.it>
Francesco Conti <f.conti@unibo.it>

Bringing DNN Inference to the Edge

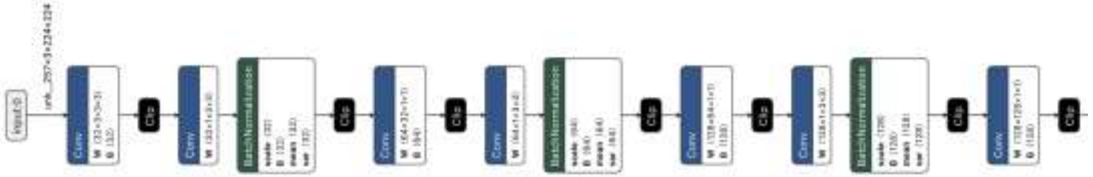
ImageNet Top-1 Accuracy
vs **Memory Footprint**

- Most entries > 10 MB
- Pareto Frontier Acc vs Memory (from 50% @ 0.5Mparam to 85% @ 445 Mparam)
- Almost always require off-chip DRAM even for **ULP**!

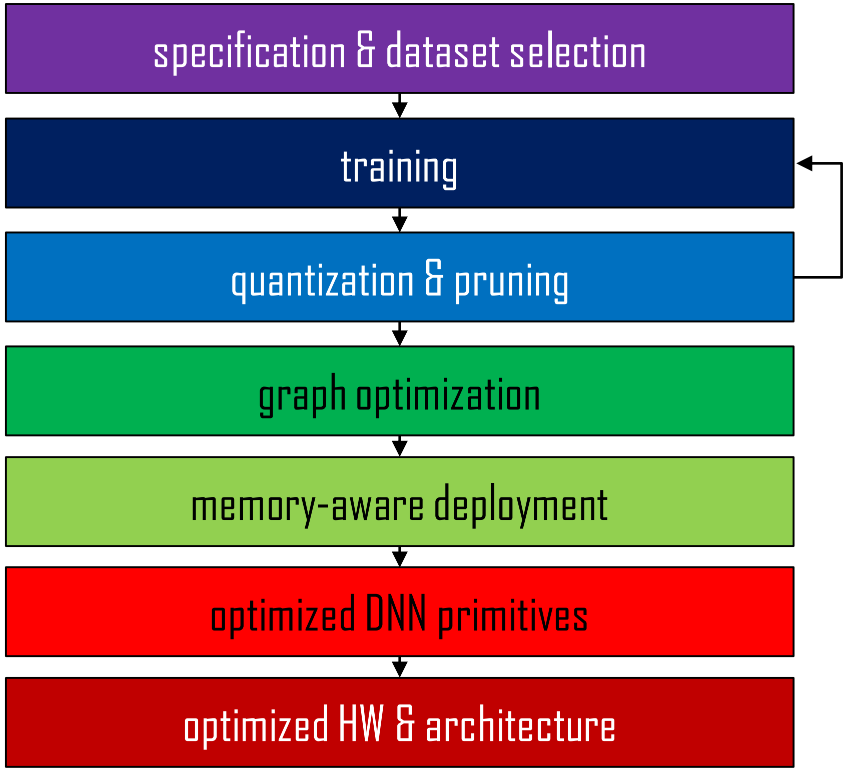


1.0-MobileNetV1-224

Unibo Flow



- **Actually enabling execution of real-world sized DNNs at extreme edge is still a challenge**
  - most state-of-the-art (e.g. CMSIS-NN) shown on very small DNNs & datasets, e.g. CIFAR10
  - challenge #1: small and manually managed on-chip memory (512 kB L2, 64 kB fast L1 on most PULP-based chips)
  - challenge #2: better support for efficient integer computation, not floating point
- **We show the Unibo Flow, a vertically integrated framework for deployment of DNNs on PULP-based extreme edge platforms**
  - from algorithm definition (PyTorch) to running the DNN on the embedded platform (e.g., on GreenWaves GAP8, Mr. Wolf, PULP simulators)



specification & dataset selection

training

quantization & pruning

graph optimization

memory-aware deployment

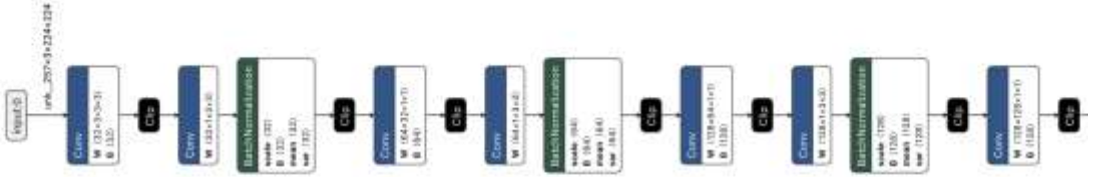
optimized DNN primitives

optimized HW & architecture

Outline

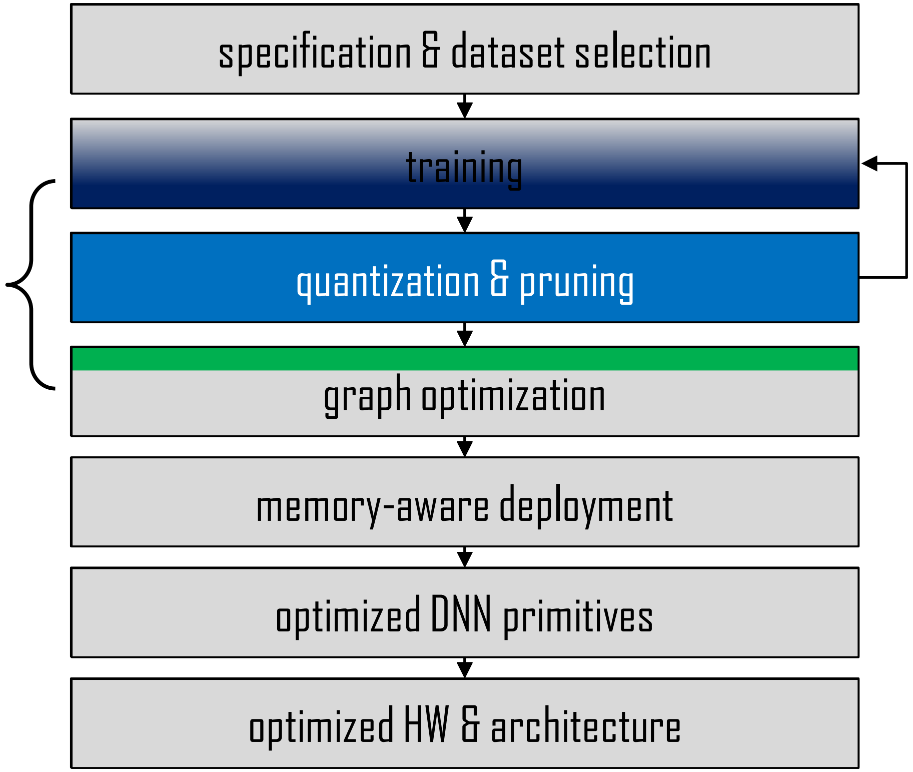
1. **Intro on the UNIBO Flow**
2. **NEMO (*NE*ural *M*inimization for pyt*O*rch)**
   1. Topological Contraints
3. **DORY (*D*eployment *O*riented to memo*RY*)**
   1. Graph and Node reading
   2. Tiling
      - L3-L2 movement
      - L2-L1 movement
      - Data movement
   3. Template writing
4. **PULP-NN**
   1. Optimized backend
   2. Supported Layers
5. **How to Generate a Network**
6. **Examples**

Unibo Flow

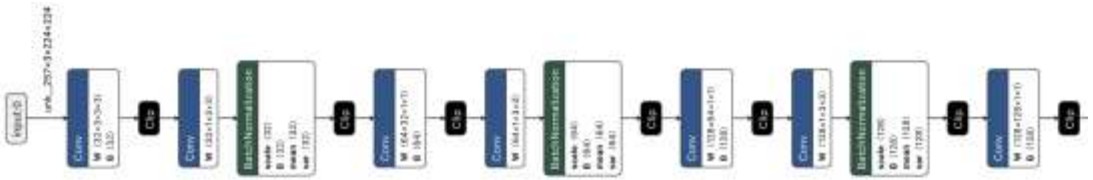


NEMO

***NE**ural **M**inimization for pyt**O**rch*

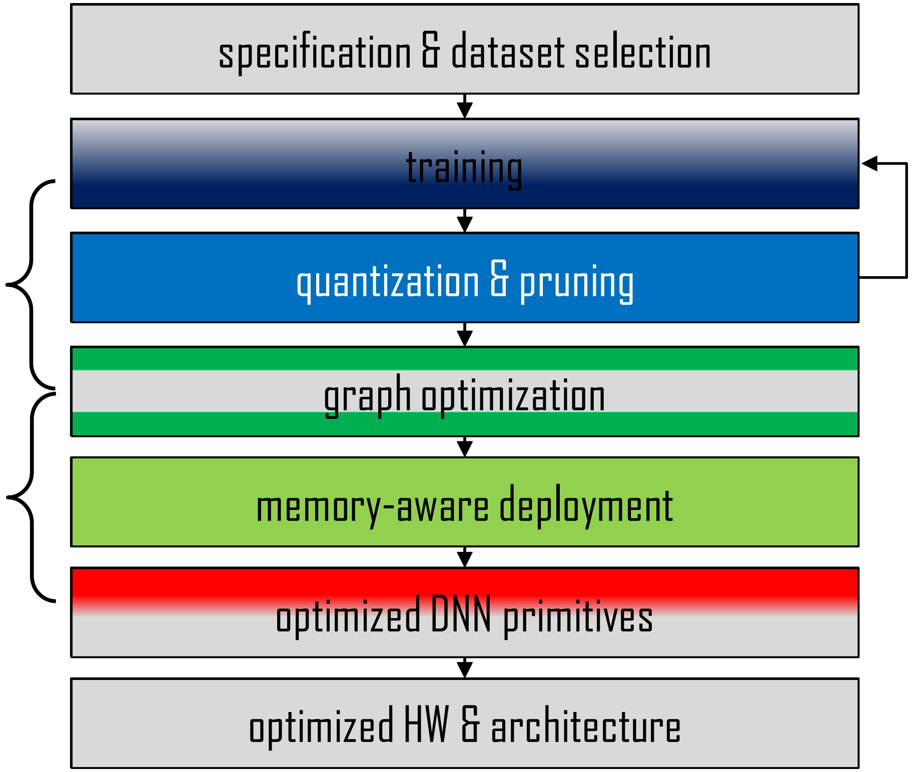


Unibo Flow



NEMO
***NE**ural **M**inimization for pyt**O**rch*

DORY
D**eployment **O**riented to memo**RY



specification & dataset selection

training

quantization & pruning

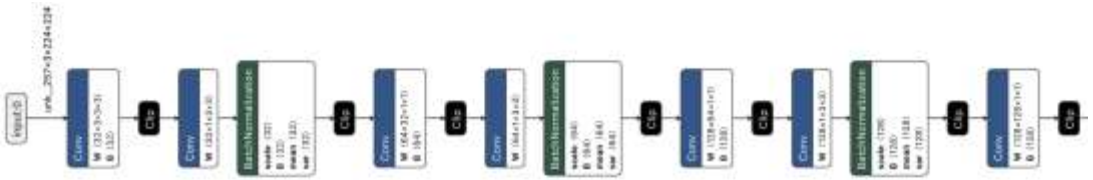
graph optimization

memory-aware deployment

optimized DNN primitives

optimized HW & architecture

Unibo Flow



NEMO

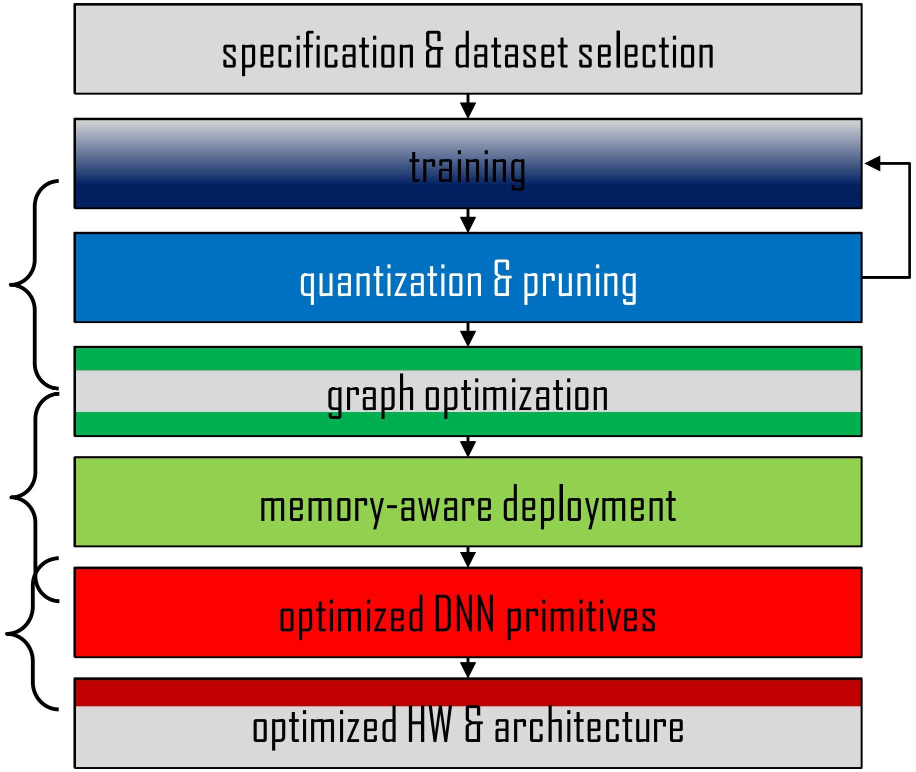
***NE**ural **M**inimization for pyt**O**rch*

DORY

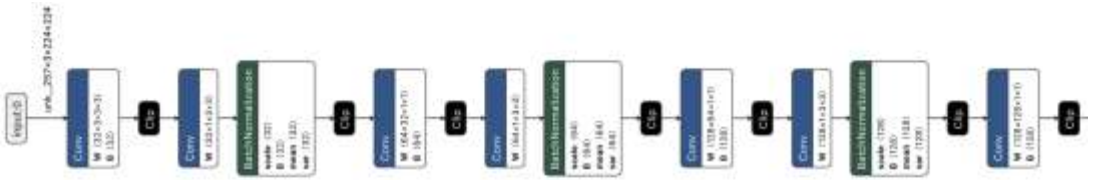
D**eployment **O**riented to memo**RY

PULP-NN

***PULP N**eural **N**etwork backend*



Contributors



NEMO
***NE**ural **M**inimization for pyt**O**rch*

Francesco Conti
Marcello Zanghieri
Leonardo Ravaglia
Lorenzo Lamberti

DORY
D**eployment **O**riented to memo**RY

Alessio Burrello
Francesco Conti
Thorir Ingolfsson

PULP-NN
***PULP N**eural **N**etwork backend*

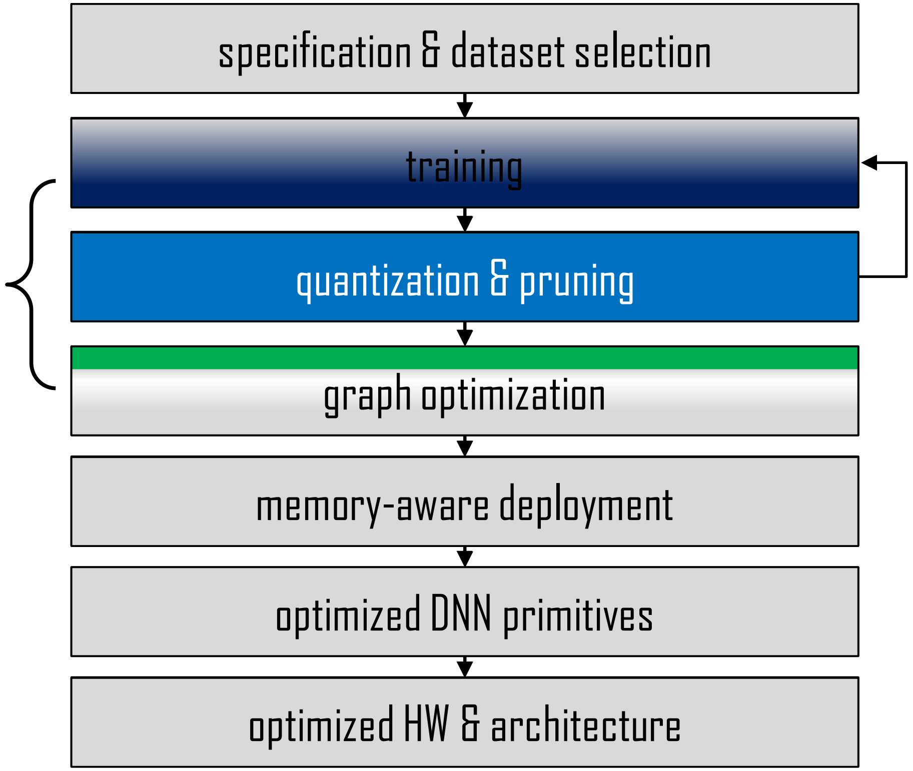
Angelo Garofalo
Nazareno Bruschi

NEMO: fp32 to full-integer networks

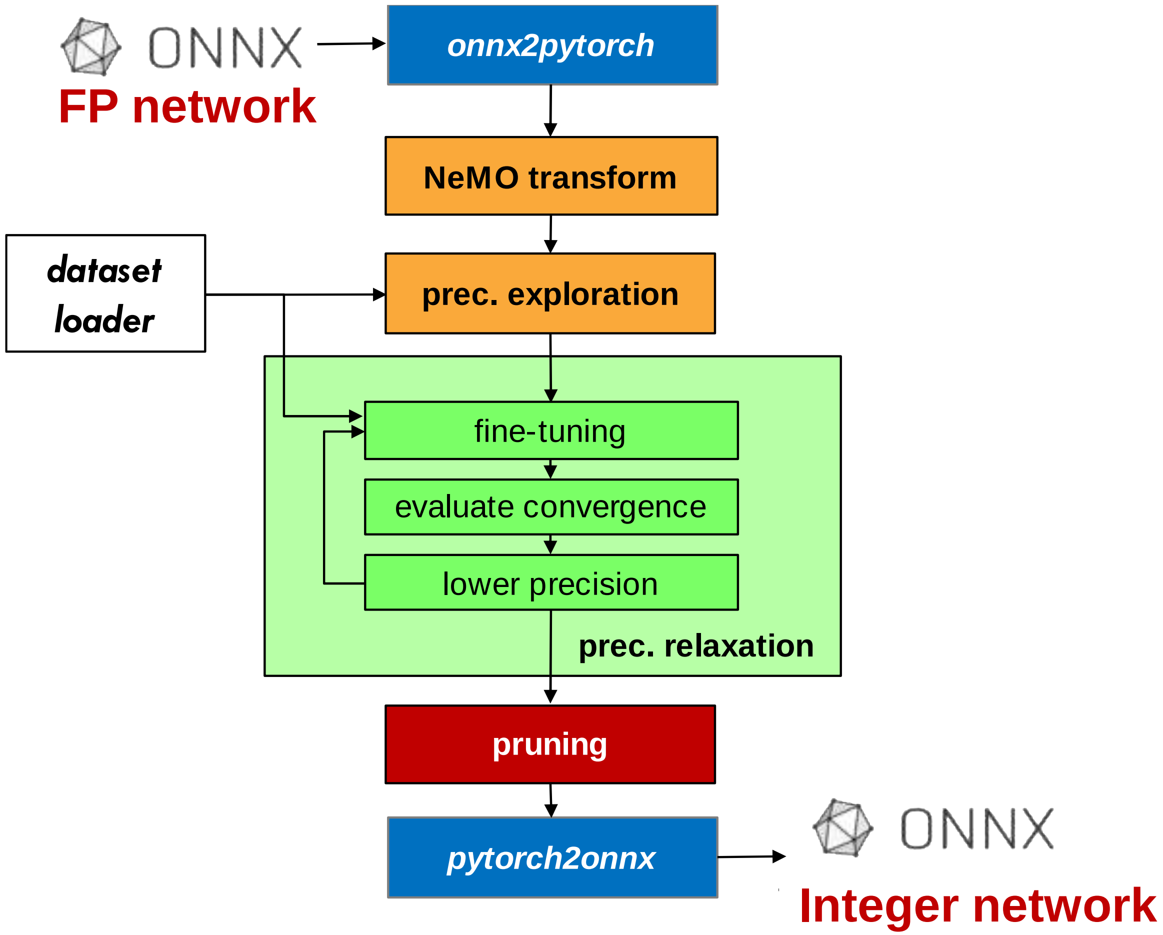
NEMO

***NE**ural **M**inimization for pyt**O**rch*

From a full-precision representation to a fully integer (**<u>not fixed-point</u>**) HW-deployable one



NEMO: quantization-aware retraining



ONNX

FP network

onnx2pytorch

NeMO transform

dataset loader

prec. exploration

fine-tuning

evaluate convergence

lower precision

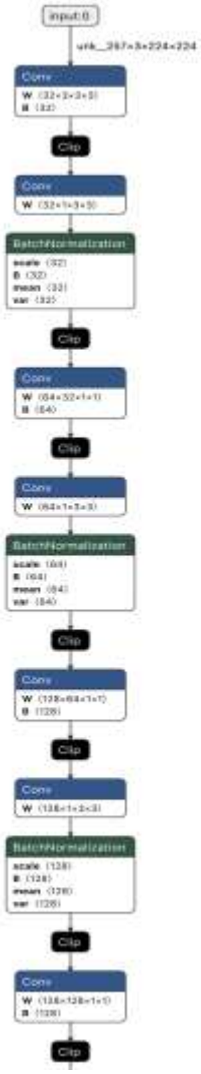
prec. relaxation

pruning

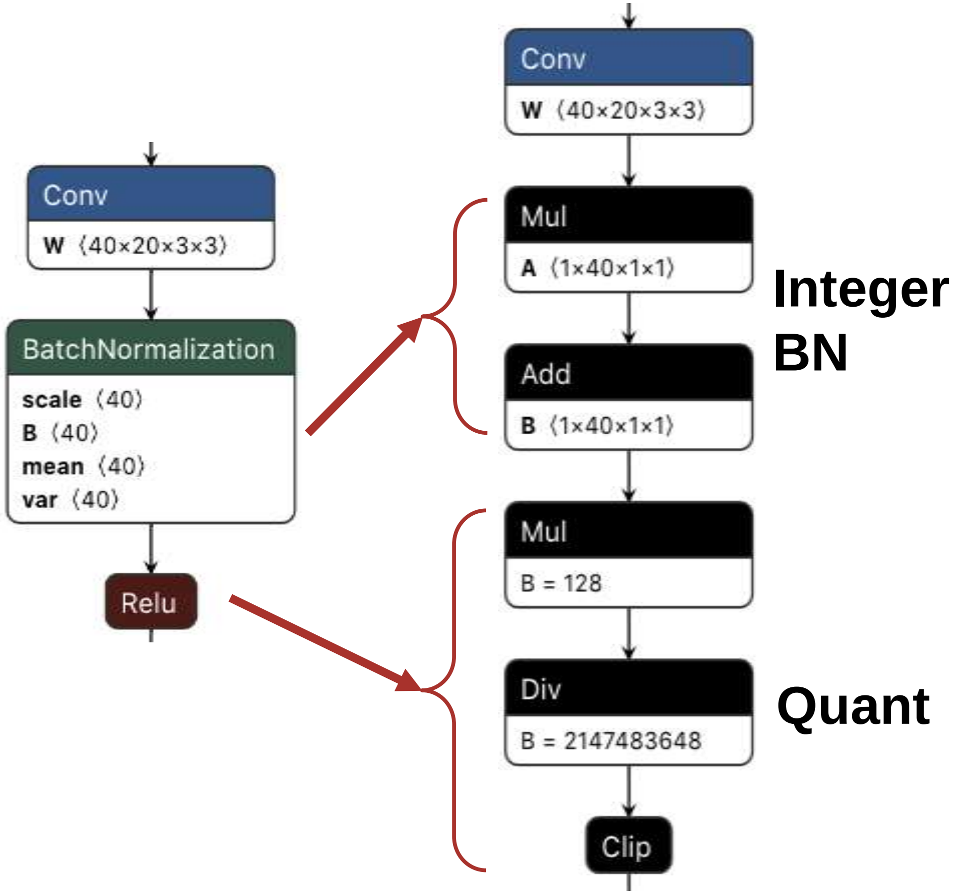
pytorch2onnx

ONNX

Integer network



NEMO: topological constraints



1. Recognize *super-layers* in the network
   - typically, Conv+BN+Clip (quantization is implicit in QF format)
2. Represent all tensors in the quantized form

$$\underbrace{T_{int}}_{\text{integer tensor (integer image)}} \cdot \underbrace{\varepsilon_T}_{\text{real-valued scalar (quantum)}}$$

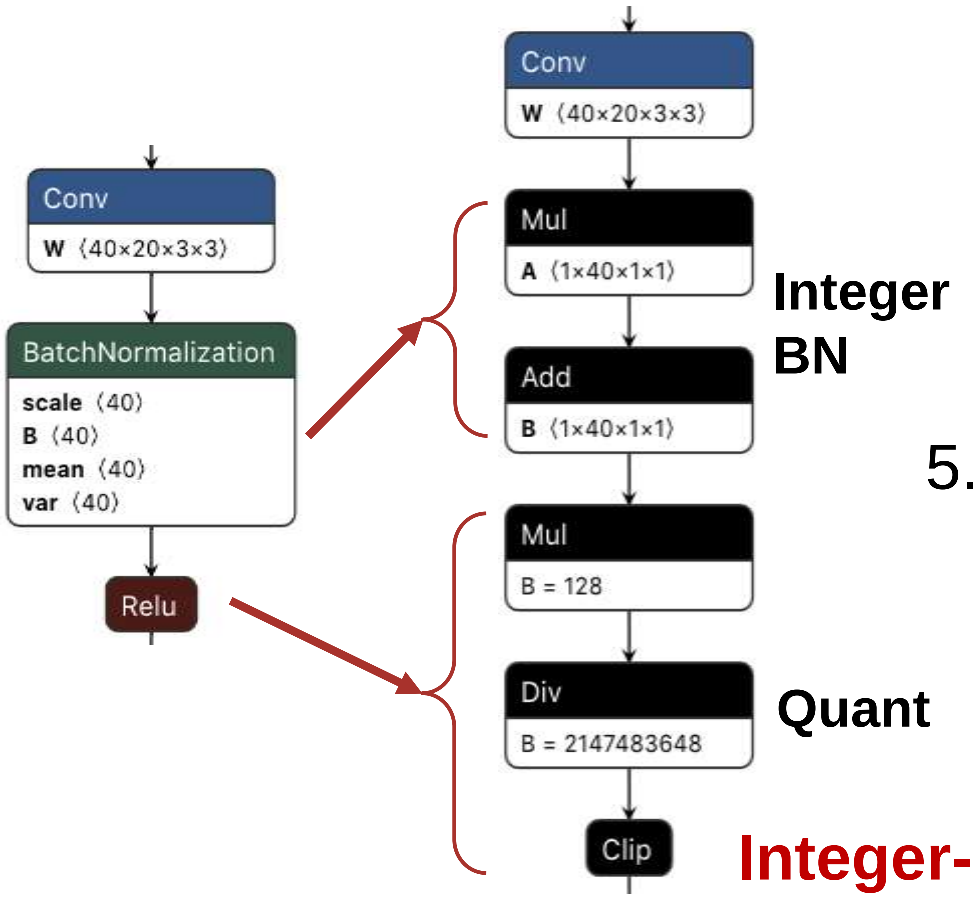
$$T = T_{int} \cdot \varepsilon_T$$

integer tensor (**integer image**)

real-valued scalar (**quantum**)

3. Replace BN and Clip/Quant operations with equivalent working on quantized form and producing quantized tensors

NEMO: topological constraints

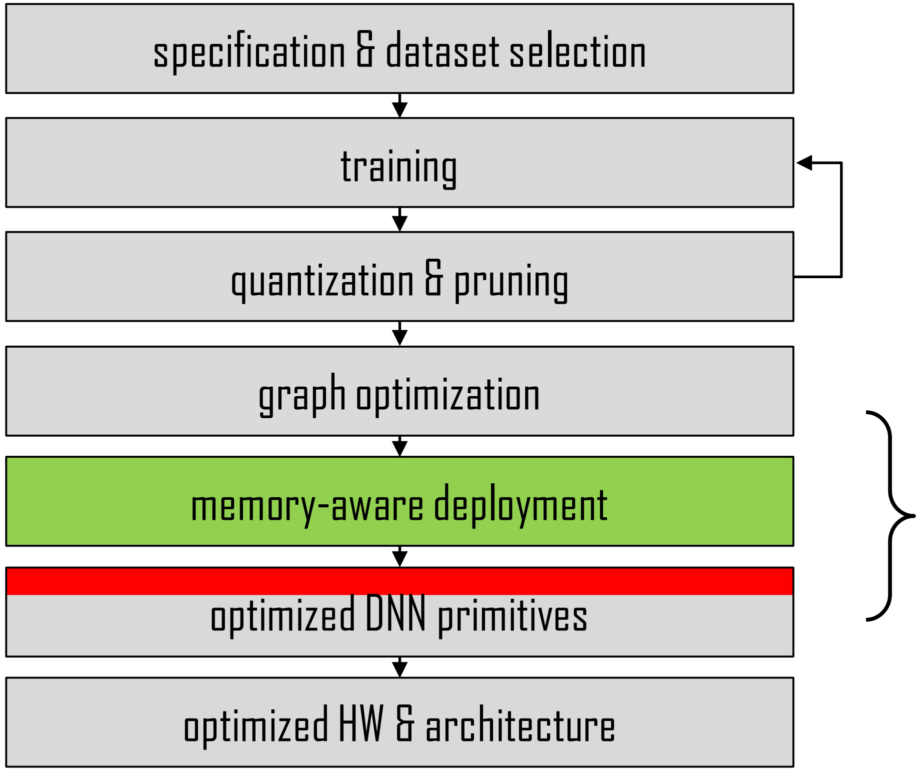


4. Keep track of ε_T quanta along the network
   - linear operations produce outputs with smaller quantum (more bits)
   - non-linear activation produced outputs with quantum "collapsed" to a new value (usually requiring less bits) with **requantization**
5. Replace all tensors by their integer image

$$T \rightarrow T_{int}$$

Integer-Deployable Network

DORY: Tiling & Code Generation



DORY

D**eployment **O**riented to memo**RY

From an int8 quantized onnx network to a C compilable and runnable network

DORY: Tiling & Code Generation

DORY

D**eployment **O**riented to memo**RY

1. Reading of the ONNX output
   1. Recognize backend implemented nodes
   2. Reconstruct the graph with backend nodes input-output dimensions
2. Layer-by-Layer tiling
   1. L3-L2 tiling
   2. L2-L1 tiling
   3. Memory allocation in L2
3. Layer template compilation
4. Network compilation

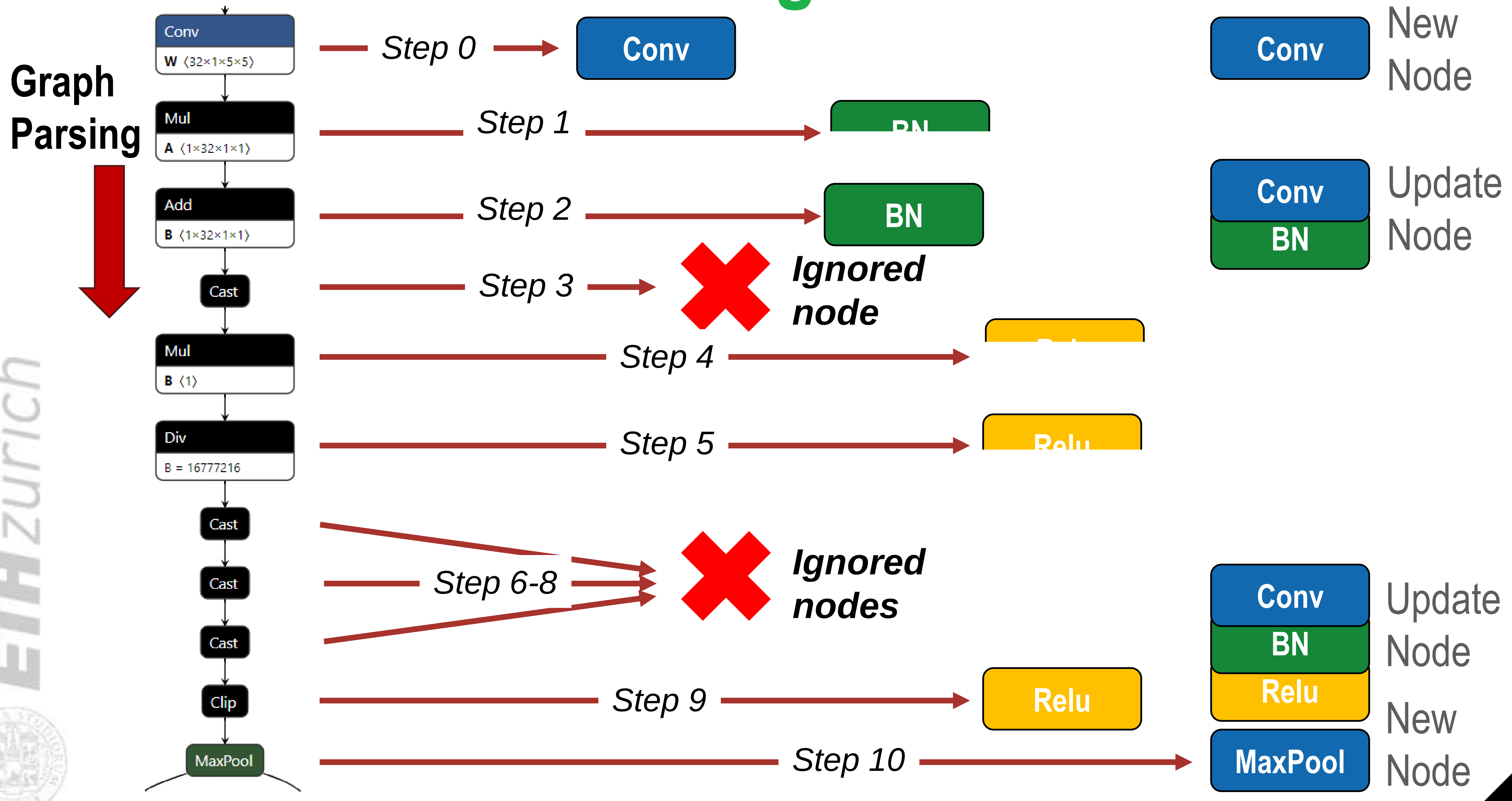
DORY: Tiling & Code Generation

DORY

D**eployment **O**riented to memo**RY

1. **Reading of the ONNX output**
    1. **Recognize backend implemented nodes**
    2. **Reconstruct the graph with backend nodes input-output dimensions**
2. Layer-by-Layer tiling
    1. L3-L2 tiling
    2. L2-L1 tiling
    3. Memory allocation in L2
3. Layer template compilation
4. Network compilation

DORY: ONNX Decoding



Graph Parsing

Conv
W ⟨32×1×5×5⟩
Mul
A ⟨1×32×1×1⟩
Add
B ⟨1×32×1×1⟩
Cast
Mul
B ⟨1⟩
Div
B = 16777216
Cast
Cast
Cast
Clip
MaxPool

Step 0 → Conv

Step 1 → BN

Step 2 → BN

Step 3 → Ignored node

Step 4 →

Step 5 → Relu

Step 6-8 → Ignored nodes

Step 9 → Relu

Step 10 → MaxPool

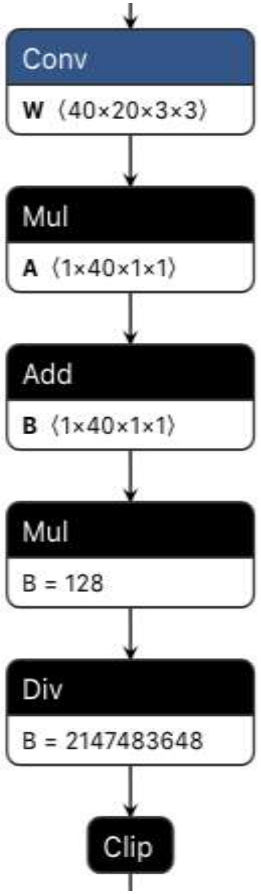
Conv — New Node

Conv BN — Update Node

Conv BN Relu — Update Node

MaxPool — New Node

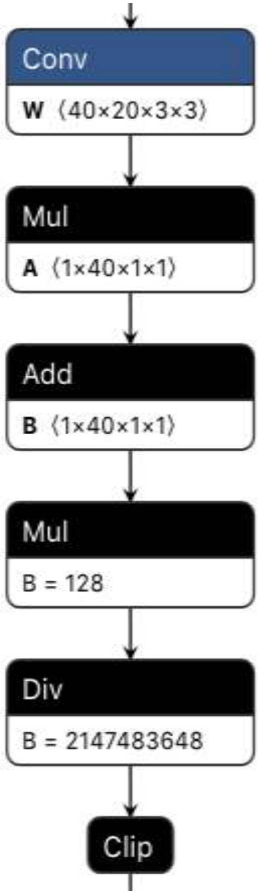
DORY: ONNX Decoding



ONNX READER →

New node_iterating:
ConvBNRelu → ***Layer name***
Filter Dimension
Stride
Padding
Groups
MACs
In-Out dimensions
k: present
lambd: present
outmul: present
outshift: present
Input branch: No
Output branch: No
Input: 93
Output: 105

DORY: ONNX Decoding



ONNX READER →

New node_iterating:

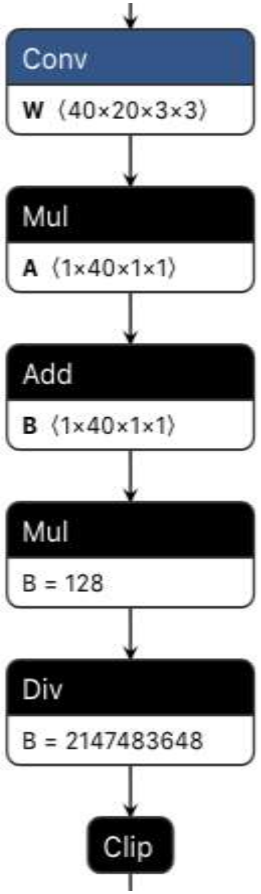
ConvBNRelu → ***Layer name***

Filter Dimension
Stride
Padding
Groups
MACs
In-Out dimensions

Conv/Linear Parameters

k: present
lambd: present
outmul: present
outshift: present
Input branch: No
Output branch: No
Input: 93
Output: 105

DORY: ONNX Decoding



ONNX READER →

New node_iterating:

ConvBNRelu → ***Layer name***

Filter Dimension
Stride
Padding
Groups
MACs
In-Out dimensions

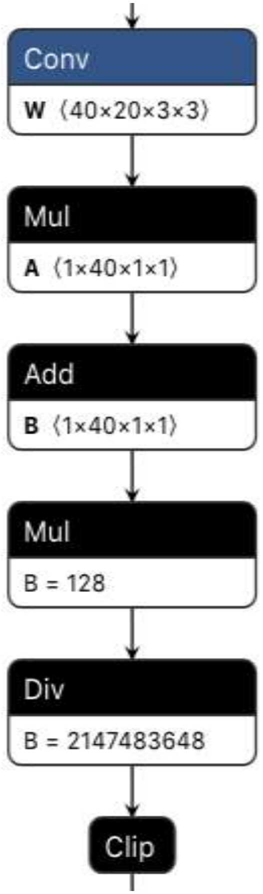
Conv/Linear Parameters

k: present
lambd: present

Batchnorm: in x k + λ

outmul: present
outshift: present
Input branch: No
Output branch: No
Input: 93
Output: 105

DORY: ONNX Decoding



ONNX READER →

New node_iterating:

ConvBNRelu → ***Layer name***

Filter Dimension
Stride
Padding
Groups
MACs
In-Out dimensions

Conv/Linear Parameters

k: present
lambd: present

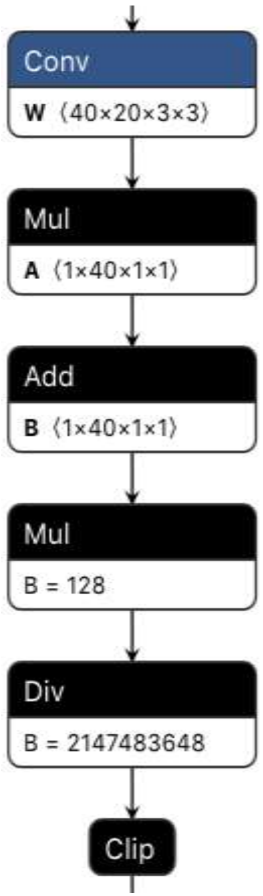
Batchnorm: in x k + λ

outmul: present
outshift: present

Relu: clip8(in x mul >> shift)

Input branch: No
Output branch: No
Input: 93
Output: 105

DORY: ONNX Decoding



ONNX READER

New node_iterating:

ConvBNRelu → ***Layer name***

Filter Dimension
Stride
Padding
Groups
MACs
In-Out dimensions

Conv/Linear Parameters

k: present
lambd: present

Batchnorm: in x k + λ

outmul: present
outshift: present

Relu: clip8(in x mul >> shift)

Input branch: No
Output branch: No
Input: 93
Output: 105

Network topology parameters

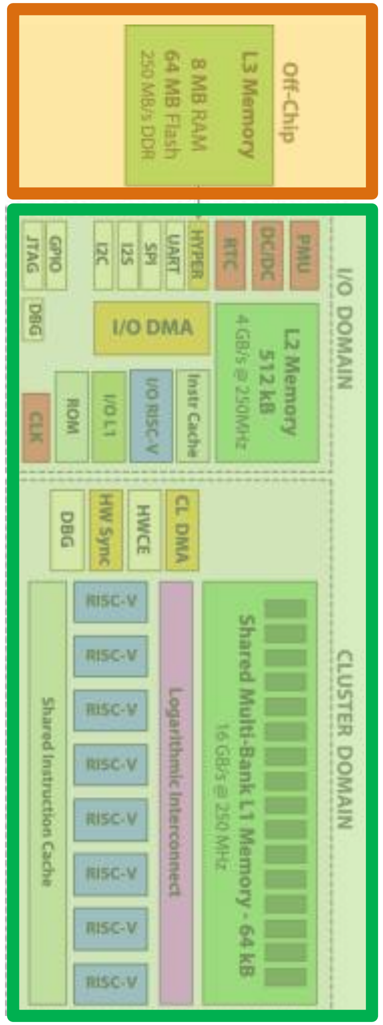
DORY: Tiling & Code Generation

DORY

__D__eployment __O__riented to memo__RY__

1. Reading of the ONNX output
   1. Recognize backend implemented nodes
   2. Reconstruct the graph with backend nodes input-output dimensions
2. **Layer-by-Layer tiling**
   1. **L3-L2 tiling**
   2. **L2-L1 tiling**
   3. **Memory allocation in L2**
3. Layer template compilation
4. Network compilation

DORY: Tiler

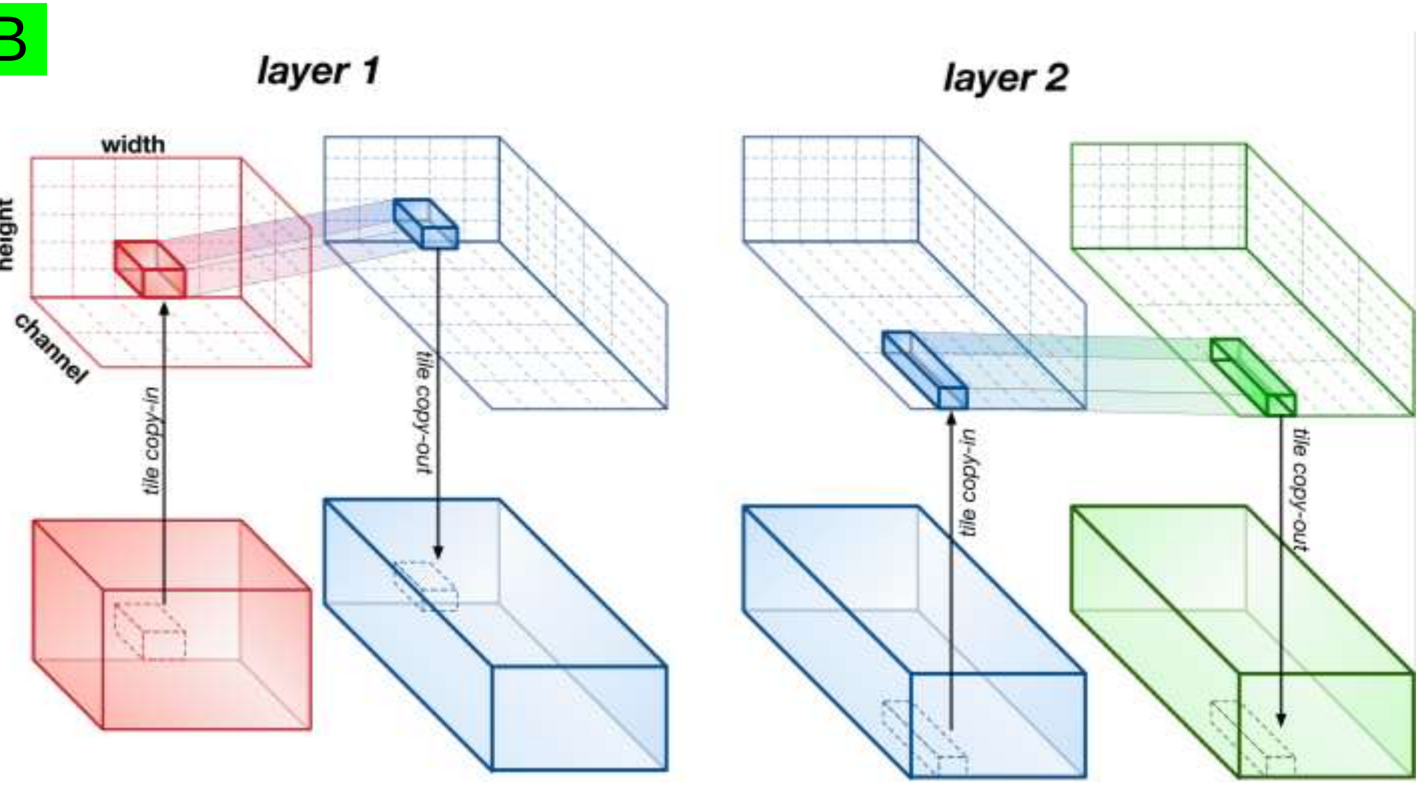


L3 / L2 tiling

64 MB / 512 kB

small memory

big memory



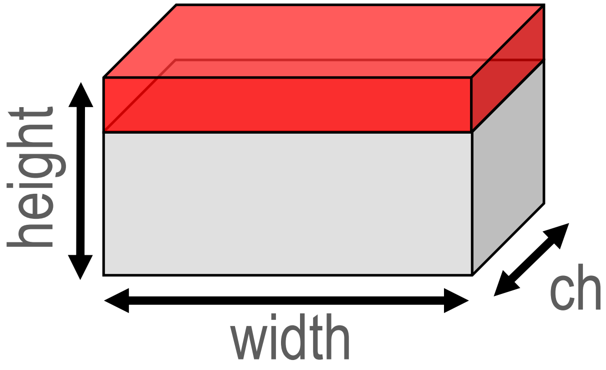
DORY: Tiler – L3/L2

L3/L2 Tiling:

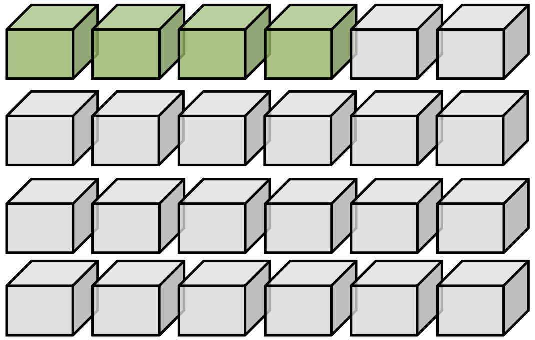
- Large L3 Memory → Enable Big Networks 🙂
- Small Memory Bandwidth → Slow Down Execution 🙁

L3/L2 Tiling *steps*:

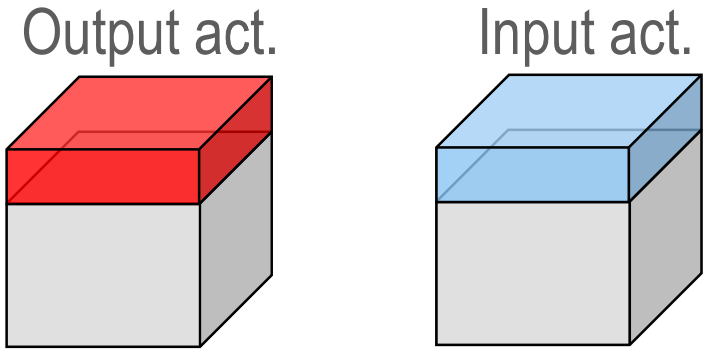
1. Input tiling 🙂



2. Weights tiling 🙂

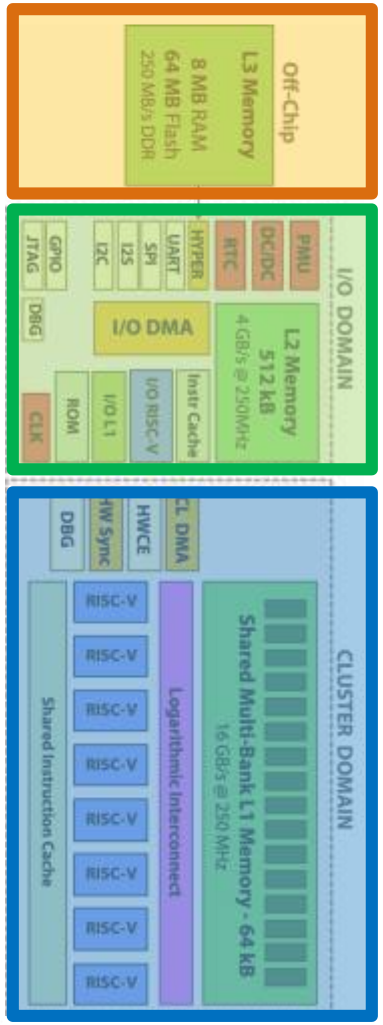


3. Output tiling 🙁



All tiles from L3 to L2 are 1D. Only uDMA linear transfers are required.

DORY: Tiler – L2/L1



L3 / L2 tiling

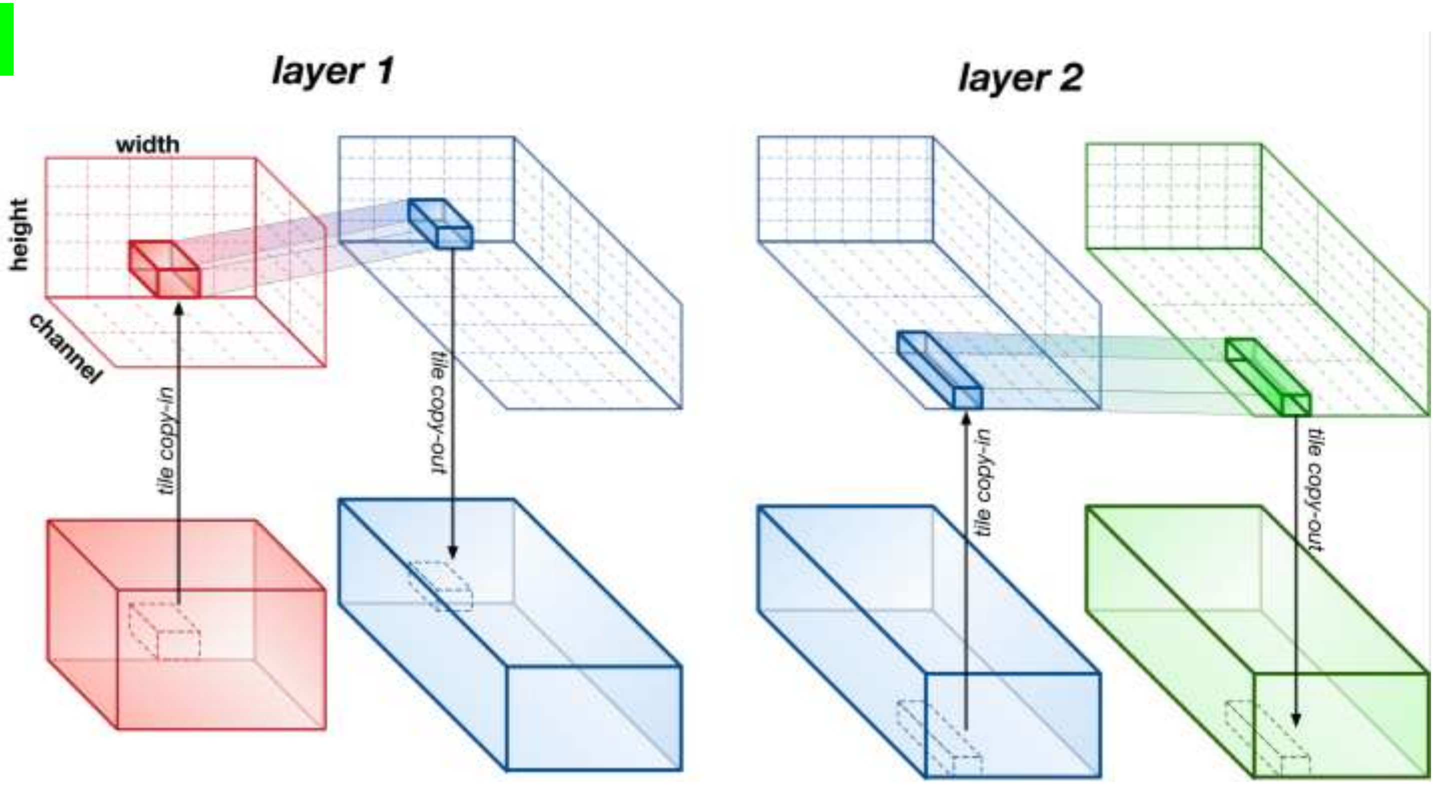
64 MB / 512 kB

small memory

L2 / L1 tiling

512 kB / 64 kB

big memory

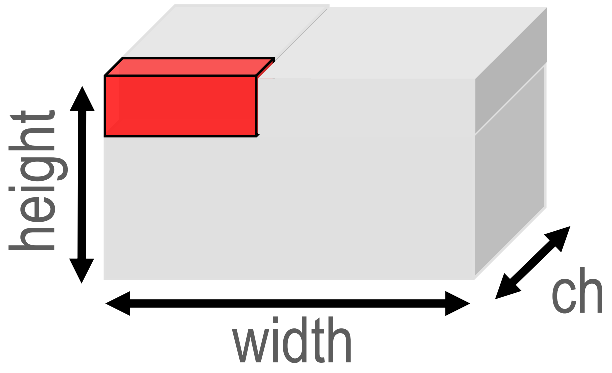


DORY: Tiler – L2/L1

L2/L1 Tiling:

- Relatively low L2 Memory 🙁
- Large Memory Bandwidth 🙂

All tiles from L3 to L2 are 3D



L2/L1 tiling is formalized as an **optimization problem.**

We use **Constraint Programming** to formalize the problem and find a feasible solution

DORY: Tiler – L2/L1

Constraint Programming problem → tiles size

$$\mathbf{cost} = \max \; Size(W_{tile}) + Size(x_{tile}) + Size(y_{tile})$$

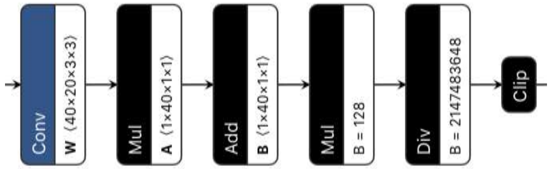
MEMORY → $\text{s.t.} \quad Size(W_{tile}) + Size(x_{tile}) + Size(y_{tile}) < \; L1size$

GEOMETRY → $\text{s.t.} \quad \{y_{tile}[ch_{out}] = W_{tile}[ch_{out}], \ldots\}$

EFF. HEURISTICS → $\mathbf{cost}' = \mathbf{cost} + \; \{y_{tile}[ch_{out}] \text{ divisible by } 4, \ldots\}$

Performance is maximum for configurations that use PULP-NN primitives more efficiently (e.g., full parallelism)

ONNX



Integer DNN

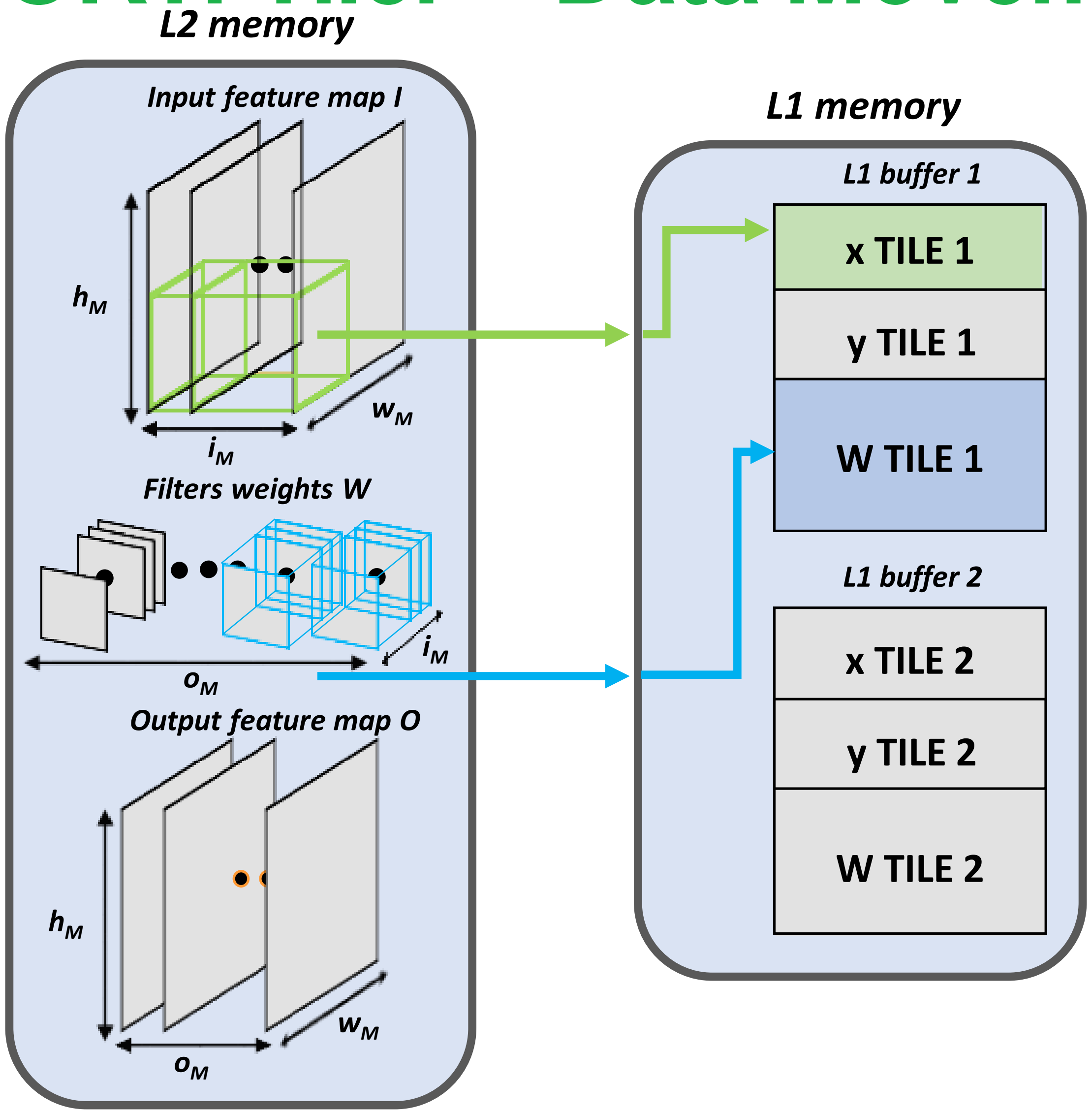
↓

Google **ORTools**

↓

Integer DNN
+
tile sizes

DORY: Tiler – Data Movement



L2 memory

Input feature map I

Filters weights W

Output feature map O

L1 memory

L1 buffer 1

x TILE 1

y TILE 1

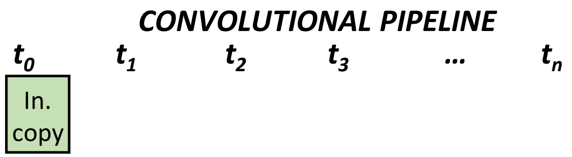
W TILE 1

L1 buffer 2

x TILE 2

y TILE 2

W TILE 2

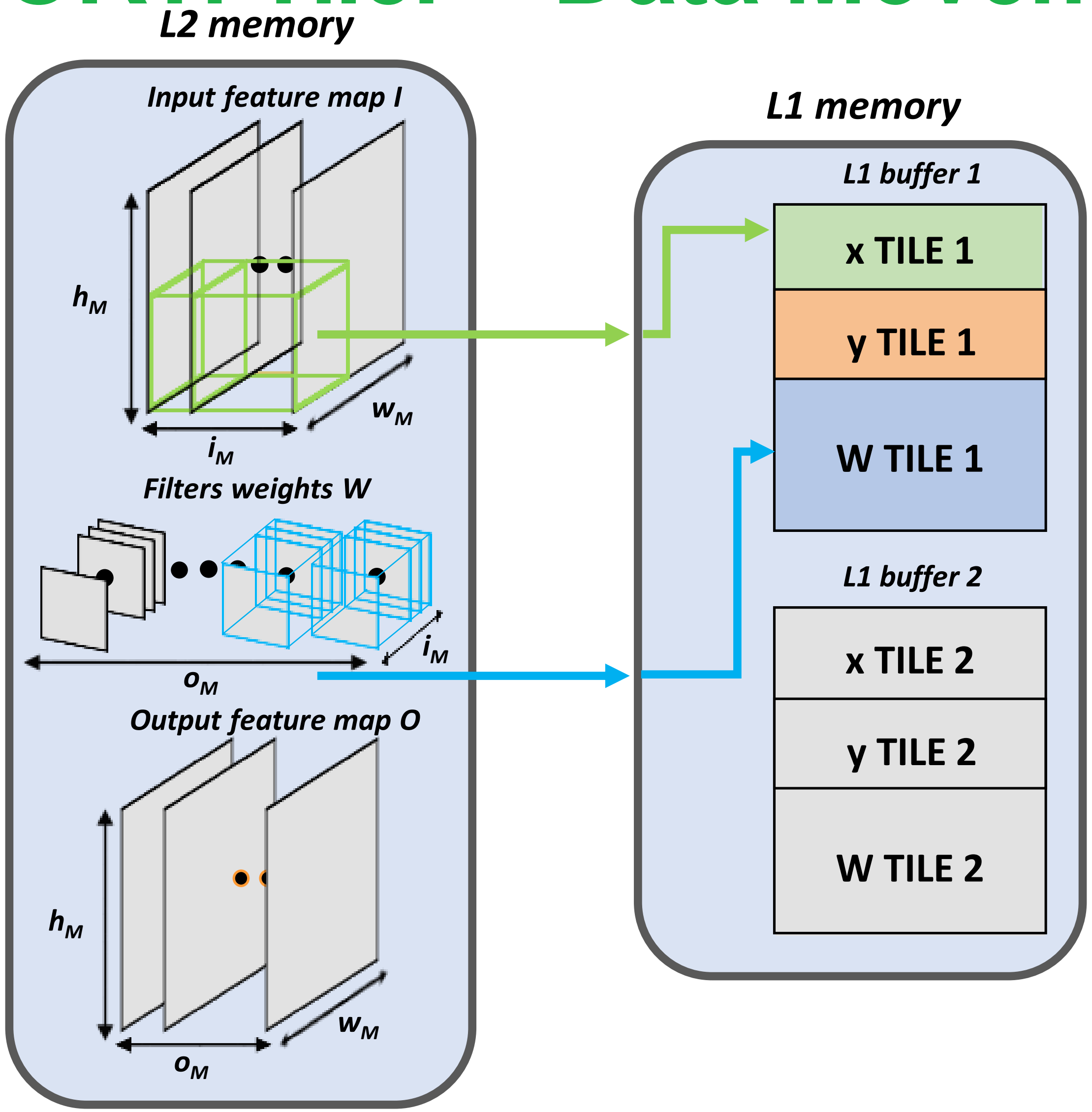


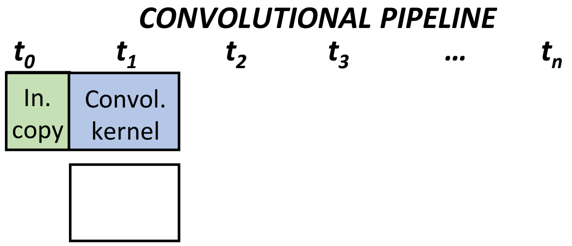
CONVOLUTIONAL PIPELINE

🟩 DMA ch. 0-1 🟧 DMA ch. 2

🟦 Cluster computation

DORY: Tiler – Data Movement



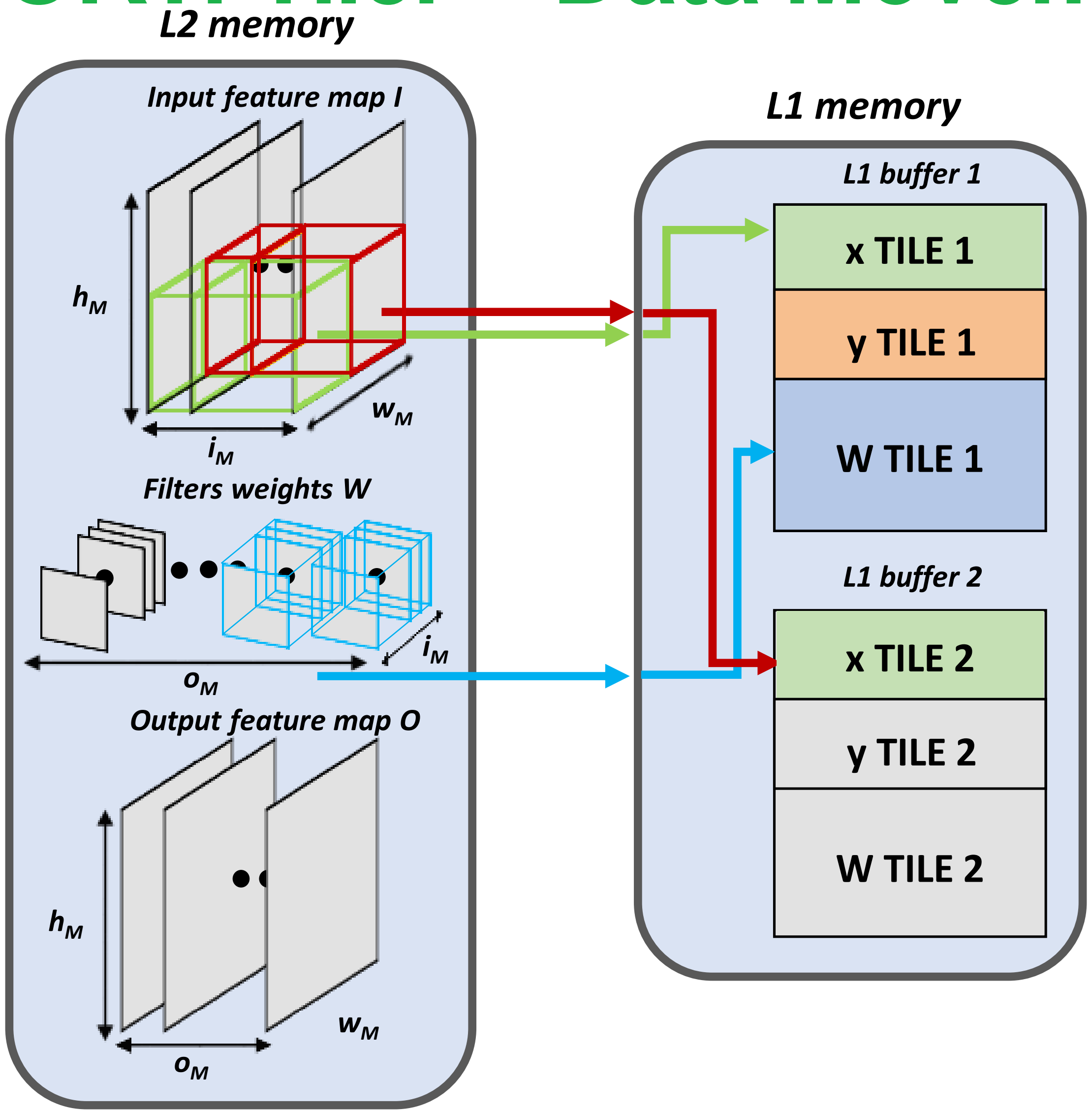


🟩 DMA ch. 0-1

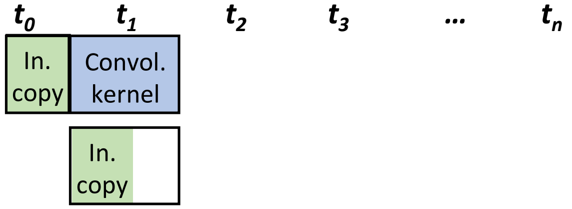
🟧 DMA ch. 2

🟦 Cluster computation

DORY: Tiler – Data Movement



CONVOLUTIONAL PIPELINE

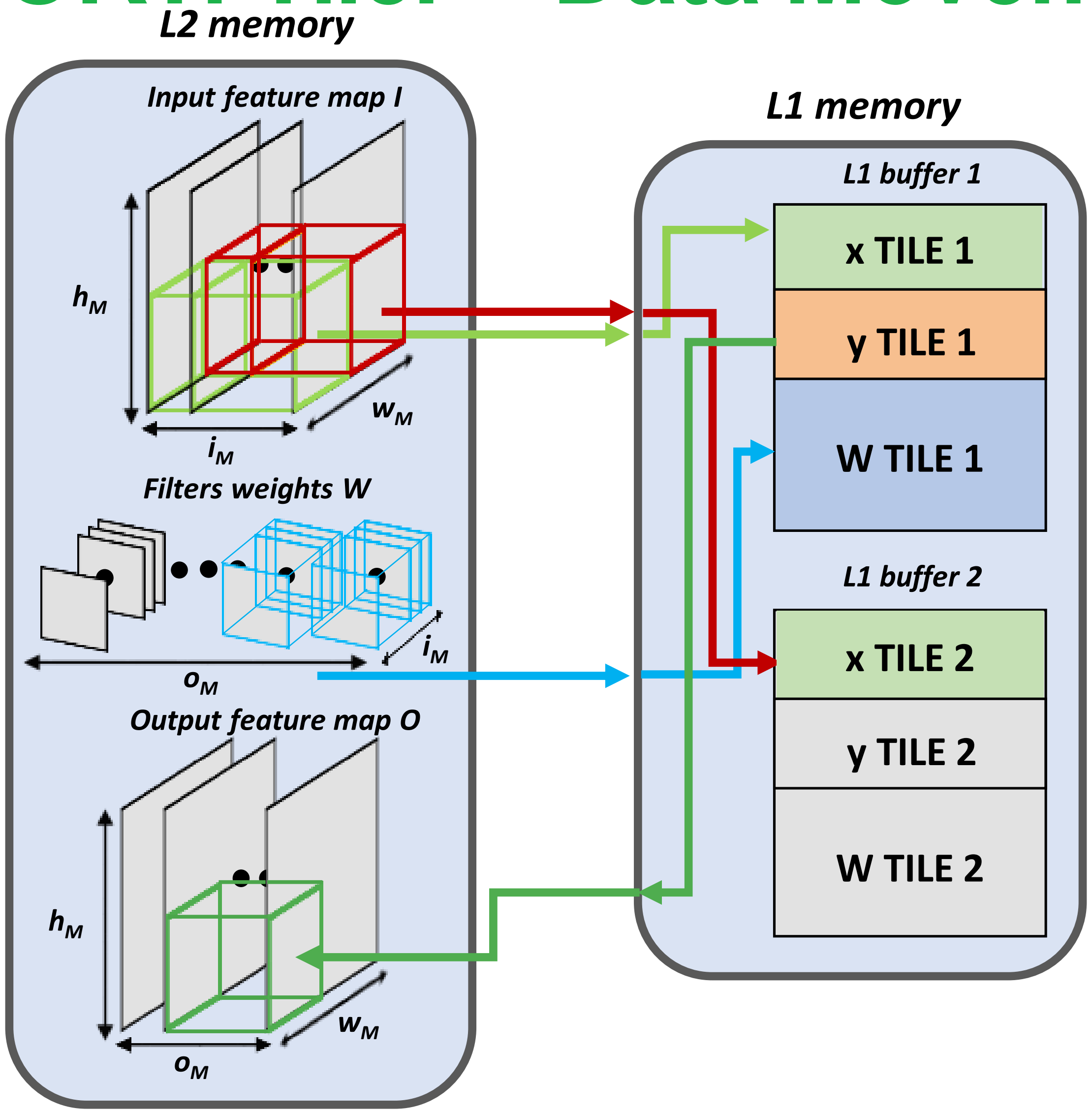


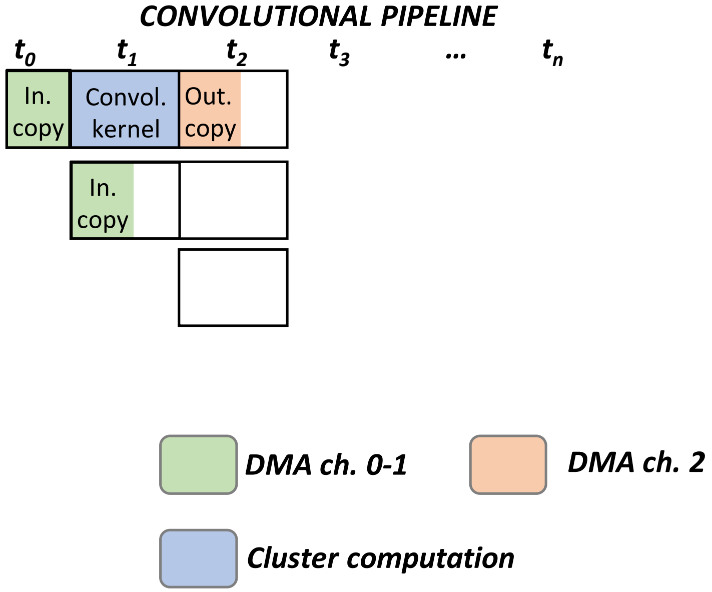
🟩 DMA ch. 0-1

🟧 DMA ch. 2

🟦 Cluster computation

DORY: Tiler – Data Movement



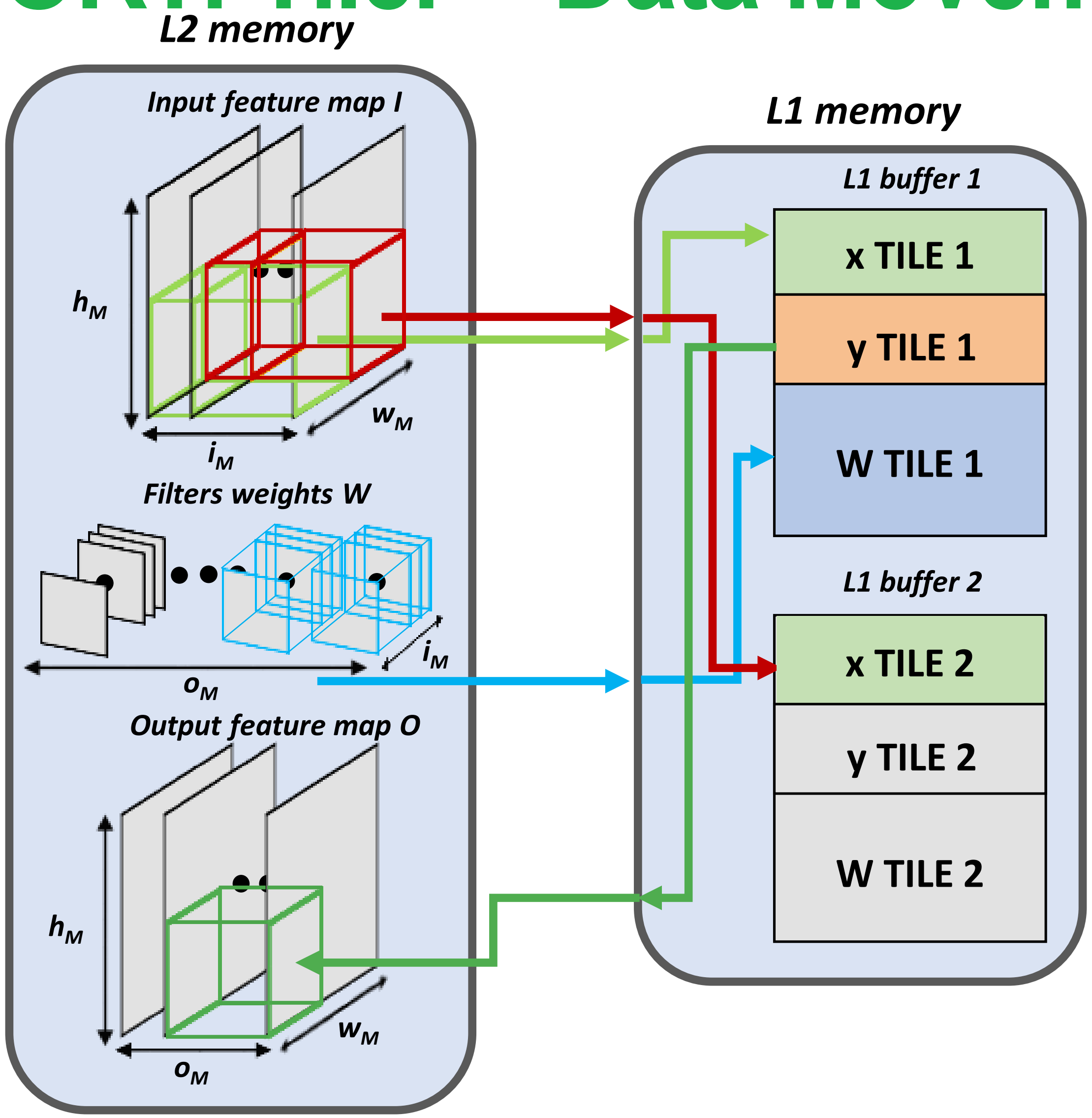


🟩 DMA ch. 0-1

🟧 DMA ch. 2

🟦 Cluster computation

DORY: Tiler – Data Movement



CONVOLUTIONAL PIPELINE

t_0	t_1	t_2	t_3	…	t_n
In. copy	Convol. kernel	Out. copy	In. copy	• • •	Convol. kernel
	In. copy	Convol. kernel	Out. copy	• • •	In. copy
		In. copy	Convol. kernel	• • •	Out. copy

🟩 DMA ch. 0-1

🟧 DMA ch. 2

🟦 Cluster computation

DORY: Tiling & Code Generation

DORY

D_eployment_ ***O***_riented to memo_***RY***

1. Reading of the ONNX output
    1. Recognize backend implemented nodes
    2. Reconstruct the graph with backend nodes input-output dimensions
2. Layer-by-Layer tiling
    1. L3-L2 tiling
    2. L2-L1 tiling
    3. Memory allocation in L2
3. **Layer template compilation**
4. Network compilation

DORY: Template Writing

Neural Network Layers generation

mako.template → python compilation of c templates

```
dory_dma_memcpy_3d(input_0, ${args});
dory_dma_memcpy_3d(weights_0, ${args});
dory_dma_wait();
for (i=0; i<${tile_dim_nof * tile_dim_nif * tile_dim_h * tile_dim_w}; i++)
      dory_dma_memcpy_3d(input_i+1, ${args});
      dory_dma_memcpy_3d(weights_i+1, ${args});
      pulp_nn_conv(input_i, weights_i, output, ${args});
      dory_dma_wait();
      dory_dma_memcpy_3d(output, ${args});
```

DORY: Template Writing

Neural Network Layers generation

mako.template → python compilation of c templates

```
dory_dma_memcpy_3d(input_0, ${args});
dory_dma_memcpy_3d(weights_0, ${args});
dory_dma_wait();
for (i=0; i<${tile_dim_nof * tile_dim_nif * tile_dim_h * tile_dim_w}; i++)
        dory_dma_memcpy_3d(input_i+1, ${args});
        dory_dma_memcpy_3d(weights_i+1, ${args});
        pulp_nn_conv(input_i, weights_i, output, ${args});
        dory_dma_wait();
        dory_dma_memcpy_3d(output, ${args});
```

Network exported parameters

pulp_nn kernel

DORY: Template Writing

Neural Network Layers generation

mako.template → python compilation of c templates

L2/L1 memory copies

```
dory_dma_memcpy_3d(input_0, ${args});
dory_dma_memcpy_3d(weights_0, ${args});
dory_dma_wait();
for (i=0; i<${tile_dim_nof * tile_dim_nif * tile_dim_h * tile_dim_w}; i++)
        dory_dma_memcpy_3d(input_i+1, ${args});
        dory_dma_memcpy_3d(weights_i+1, ${args});
        pulp_nn_conv(input_i, weights_i, output, ${args});
        dory_dma_wait();
        dory_dma_memcpy_3d(output, ${args});
```

→ First tile allocation

DORY: Template Writing

Neural Network Layers generation

mako.template → python compilation of c templates

```
dory_dma_memcpy_3d(input_0, ${args});
dory_dma_memcpy_3d(weights_0, ${args});
dory_dma_wait();
for (i=0; i<${tile_dim_nof * tile_dim_nif * tile_dim_h * tile_dim_w}; i++)
        dory_dma_memcpy_3d(input_i+1, ${args});
        dory_dma_memcpy_3d(weights_i+1, ${args});
        pulp_nn_conv(input_i, weights_i, output, ${args});
        dory_dma_wait();
        dory_dma_memcpy_3d(output, ${args});
```

→ First tile allocation

→ Tile loop

DORY: Template Writing

Neural Network Layers generation

mako.template → python compilation of c templates

```
dory_dma_memcpy_3d(input_0, ${args});
dory_dma_memcpy_3d(weights_0, ${args});
dory_dma_wait();
for (i=0; i<${tile_dim_nof * tile_dim_nif * tile_dim_h * tile_dim_w}; i++)
      dory_dma_memcpy_3d(input_i+1, ${args});
      dory_dma_memcpy_3d(weights_i+1, ${args});
      pulp_nn_conv(input_i, weights_i, output, ${args});
      dory_dma_wait();
      dory_dma_memcpy_3d(output, ${args});
```

→ Async Data movement

→ Kernel Computation

→ Async Data movement

DORY: Tiling & Code Generation

DORY

__D__eployment __O__riented to memo__RY__

1. Reading of the ONNX output
    1. Recognize backend implemented nodes
    2. Reconstruct the graph with backend nodes input-output dimensions
2. Layer-by-Layer tiling
    1. L3-L2 tiling
    2. L2-L1 tiling
    3. Memory allocation in L2
3. Layer template compilation
4. **Network compilation**

DORY: Network Generation

Neural Network generation → mako.template

```
for (int i = 0; i < ${len(PULP_Nodes_Graph)}; i++)
        pi_cl_ram_read_wait(&buff_req1);
        pi_cl_ram_read(&ram, transfer_weights, ${args}, &buff_req1);
        switch (i)
        {
        % for i in range(len(PULP_Nodes_Graph)):
            case ${i}:
                ${func_name[i]}(args);
                break;
        % endfor
        }
       dory_L2_memory_management();
```

DORY: Network Generation

Neural Network generation → mako.template

Loop over layers

```
for (int i = 0; i < ${len(PULP_Nodes_Graph)}; i++)
         pi_cl_ram_read_wait(&buff_req1);
         pi_cl_ram_read(&ram, transfer_weights, ${args}, &buff_req1);
         switch (i)
         {
         % for i in range(len(PULP_Nodes_Graph)):
              case ${i}:
                  ${func_name[i]}(args);
                  break;
         % endfor
         }
        dory_L2_memory_management();
```

DORY: Network Generation

Neural Network generation → mako.template

L3 DMA weights memory copy

```
for (int i = 0; i < ${len(PULP_Nodes_Graph)}; i++)
    pi_cl_ram_read_wait(&buff_req1);
    pi_cl_ram_read(&ram, transfer_weights, ${args}, &buff_req1);
    switch (i)
    {
    % for i in range(len(PULP_Nodes_Graph)):
        case ${i}:
            ${func_name[i]}(args);
            break;
    % endfor
    }
    dory_L2_memory_management();
```

DORY: Network Generation

Neural Network generation → mako.template

```
for (int i = 0; i < ${len(PULP_Nodes_Graph)}; i++)
    pi_cl_ram_read_wait(&buff_req1);
    pi_cl_ram_read(&ram, transfer_weights, ${args}, &buff_req1);
    switch (i)
    {
    % for i in range(len(PULP_Nodes_Graph)):
      case ${i}:
        ${func_name[i]}(args);
        break;
    % endfor
    }
    dory_L2_memory_management();
```

Convolutional layers

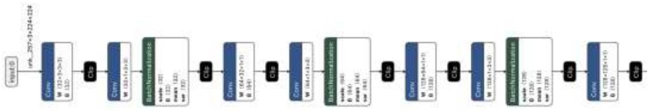
DORY: Network Generation

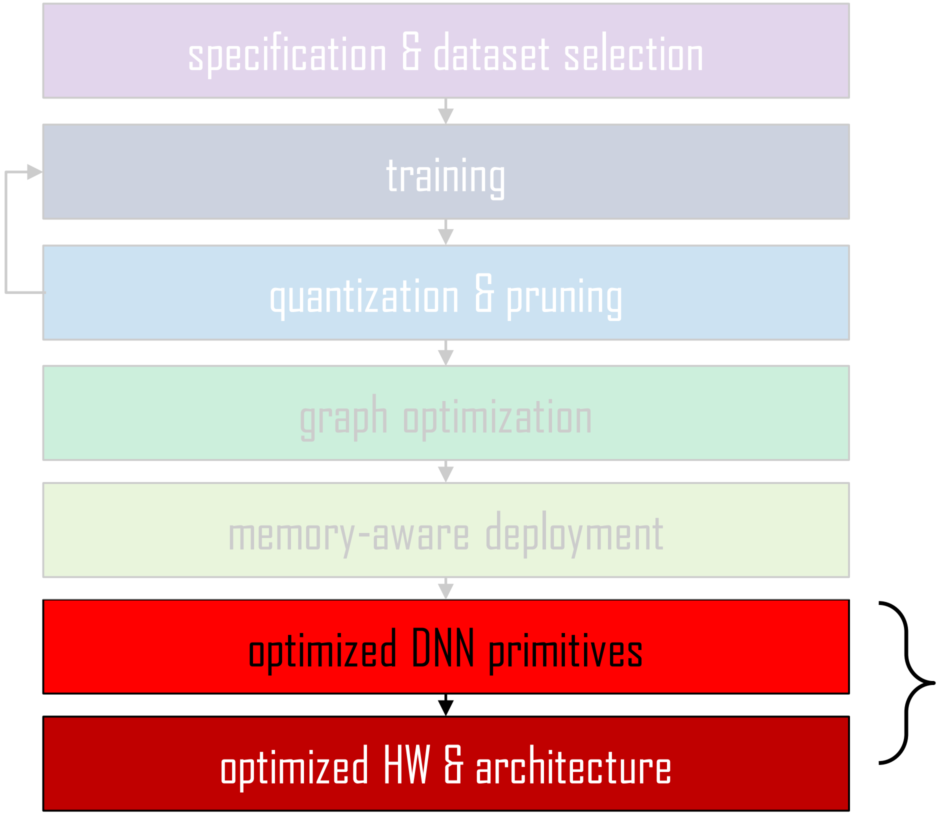
Neural Network generation → mako.template

```
for (int i = 0; i < ${len(PULP_Nodes_Graph)}; i++)
      pi_cl_ram_read_wait(&buff_req1);
      pi_cl_ram_read(&ram, transfer_weights, ${args}, &buff_req1);
      switch (i)
      {
      % for i in range(len(PULP_Nodes_Graph)):
         case ${i}:
            ${func_name[i]}(args);
            break;
      % endfor
      }
     dory_L2_memory_management();
```

L2 memory allocation/deallocation

PULP-NN: Optimized Back-End





PULP-NN
Parallel ULP
Neural Network library

PULP-NN: Optimized Back-End

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

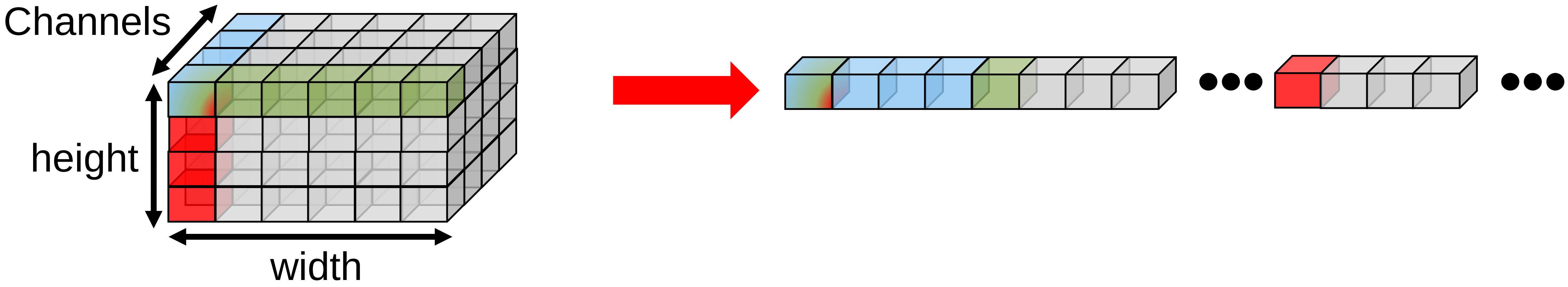
PULP-NN: Optimized Back-End

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

↓

HWC format



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

PULP-NN: Optimized Back-End

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

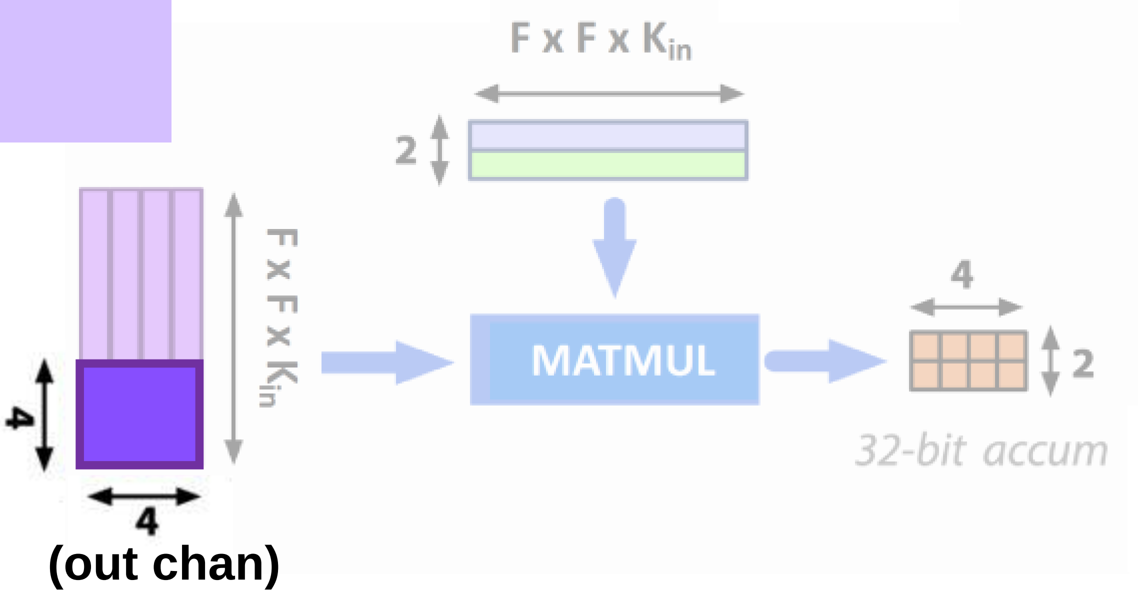
HWC format

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Load 16 weights (8-bit)

4 out chan, 4 in chan

address post-increment



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

PULP-NN: Optimized Back-End

Target **int8** execution of CONV, FC, ... primitives

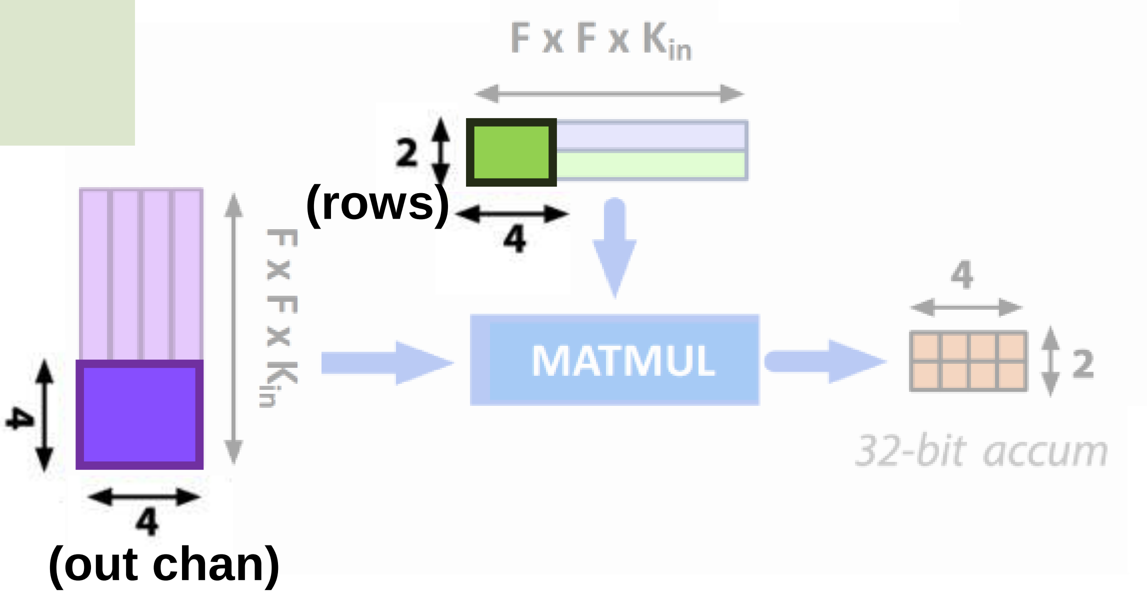
1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Load 8 pixels

2 rows, 4 in chan

address post-increment



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

PULP-NN: Optimized Back-End

Target **int8** execution of CONV, FC, ... primitives

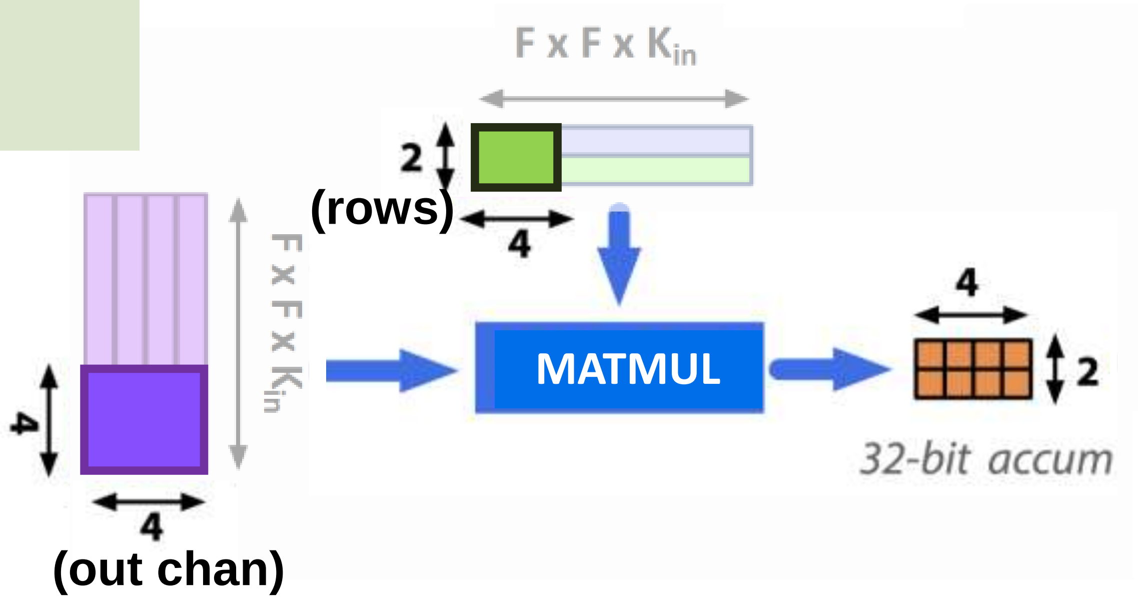
1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Load 8 pixels

2 rows, 4 in chan

address post-increment



Compute 32 MAC over 8 accumulators

dot-product instructions

PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

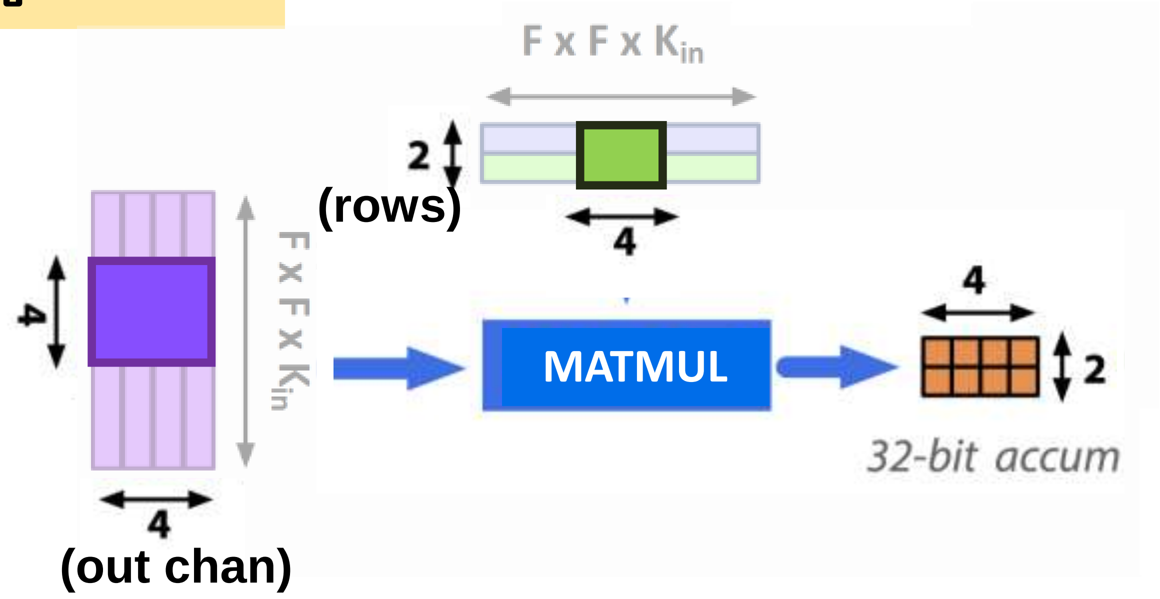
PULP-NN: Optimized Back-End

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Loop over in chan, filter size



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

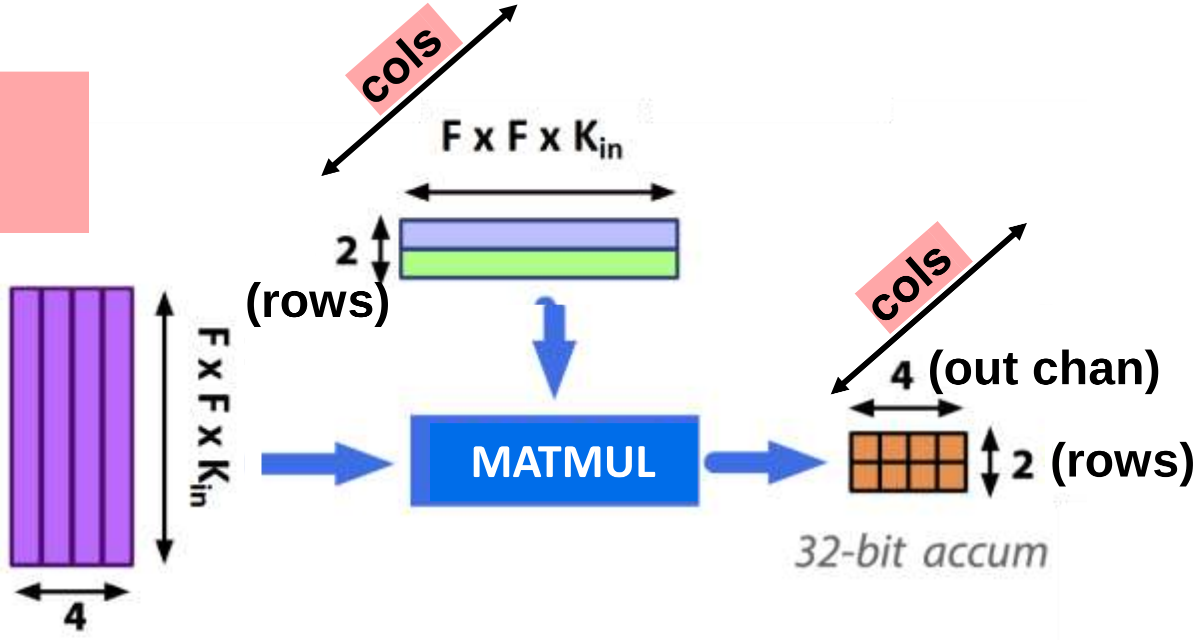
PULP-NN: Optimized Back-End

Target **int8** execution of CONV, FC, ... primitives

1) maximize **data reuse in register file** 2) improve **kernel regularity** 3) exploit **parallelism**

```
lp.setup
  p.lw  w0, 4(W0!)
  p.lw  w1, 4(W1!)
  p.lw  w2, 4(W2!)
  p.lw  w3, 4(W3!)
  p.lw  x1, 4(X0!)
  p.lw  x2, 4(X1!)
  pv.sdotsp.b    acc1, w0, x0
  pv.sdotsp.b    acc2, w0, x1
  pv.sdotsp.b    acc3, w1, x0
  pv.sdotsp.b    acc4, w1, x1
  pv.sdotsp.b    acc5, w2, x0
  pv.sdotsp.b    acc6, w2, x1
  pv.sdotsp.b    acc7, w3, x0
  pv.sdotsp.b    acc8, w3, x1
end
```

Parallelize over 8 cores
column dimension



PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

PULP-NN: Layers Supported (@ 25-2-2021)

Convolutions

- Conv_Ho_parallel (+bn, +Relu)
- Conv_HoWo_parallel (+bn, +Relu)
- Conv_Co_parallel (+bn, +Relu)

Point-wise Convolutions

- Pointwise_Ho_parallel (+bn, +Relu)
- Pointwise_HoWo_parallel (+bn, +Relu)
- Pointwise_Co_parallel (+bn, +Relu)

Depth-wise Convolutions

- Depthwise_3x3s1 (+bn, +Relu)
- Depthwise_generic (+bn, +Relu)

Linear Layers

- Linear (+bn, +Relu)
- Linear_out_fp32

Other Layers

- Add (+bn, +Relu)
- Avgpool
- Maxpool

PULP-NN [Garofalo 19] https://arxiv.org/abs/1908.11263

https://github.com/pulp-platform/pulp-nn

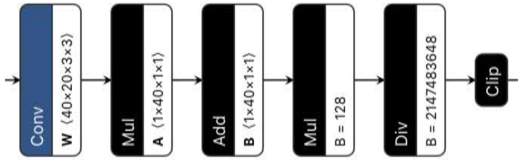
Requirements – DORY + PULP-NN

- **DORY is available at https://github.com/pulp-platform/dory**
- **On Ubuntu 18.04 you need the following packages and tools:**
  - **python>=3.6** or **python3.5 with future-fstrings** package
  - **pulp-sdk** available at **https://github.com/pulp-platform/pulp-sdk**
  - Python packages:
    - **onnx>=1.8.1**
    - **torch>=1.5.1**
    - **pandas>=0.24.2**
    - **ortools>=8.0.8283**

- No installation required for DORY and PULP-NN

https://github.com/pulp-platform/pulp-nn

Network Generation



Integer Network + tile sizes

Code Generation
from templates

Network-level C code

- L3/L2 transfer boilerplate
- double buffering for weights
- calls to layer-level code

Layer-level C code

- L2/L1 transfer boilerplate
- calls to PULP-NN backend library

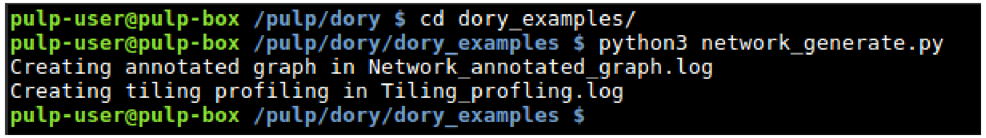
NEMO
Post-training Tutorial:
https://github.com/pulp-platform/nemo

DORY
Tutorial:
https://github.com/pulp-platform/dory_examples

Full stack tutorial in the SDK documentation
https://github.com/pulp-platform/pulp-sdk

Generate a neural network with default settings

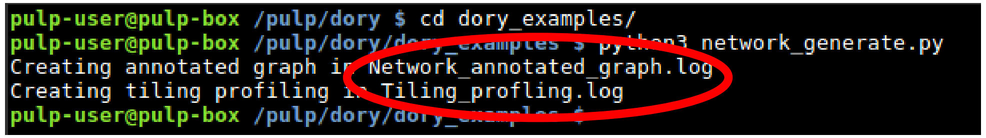
- Generate the default network



```
pulp-user@pulp-box /pulp/dory $ cd dory_examples/
pulp-user@pulp-box /pulp/dory/dory_examples $ python3 network_generate.py
Creating annotated graph in Network_annotated_graph.log
Creating tiling profiling in Tiling_profling.log
pulp-user@pulp-box /pulp/dory/dory_examples $
```

Generate a neural network with default settings

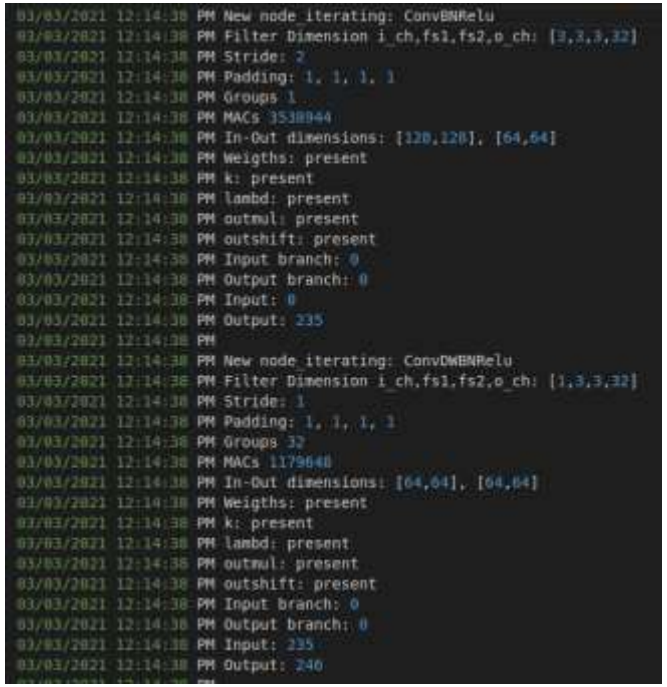
- Generate the default network



```
pulp-user@pulp-box /pulp/dory $ cd dory_examples/
pulp-user@pulp-box /pulp/dory/dory_examples $ python3 network_generate.py
Creating annotated graph in Network_annotated_graph.log
Creating tiling profiling in Tiling_profling.log
pulp-user@pulp-box /pulp/dory/dory_examples $
```

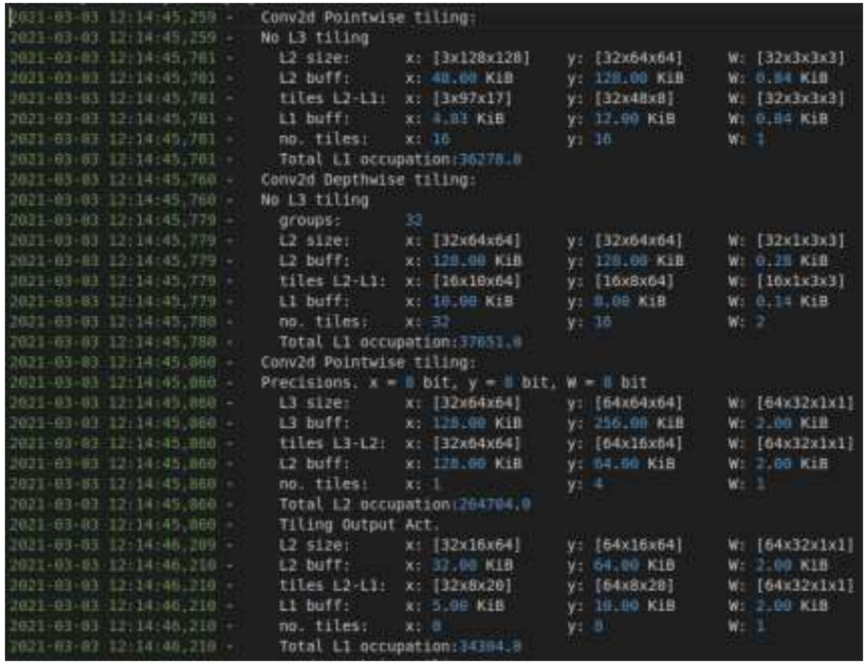
- Inspect the two output files

Network_annotated_graph



```
03/03/2021 12:14:38 PM New node_iterating: ConvBNRelu
03/03/2021 12:14:38 PM Filter Dimension i_ch,fs1,fs2,o_ch: [3,3,3,32]
03/03/2021 12:14:38 PM Stride: 2
03/03/2021 12:14:38 PM Padding: 1, 1, 1, 1
03/03/2021 12:14:38 PM Groups 1
03/03/2021 12:14:38 PM MACs 3538944
03/03/2021 12:14:38 PM In-Out dimensions: [128,128], [64,64]
03/03/2021 12:14:38 PM Weigths: present
03/03/2021 12:14:38 PM k: present
03/03/2021 12:14:38 PM lambd: present
03/03/2021 12:14:38 PM outmul: present
03/03/2021 12:14:38 PM outshift: present
03/03/2021 12:14:38 PM Input branch: 0
03/03/2021 12:14:38 PM Output branch: 0
03/03/2021 12:14:38 PM Input: 0
03/03/2021 12:14:38 PM Output: 235
03/03/2021 12:14:38 PM
03/03/2021 12:14:38 PM New node_iterating: ConvDWBNRelu
03/03/2021 12:14:38 PM Filter Dimension i_ch,fs1,fs2,o_ch: [1,3,3,32]
03/03/2021 12:14:38 PM Stride: 1
03/03/2021 12:14:38 PM Padding: 1, 1, 1, 1
03/03/2021 12:14:38 PM Groups 32
03/03/2021 12:14:38 PM MACs 1179648
03/03/2021 12:14:38 PM In-Out dimensions: [64,64], [64,64]
03/03/2021 12:14:38 PM Weigths: present
03/03/2021 12:14:38 PM k: present
03/03/2021 12:14:38 PM lambd: present
03/03/2021 12:14:38 PM outmul: present
03/03/2021 12:14:38 PM outshift: present
03/03/2021 12:14:38 PM Input branch: 0
03/03/2021 12:14:38 PM Output branch: 0
03/03/2021 12:14:38 PM Input: 235
03/03/2021 12:14:38 PM Output: 240
03/03/2021 12:14:38 PM
```

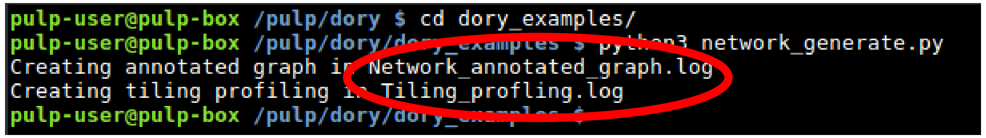
Tiling profiling



```
2021-03-03 12:14:45,259 -   Conv2d Pointwise tiling:
2021-03-03 12:14:45,259 -   No L3 tiling
2021-03-03 12:14:45,761 -     L2 size:      x: [3x128x128]     y: [32x64x64]     W: [32x3x3x3]
2021-03-03 12:14:45,761 -     L2 buff:      x: 48.00 KiB       y: 128.00 KiB     W: 0.84 KiB
2021-03-03 12:14:45,761 -     tiles L2-L1:  x: [3x97x17]       y: [32x48x8]      W: [32x3x3x3]
2021-03-03 12:14:45,761 -     L1 buff:      x: 4.83 KiB        y: 12.00 KiB      W: 0.84 KiB
2021-03-03 12:14:45,761 -     no. tiles:    x: 16              y: 16             W: 1
2021-03-03 12:14:45,761 -     Total L1 occupation:36278.0
2021-03-03 12:14:45,768 -   Conv2d Depthwise tiling:
2021-03-03 12:14:45,768 -   No L3 tiling
2021-03-03 12:14:45,779 -     groups:       32
2021-03-03 12:14:45,779 -     L2 size:      x: [32x64x64]      y: [32x64x64]     W: [32x1x3x3]
2021-03-03 12:14:45,779 -     L2 buff:      x: 128.00 KiB      y: 128.00 KiB     W: 0.28 KiB
2021-03-03 12:14:45,779 -     tiles L2-L1:  x: [16x16x64]      y: [16x8x64]      W: [16x1x3x3]
2021-03-03 12:14:45,779 -     L1 buff:      x: 16.00 KiB       y: 8.00 KiB       W: 0.14 KiB
2021-03-03 12:14:45,780 -     no. tiles:    x: 32              y: 16             W: 2
2021-03-03 12:14:45,780 -     Total L1 occupation:37651.0
2021-03-03 12:14:45,860 -   Conv2d Pointwise tiling:
2021-03-03 12:14:45,860 -   Precisions. x = 8 bit, y = 8 bit, W = 8 bit
2021-03-03 12:14:45,860 -     L3 size:      x: [32x64x64]      y: [64x64x64]     W: [64x32x1x1]
2021-03-03 12:14:45,860 -     L3 buff:      x: 128.00 KiB      y: 256.00 KiB     W: 2.00 KiB
2021-03-03 12:14:45,860 -     tiles L3-L2:  x: [32x64x64]      y: [64x16x64]     W: [64x32x1x1]
2021-03-03 12:14:45,860 -     L2 buff:      x: 128.00 KiB      y: 64.00 KiB      W: 2.00 KiB
2021-03-03 12:14:45,860 -     no. tiles:    x: 1               y: 4              W: 1
2021-03-03 12:14:45,860 -     Total L2 occupation:264704.0
2021-03-03 12:14:45,860 -     Tiling Output Act.
2021-03-03 12:14:46,209 -     L2 size:      x: [32x16x64]      y: [64x16x64]     W: [64x32x1x1]
2021-03-03 12:14:46,210 -     L2 buff:      x: 32.00 KiB       y: 64.00 KiB      W: 2.00 KiB
2021-03-03 12:14:46,210 -     tiles L2-L1:  x: [32x8x20]       y: [64x8x20]      W: [64x32x1x1]
2021-03-03 12:14:46,210 -     L1 buff:      x: 5.00 KiB        y: 10.00 KiB      W: 2.00 KiB
2021-03-03 12:14:46,210 -     no. tiles:    x: 8               y: 8              W: 1
2021-03-03 12:14:46,210 -     Total L1 occupation:34304.0
```

Generate a neural network with default settings

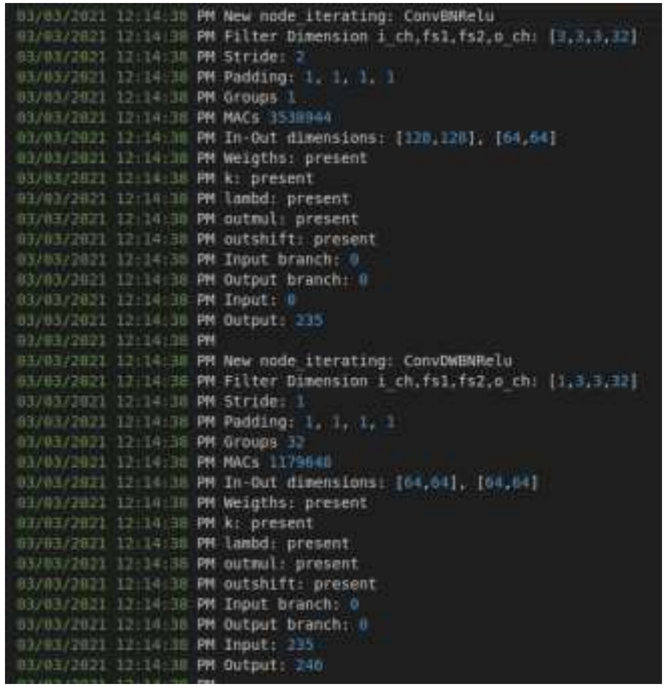
- Generate the default network



```
pulp-user@pulp-box /pulp/dory $ cd dory_examples/
pulp-user@pulp-box /pulp/dory/dory_examples $ python3 network_generate.py
Creating annotated graph in Network_annotated_graph.log
Creating tiling profiling in Tiling_profling.log
pulp-user@pulp-box /pulp/dory/dory_examples $
```

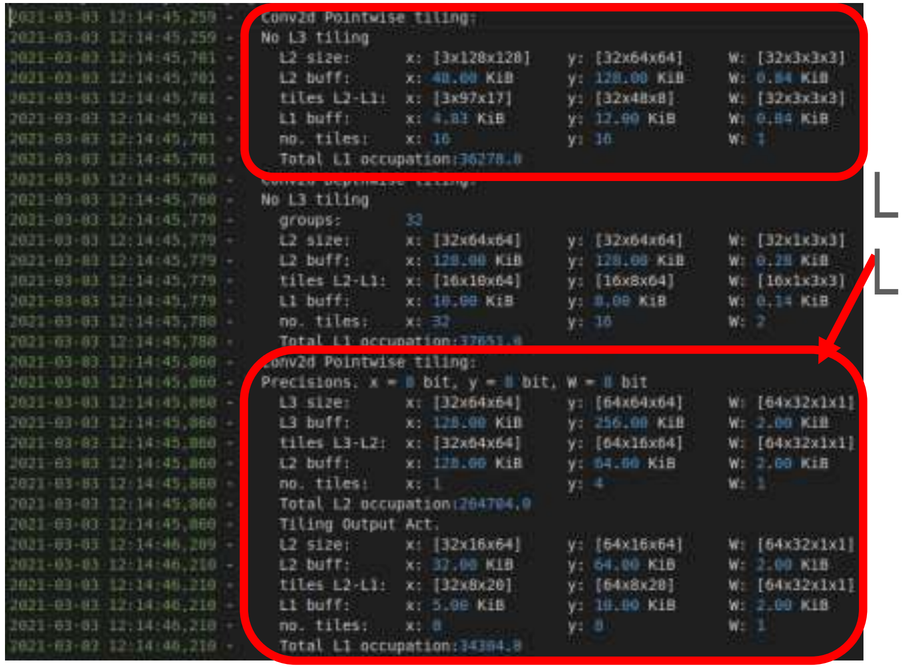
- Inspect the two output files

Network_annotated_graph



```
03/03/2021 12:14:38 PM New node_iterating: ConvBNRelu
03/03/2021 12:14:38 PM Filter Dimension i_ch,fs1,fs2,o_ch: [3,3,3,32]
03/03/2021 12:14:38 PM Stride: 2
03/03/2021 12:14:38 PM Padding: 1, 1, 1, 1
03/03/2021 12:14:38 PM Groups 1
03/03/2021 12:14:38 PM MACs 3538944
03/03/2021 12:14:38 PM In-Out dimensions: [128,128], [64,64]
03/03/2021 12:14:38 PM Weigths: present
03/03/2021 12:14:38 PM k: present
03/03/2021 12:14:38 PM lambd: present
03/03/2021 12:14:38 PM outmul: present
03/03/2021 12:14:38 PM outshift: present
03/03/2021 12:14:38 PM Input branch: 0
03/03/2021 12:14:38 PM Output branch: 0
03/03/2021 12:14:38 PM Input: 0
03/03/2021 12:14:38 PM Output: 235
03/03/2021 12:14:38 PM
03/03/2021 12:14:38 PM New node_iterating: ConvDWBNRelu
03/03/2021 12:14:38 PM Filter Dimension i_ch,fs1,fs2,o_ch: [1,3,3,32]
03/03/2021 12:14:38 PM Stride: 1
03/03/2021 12:14:38 PM Padding: 1, 1, 1, 1
03/03/2021 12:14:38 PM Groups 32
03/03/2021 12:14:38 PM MACs 1179648
03/03/2021 12:14:38 PM In-Out dimensions: [64,64], [64,64]
03/03/2021 12:14:38 PM Weigths: present
03/03/2021 12:14:38 PM k: present
03/03/2021 12:14:38 PM lambd: present
03/03/2021 12:14:38 PM outmul: present
03/03/2021 12:14:38 PM outshift: present
03/03/2021 12:14:38 PM Input branch: 0
03/03/2021 12:14:38 PM Output branch: 0
03/03/2021 12:14:38 PM Input: 235
03/03/2021 12:14:38 PM Output: 240
03/03/2021 12:14:38 PM
```

Tiling profiling



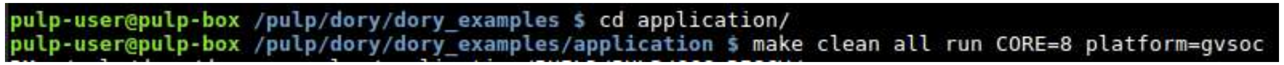
```
2021-03-03 12:14:45,259 - Conv2d Pointwise tiling:
2021-03-03 12:14:45,259 - No L3 tiling
2021-03-03 12:14:45,761 -   L2 size:      x: [3x128x128]    y: [32x64x64]    W: [32x3x3x3]
2021-03-03 12:14:45,761 -   L2 buff:      x: 48.00 KiB      y: 128.00 KiB    W: 0.84 KiB
2021-03-03 12:14:45,761 -   tiles L2-L1:  x: [3x97x17]      y: [32x48x8]     W: [32x3x3x3]
2021-03-03 12:14:45,761 -   L1 buff:      x: 4.83 KiB       y: 12.00 KiB     W: 0.84 KiB
2021-03-03 12:14:45,761 -   no. tiles:    x: 16             y: 16            W: 1
2021-03-03 12:14:45,761 -   Total L1 occupation:36278.0
2021-03-03 12:14:45,760 - Conv2d Depthwise tiling:
2021-03-03 12:14:45,760 - No L3 tiling
2021-03-03 12:14:45,779 -   groups:       32
2021-03-03 12:14:45,779 -   L2 size:      x: [32x64x64]     y: [32x64x64]    W: [32x1x3x3]
2021-03-03 12:14:45,779 -   L2 buff:      x: 128.00 KiB     y: 128.00 KiB    W: 0.28 KiB
2021-03-03 12:14:45,779 -   tiles L2-L1:  x: [16x16x64]     y: [16x8x64]     W: [16x1x3x3]
2021-03-03 12:14:45,779 -   L1 buff:      x: 16.00 KiB      y: 8.00 KiB      W: 0.14 KiB
2021-03-03 12:14:45,780 -   no. tiles:    x: 32             y: 16            W: 2
2021-03-03 12:14:45,780 -   Total L1 occupation:37651.0
2021-03-03 12:14:45,860 - Conv2d Pointwise tiling:
2021-03-03 12:14:45,860 - Precisions. x = 8 bit, y = 8 bit, W = 8 bit
2021-03-03 12:14:45,860 -   L3 size:      x: [32x64x64]     y: [64x64x64]    W: [64x32x1x1]
2021-03-03 12:14:45,860 -   L3 buff:      x: 128.00 KiB     y: 256.00 KiB    W: 2.00 KiB
2021-03-03 12:14:45,860 -   tiles L3-L2:  x: [32x64x64]     y: [64x16x64]    W: [64x32x1x1]
2021-03-03 12:14:45,860 -   L2 buff:      x: 128.00 KiB     y: 64.00 KiB     W: 2.00 KiB
2021-03-03 12:14:45,860 -   no. tiles:    x: 1              y: 4             W: 1
2021-03-03 12:14:45,860 -   Total L2 occupation:264704.0
2021-03-03 12:14:45,860 -   Tiling Output Act.
2021-03-03 12:14:46,209 -   L2 size:      x: [32x16x64]     y: [64x16x64]    W: [64x32x1x1]
2021-03-03 12:14:46,210 -   L2 buff:      x: 32.00 KiB      y: 64.00 KiB     W: 2.00 KiB
2021-03-03 12:14:46,210 -   tiles L2-L1:  x: [32x8x20]      y: [64x8x20]     W: [64x32x1x1]
2021-03-03 12:14:46,210 -   L1 buff:      x: 5.00 KiB       y: 10.00 KiB     W: 2.00 KiB
2021-03-03 12:14:46,210 -   no. tiles:    x: 8              y: 8             W: 1
2021-03-03 12:14:46,210 -   Total L1 occupation:34304.0
```

L2-L1 tiling

L3-L2 tiling +
L2-L1 tiling

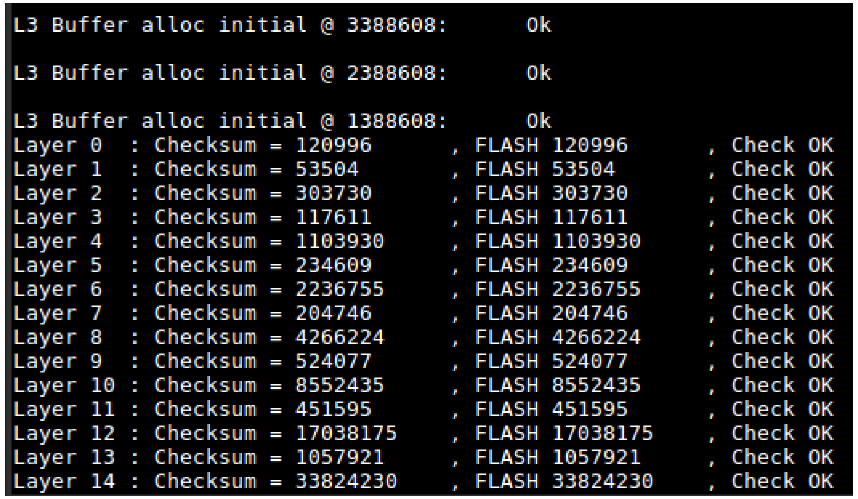
Generate a neural network with default settings

- Run the network on pulp gvsoc



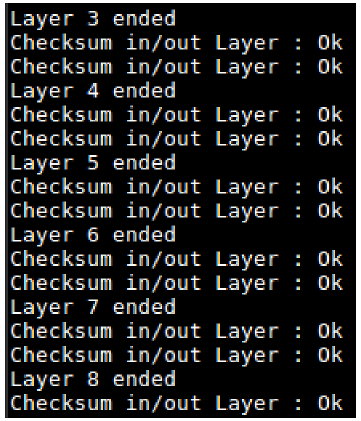
```
pulp-user@pulp-box /pulp/dory/dory_examples $ cd application/
pulp-user@pulp-box /pulp/dory/dory_examples/application $ make clean all run CORE=8 platform=gvsoc
```

Weights checksum



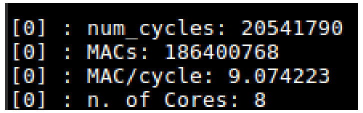
```
L3 Buffer alloc initial @ 3388608:      Ok

L3 Buffer alloc initial @ 2388608:      Ok

L3 Buffer alloc initial @ 1388608:      Ok
Layer 0  : Checksum = 120996      , FLASH 120996      , Check OK
Layer 1  : Checksum = 53504       , FLASH 53504       , Check OK
Layer 2  : Checksum = 303730      , FLASH 303730      , Check OK
Layer 3  : Checksum = 117611      , FLASH 117611      , Check OK
Layer 4  : Checksum = 1103930     , FLASH 1103930     , Check OK
Layer 5  : Checksum = 234609      , FLASH 234609      , Check OK
Layer 6  : Checksum = 2236755     , FLASH 2236755     , Check OK
Layer 7  : Checksum = 204746      , FLASH 204746      , Check OK
Layer 8  : Checksum = 4266224     , FLASH 4266224     , Check OK
Layer 9  : Checksum = 524077      , FLASH 524077      , Check OK
Layer 10 : Checksum = 8552435     , FLASH 8552435     , Check OK
Layer 11 : Checksum = 451595      , FLASH 451595      , Check OK
Layer 12 : Checksum = 17038175    , FLASH 17038175    , Check OK
Layer 13 : Checksum = 1057921     , FLASH 1057921     , Check OK
Layer 14 : Checksum = 33824230    , FLASH 33824230    , Check OK
```

Activations checksum



```
Layer 3 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
Layer 4 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
Layer 5 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
Layer 6 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
Layer 7 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
Layer 8 ended
Checksum in/out Layer : Ok
```

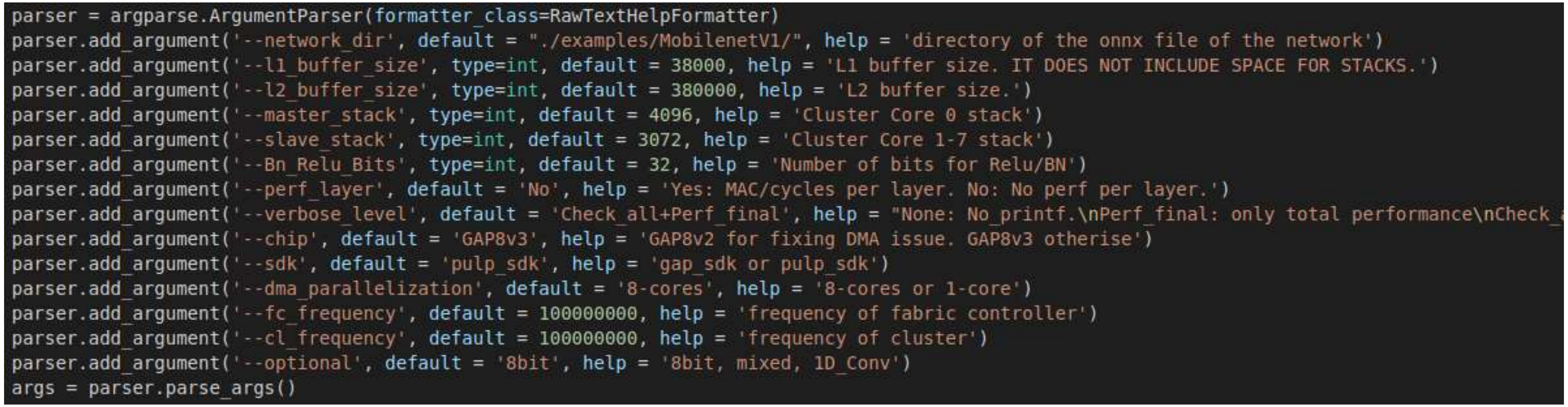
Performance



```
[0] : num_cycles: 20541790
[0] : MACs: 186400768
[0] : MAC/cycle: 9.074223
[0] : n. of Cores: 8
```

Change default settings

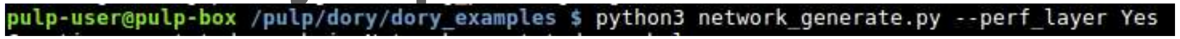
- Set of arguments that you can pass to DORY



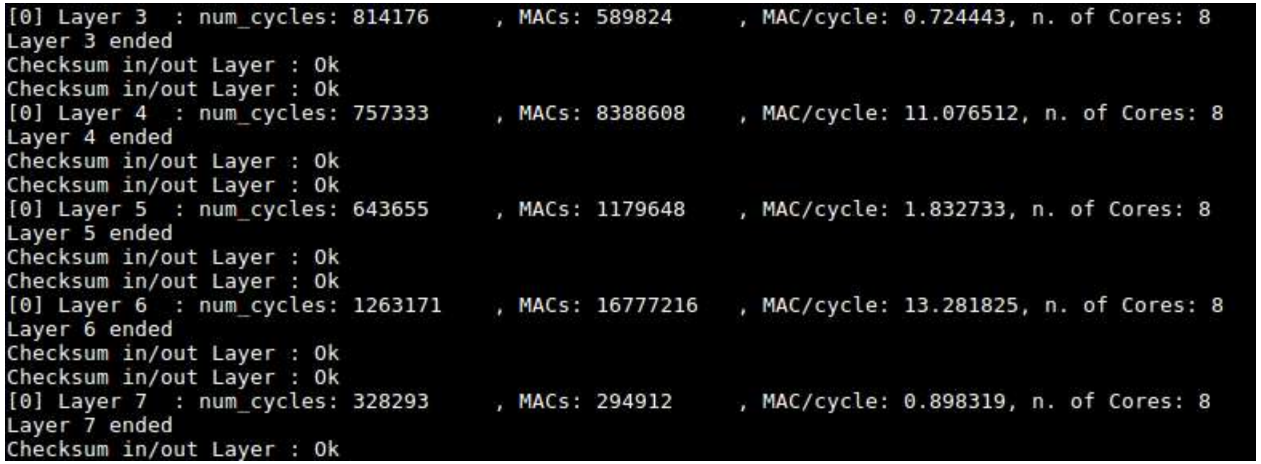
```
parser = argparse.ArgumentParser(formatter_class=RawTextHelpFormatter)
parser.add_argument('--network_dir', default = "./examples/MobilenetV1/", help = 'directory of the onnx file of the network')
parser.add_argument('--l1_buffer_size', type=int, default = 38000, help = 'L1 buffer size. IT DOES NOT INCLUDE SPACE FOR STACKS.')
parser.add_argument('--l2_buffer_size', type=int, default = 380000, help = 'L2 buffer size.')
parser.add_argument('--master_stack', type=int, default = 4096, help = 'Cluster Core 0 stack')
parser.add_argument('--slave_stack', type=int, default = 3072, help = 'Cluster Core 1-7 stack')
parser.add_argument('--Bn_Relu_Bits', type=int, default = 32, help = 'Number of bits for Relu/BN')
parser.add_argument('--perf_layer', default = 'No', help = 'Yes: MAC/cycles per layer. No: No perf per layer.')
parser.add_argument('--verbose_level', default = 'Check_all+Perf_final', help = "None: No_printf.\nPerf_final: only total performance\nCheck_
parser.add_argument('--chip', default = 'GAP8v3', help = 'GAP8v2 for fixing DMA issue. GAP8v3 otherise')
parser.add_argument('--sdk', default = 'pulp_sdk', help = 'gap_sdk or pulp_sdk')
parser.add_argument('--dma_parallelization', default = '8-cores', help = '8-cores or 1-core')
parser.add_argument('--fc_frequency', default = 100000000, help = 'frequency of fabric controller')
parser.add_argument('--cl_frequency', default = 100000000, help = 'frequency of cluster')
parser.add_argument('--optional', default = '8bit', help = '8bit, mixed, 1D_Conv')
args = parser.parse_args()
```

Change default settings

- Enable layer performance verbose

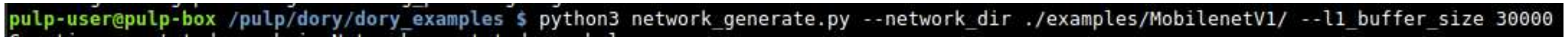


```
pulp-user@pulp-box /pulp/dory/dory_examples $ python3 network_generate.py --perf_layer Yes
```

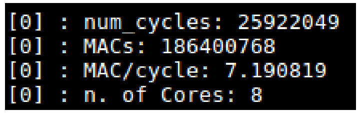


```
[0] Layer 3  : num_cycles: 814176     , MACs: 589824     , MAC/cycle: 0.724443, n. of Cores: 8
Layer 3 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
[0] Layer 4  : num_cycles: 757333     , MACs: 8388608    , MAC/cycle: 11.076512, n. of Cores: 8
Layer 4 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
[0] Layer 5  : num_cycles: 643655     , MACs: 1179648    , MAC/cycle: 1.832733, n. of Cores: 8
Layer 5 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
[0] Layer 6  : num_cycles: 1263171    , MACs: 16777216   , MAC/cycle: 13.281825, n. of Cores: 8
Layer 6 ended
Checksum in/out Layer : Ok
Checksum in/out Layer : Ok
[0] Layer 7  : num_cycles: 328293     , MACs: 294912     , MAC/cycle: 0.898319, n. of Cores: 8
Layer 7 ended
Checksum in/out Layer : Ok
```

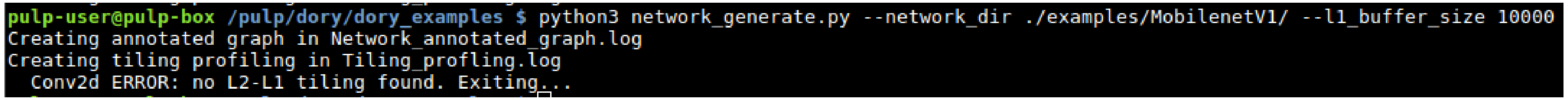
- Change L1 maximum memory footprint



```
pulp-user@pulp-box /pulp/dory/dory_examples $ python3 network_generate.py --network_dir ./examples/MobilenetV1/ --l1_buffer_size 30000
```

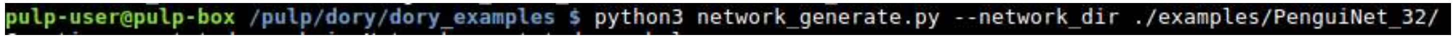


```
[0] : num_cycles: 25922049
[0] : MACs: 186400768
[0] : MAC/cycle: 7.190819
[0] : n. of Cores: 8
```



```
pulp-user@pulp-box /pulp/dory/dory_examples $ python3 network_generate.py --network_dir ./examples/MobilenetV1/ --l1_buffer_size 10000
Creating annotated graph in Network_annotated_graph.log
Creating tiling profiling in Tiling_profling.log
  Conv2d ERROR: no L2-L1 tiling found. Exiting...
```

- Generate a new network



```
pulp-user@pulp-box /pulp/dory/dory_examples $ python3 network_generate.py --network_dir ./examples/PenguiNet_32/
```

Thanks for the attention