ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

LAB07: Tiling on PULP part2

Lorenzo Lamberti – lorenzo.lamberti@unibo.it

Luka Macan – luka.macan@unibo.it

Alessio Burrello – alessio.burrello@unibo.it

Nazareno Bruschi – nazareno.bruschi@unibo.it

Francesco Conti – f.conti@unibo.it

Objective of the Class

Intro: Tiling

Tasks:

- Double Buffering
- Overlapping tiles with 3x3 convolutions

Programming Language: C

Lab duration: 2.5h

Assignment:

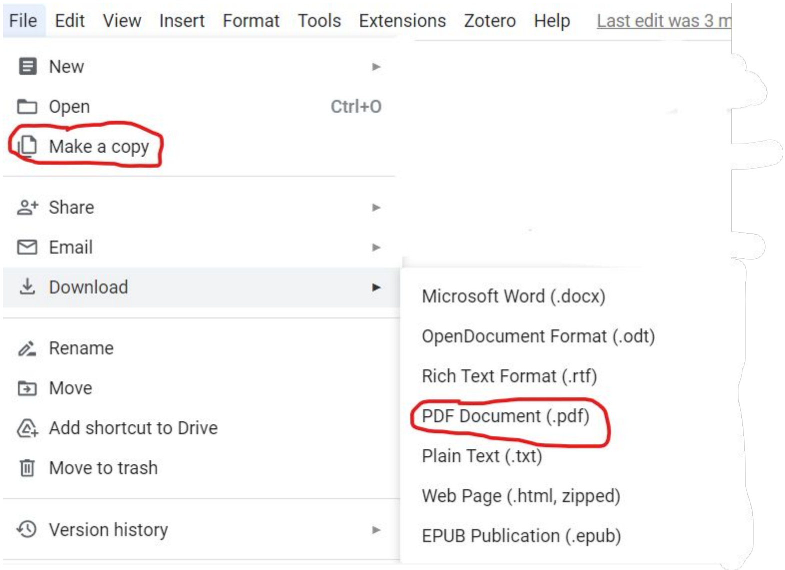
- Time for delivery: 1 week
- Submission deadline: Nov 30th 2023 (16:00)

The class is meant to be interactive: coding together, on your own, and do not be afraid to ask questions!

How to deliver the Assignment

You will deliver ONLY the GDOC assignment, no code

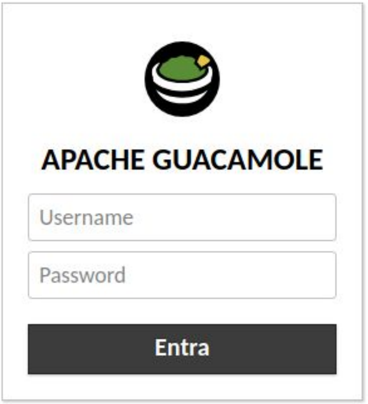
- Copy the google doc to your drive, so that you can modify it. (File -> make a copy)
- Fill the tasks on this google doc.
- Export to pdf format.
- Rename the file to: LAB<number_of_the_lesson>_APAI_<your_name>.pdf
- Use Virtuale platform to load ONLY your .pdf file



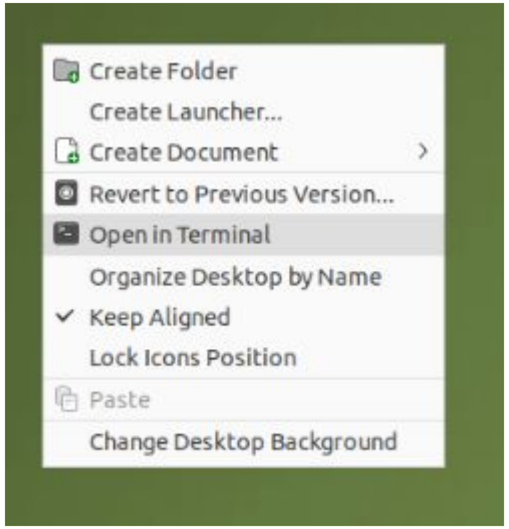
SETUP: How to access the server

1. Open this web page: https://compute.eees.dei.unibo.it:8443/guacamole/ **(works only from ALMA WIFI NETWORK!)**

2. Login. We distribute credentials by hand.



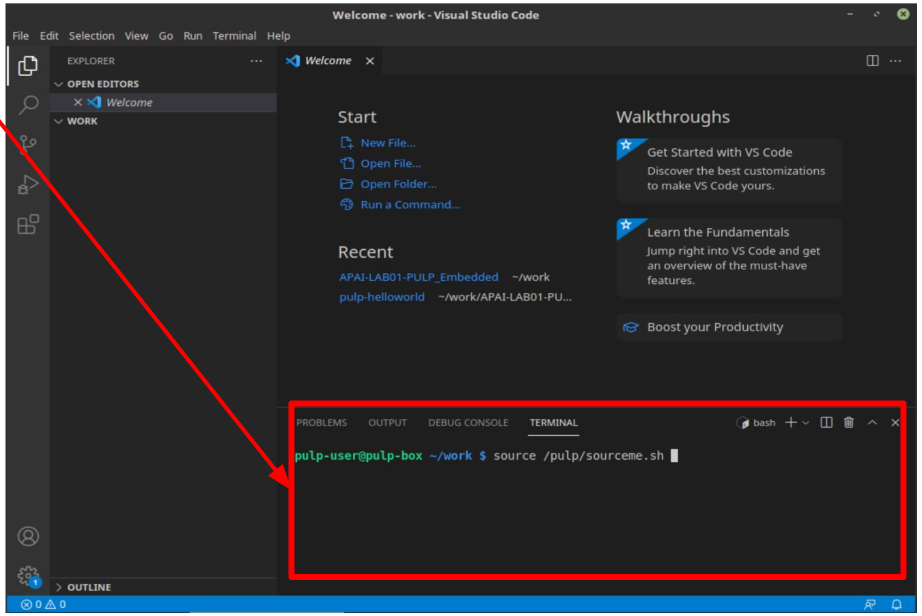
3. Open a terminal (right click – open a new terminal)



4. Open a text editor (For example "VSCode"):

```
$ code .
```

Now you can use the integrated terminal to run your applications!


Welcome - work - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
Welcome
WORK
Start
New File...
Open File...
Open Folder...
Run a Command...
Recent
APAI-LAB01-PULP_Embedded ~/work
pulp-helloworld ~/work/APAI-LAB01-PU...
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Boost your Productivity
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash
OUTLINE

```
pulp-user@pulp-box ~/work $ source /pulp/sourceme.sh
```

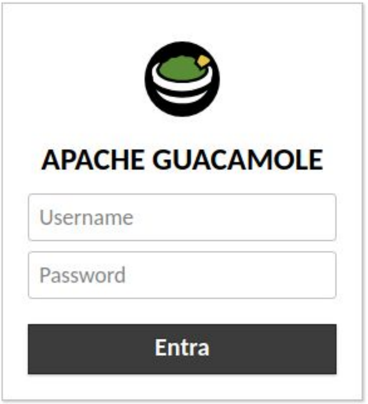
IMPORTANT: activate the pulp-sdk module file every time a new shell is open.

```
$ module load pulp-sdk
$ module load dory-conda
```

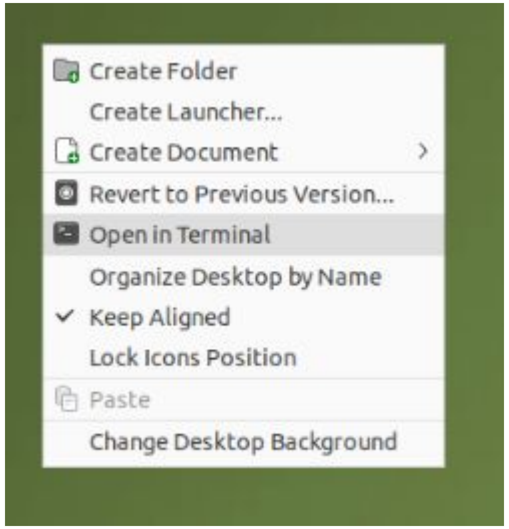
SETUP: How to access the server

1. Open this web page: https://compute.eees.dei.unibo.it:8443/guacamole/
   (works only from ALMA WIFI NETWORK!)

2. Login. We distribute credentials by hand.



3. Open a terminal (right click – open a new terminal)



4. Clone:
   `git clone https://github.com/EEESlab/<insert_here_the_right_repo!>`

5. `module load pulp-sdk`

6. `module load dory-conda`

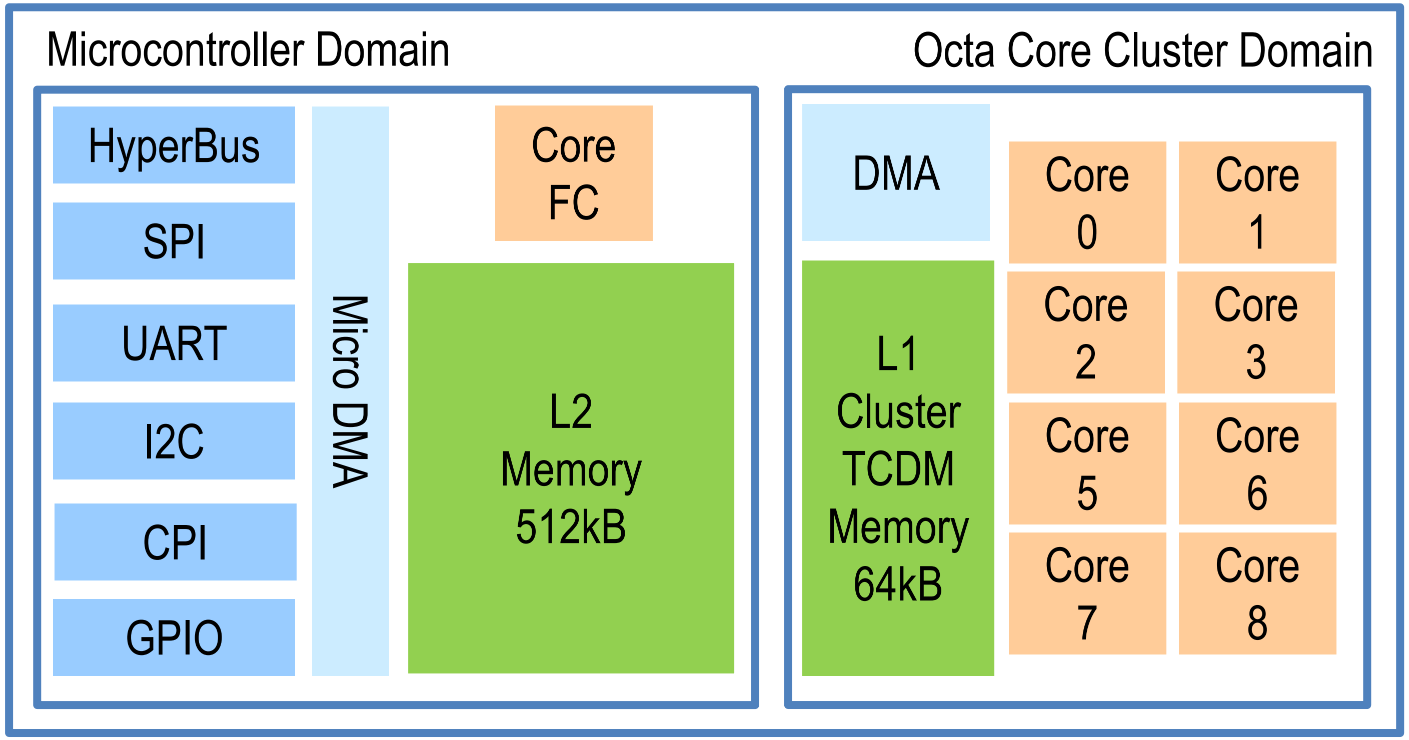
7. `cd <insert_here_the_right_repo!>`

8. `make clean all run`

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

TASK4: double buffering

PULP Platform: today we focus on the 8-cores cluster

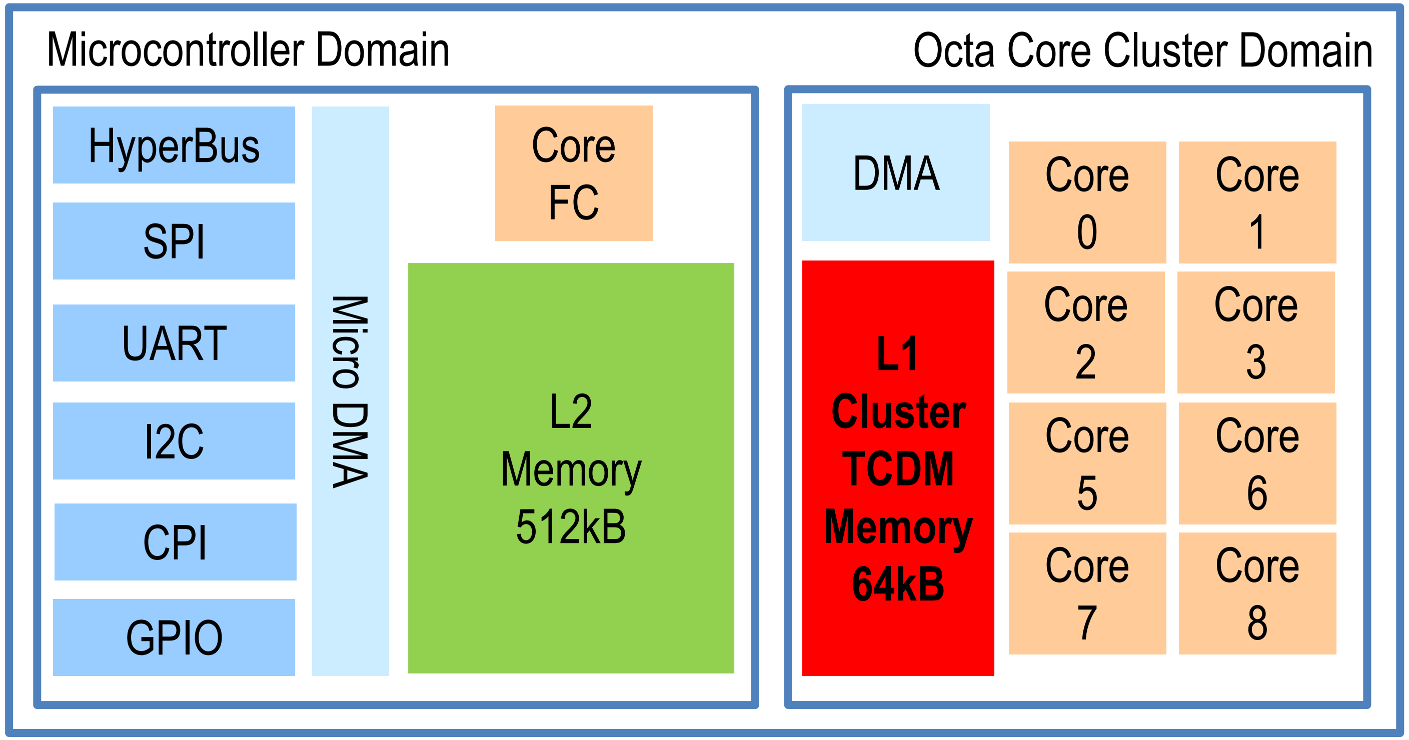


- **Cores**: 1 + 8
- **On-chip Memories**
  - A level 2 Memory, shared among all cores
  - A level 1 Memory, shared by the 8-cores cluster
- **cluster-DMA:** A multi-channel 1D/2D DMA, controlling the transactions between the L2 and L1 memories
- **micro-DMA**: A smart, lightweight and completely autonomous DMA () capable of handling complex I/O scheme
  - **Bus+Peripherals:** HyperBus, I2S, CPI, timers, SPI, GPIOs, etc…

NB: this is the architecture you find on the nano-drone!

GitHub HW Project: https://github.com/pulp-platform/pulp
HW Documentation:
https://raw.githubusercontent.com/pulp-platform/pulp/master/doc/datasheet.pdf

PULP Platform: today we focus on the 8-cores cluster

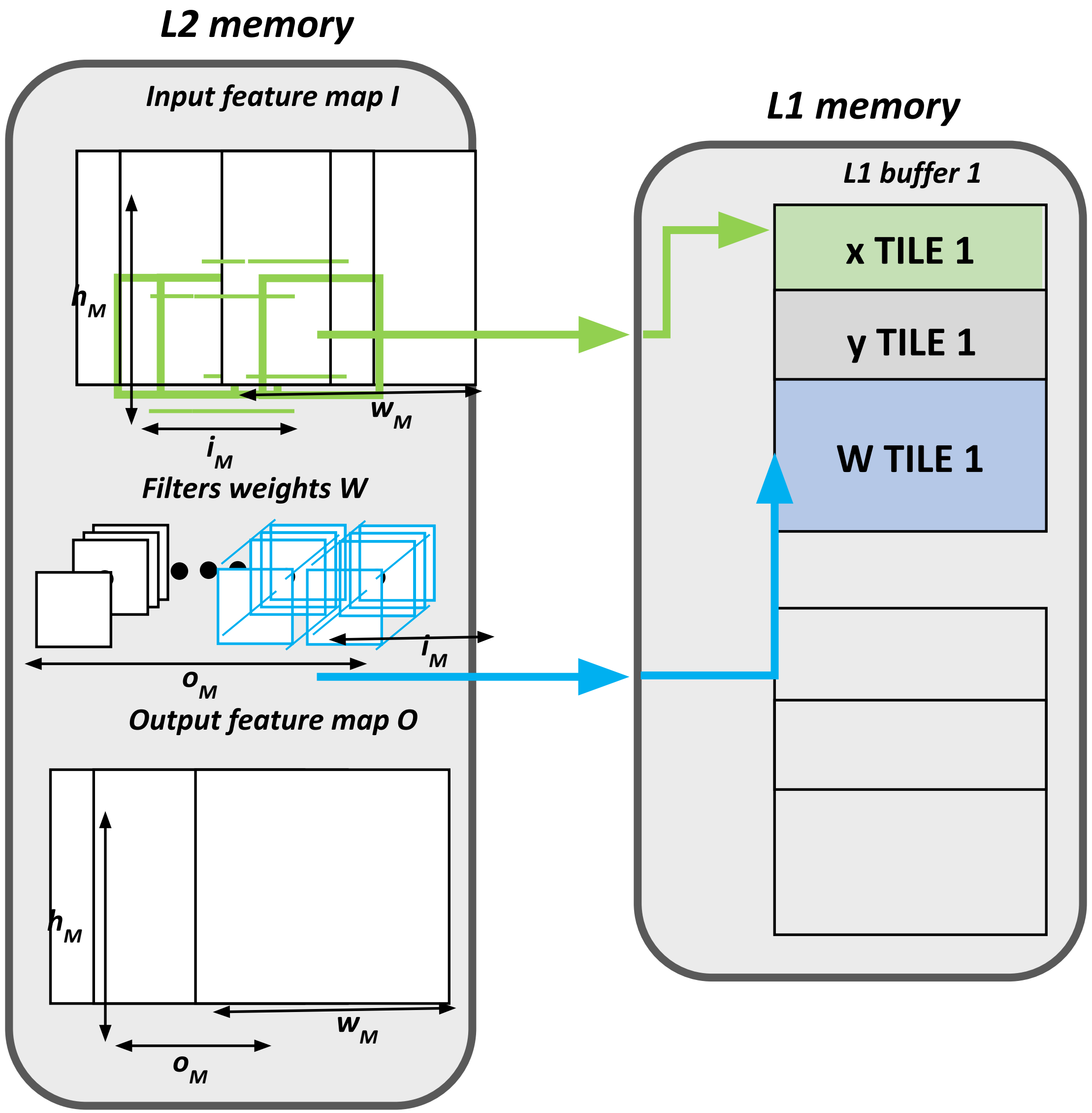


- **Cores**: 1 + 8
- **On-chip Memories**
  - A level 2 Memory, shared among all cores
  - A level 1 Memory, shared by the 8-cores cluster
- **cluster-DMA:** A multi-channel 1D/2D DMA, controlling the transactions between the L2 and L1 memories
- **micro-DMA**: A smart, lightweight and completely autonomous DMA () capable of handling complex I/O scheme
  - **Bus+Peripherals:** HyperBus, I2S, CPI, timers, SPI, GPIOs, etc…

NB: this is the architecture you find on the nano-drone!

GitHub HW Project: https://github.com/pulp-platform/pulp
HW Documentation:
https://raw.githubusercontent.com/pulp-platform/pulp/master/doc/datasheet.pdf

LAB05: Tiling from L2 to L1 <u>with no double buffering</u>



L2 memory

Input feature map I

h_M

w_M

i_M

Filters weights W

i_M

o_M

Output feature map O

h_M

w_M

o_M

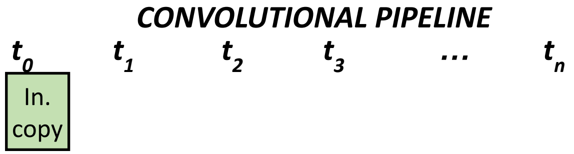
L1 memory

L1 buffer 1

x TILE 1

y TILE 1

W TILE 1



CONVOLUTIONAL PIPELINE

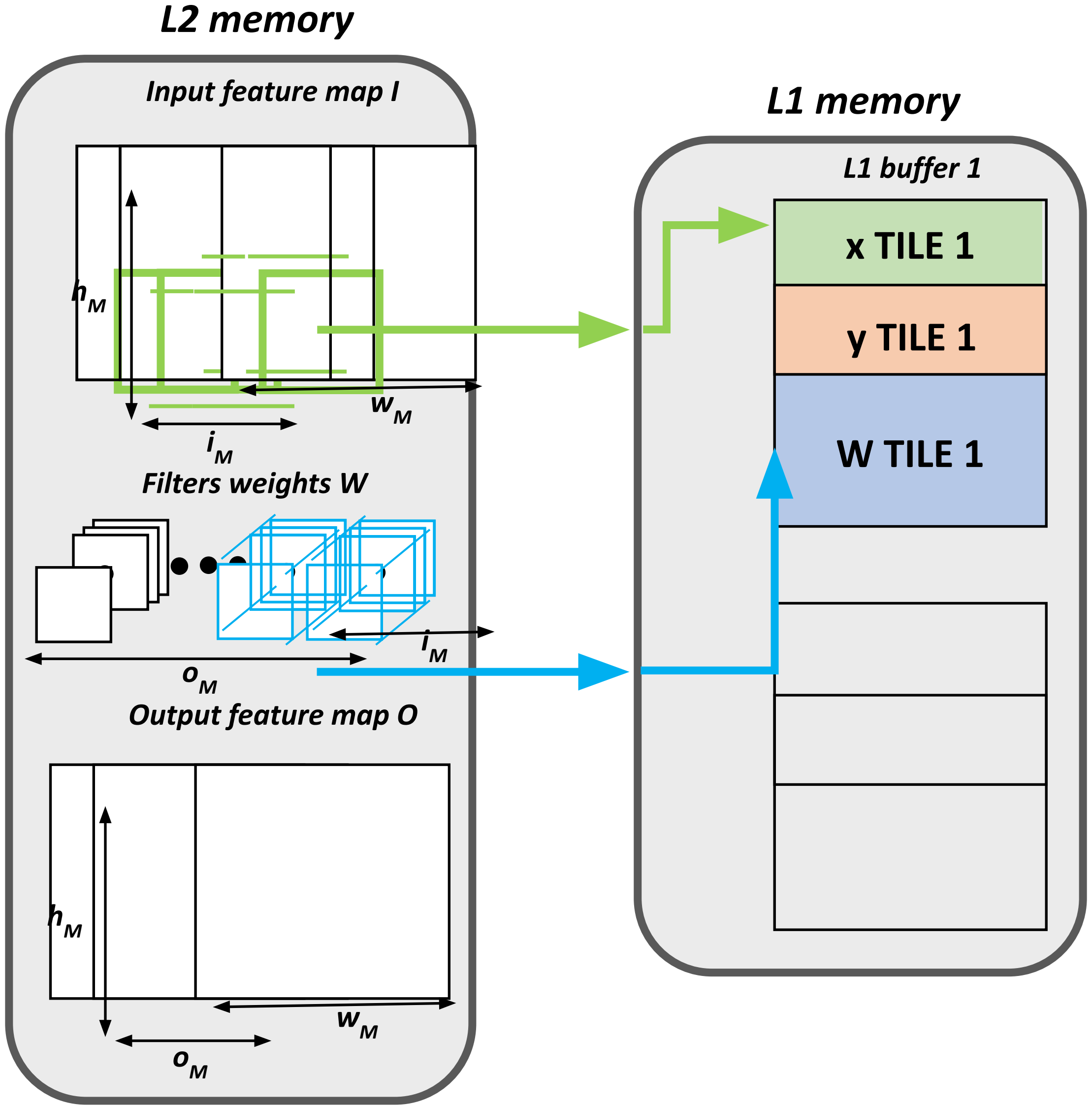
t_0 t_1 t_2 t_3 … t_n

In. copy

🟩 DMA ch. 0-1 🟧 DMA ch. 2

🟦 Cluster computation

LAB05: Tiling from L2 to L1 with no double buffering



L2 memory

Input feature map I

h_M

w_M

i_M

Filters weights W

i_M

o_M

Output feature map O

h_M

w_M

o_M

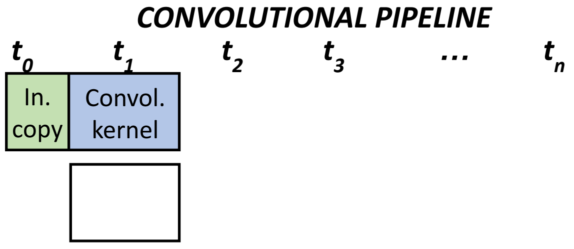
L1 memory

L1 buffer 1

x TILE 1

y TILE 1

W TILE 1



CONVOLUTIONAL PIPELINE

t_0 t_1 t_2 t_3 … t_n

In. copy

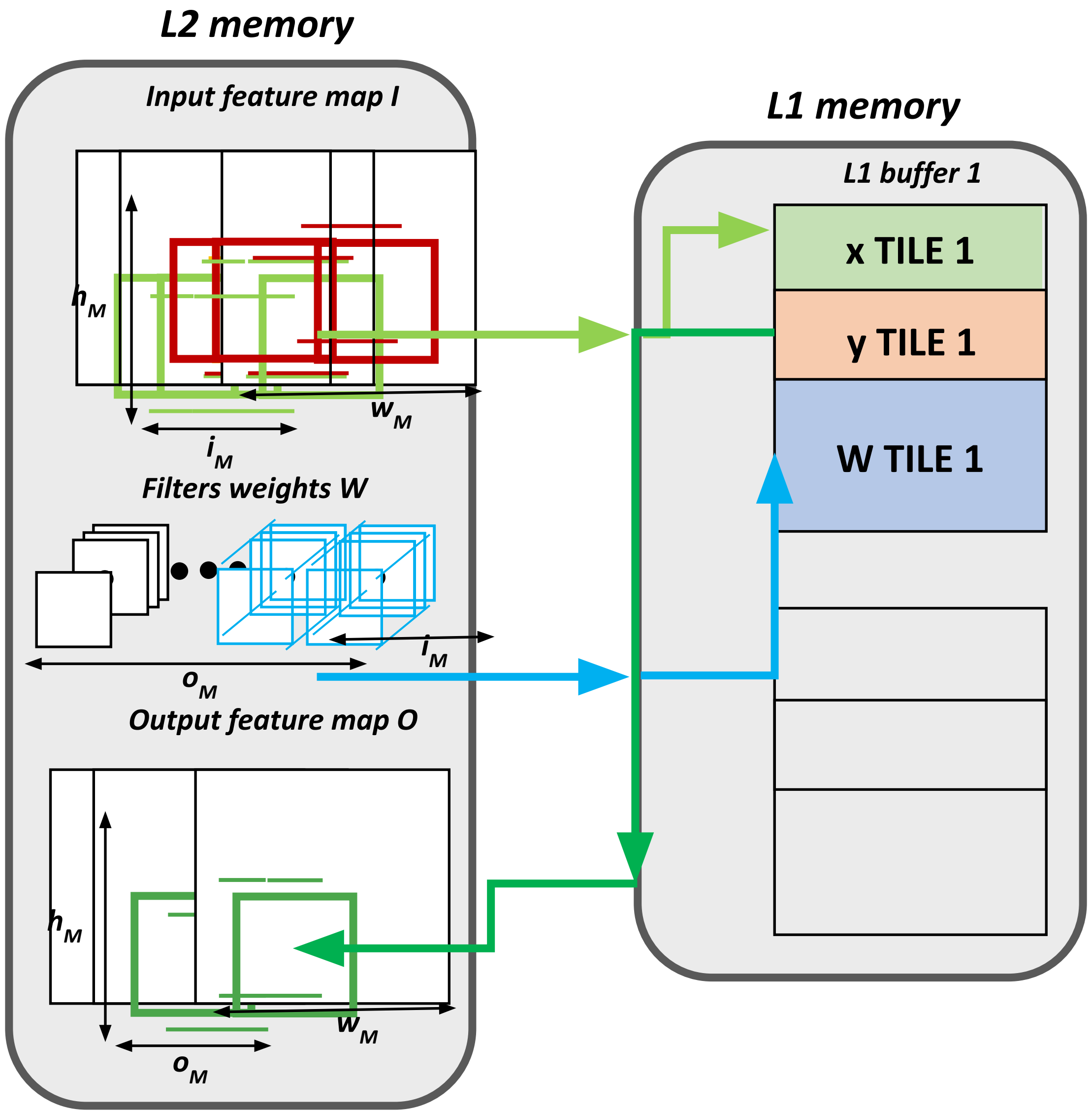
Convol. kernel

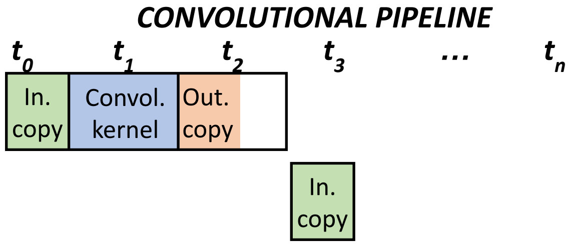
🟩 *DMA ch. 0-1*

🟧 *DMA ch. 2*

🟦 *Cluster computation*

LAB05: Tiling from L2 to L1 with no double buffering



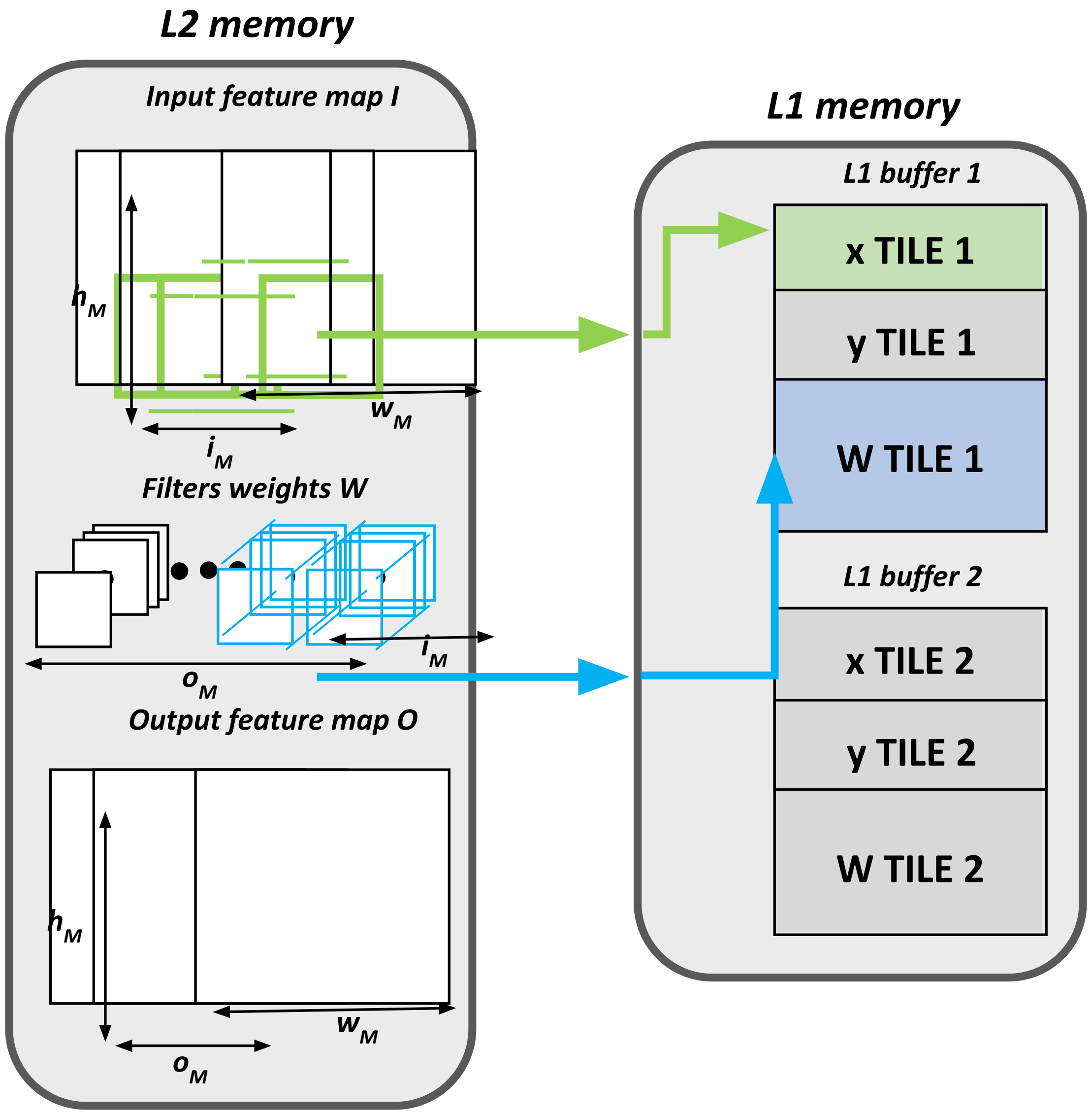


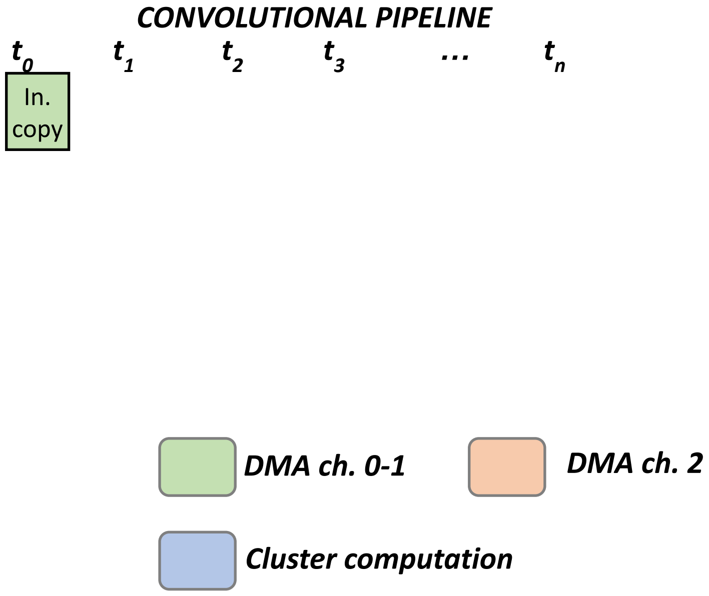
🟩 *DMA ch. 0-1* 🟧 *DMA ch. 2*

🟦 *Cluster computation*

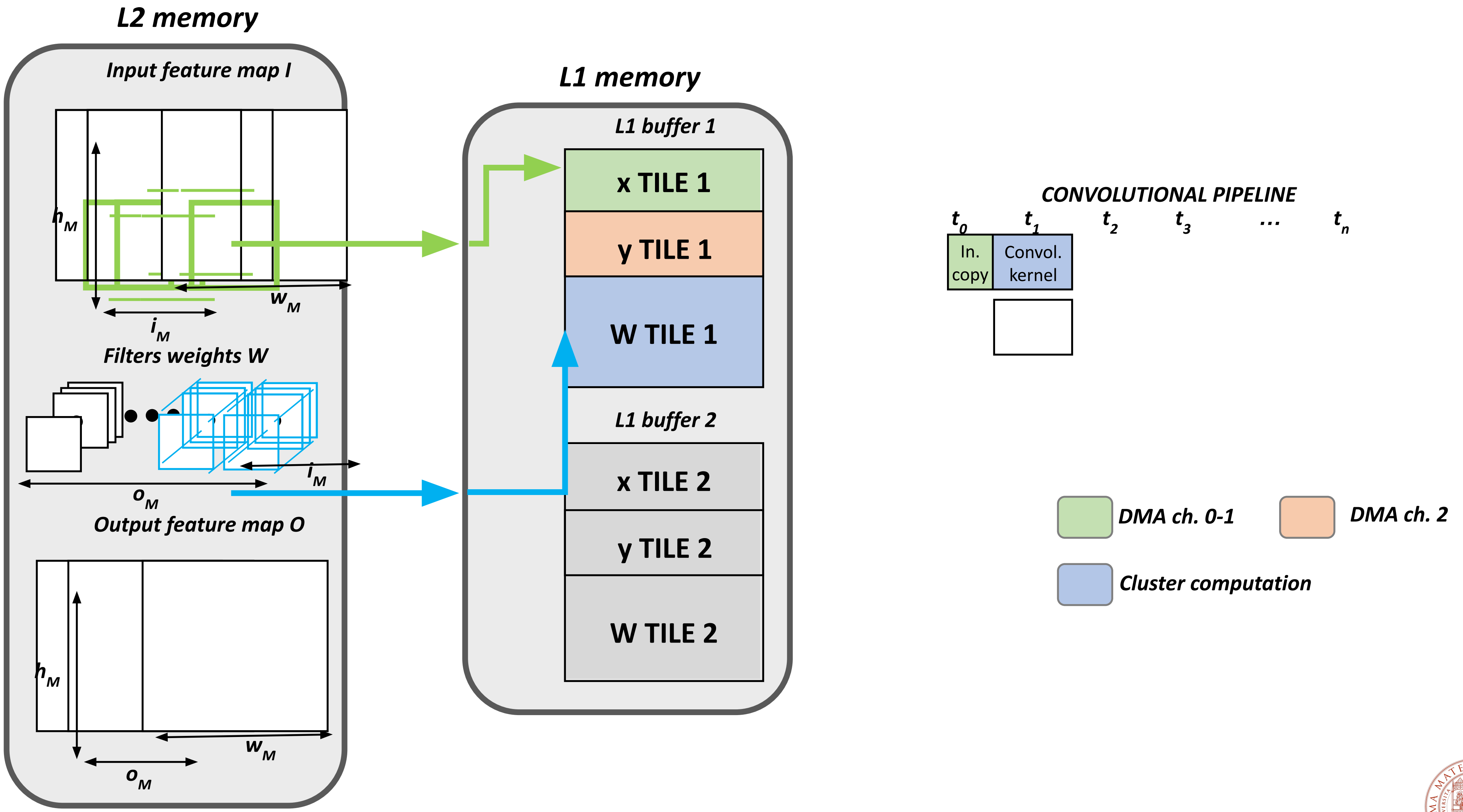
LAB06: Tiling from L2 to L1 with double buffering

LAB06: Tiling from L2 to L1 with double buffering





LAB06: Tiling from L2 to L1 with double buffering



L2 memory

Input feature map I

h_M

w_M

i_M

Filters weights W

i_M

o_M

Output feature map O

h_M

w_M

o_M

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

t_0 t_1 t_2 t_3 … t_n

In. copy

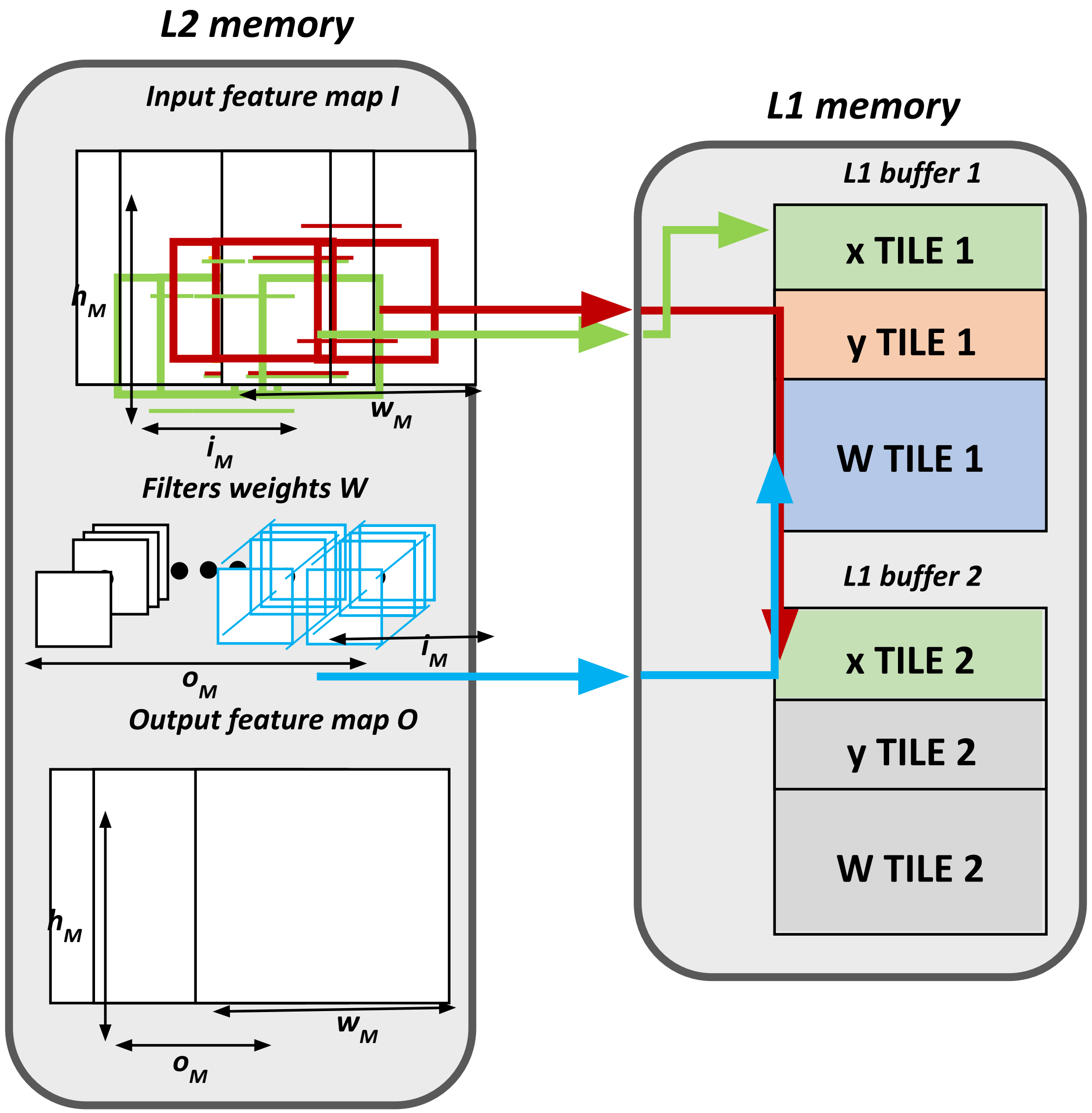
Convol. kernel

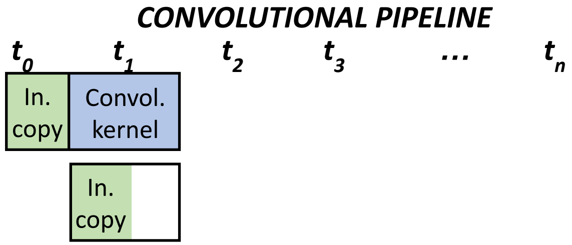
🟩 DMA ch. 0-1

🟧 DMA ch. 2

🟦 Cluster computation

LAB06: Tiling from L2 to L1 <u>with double buffering</u>



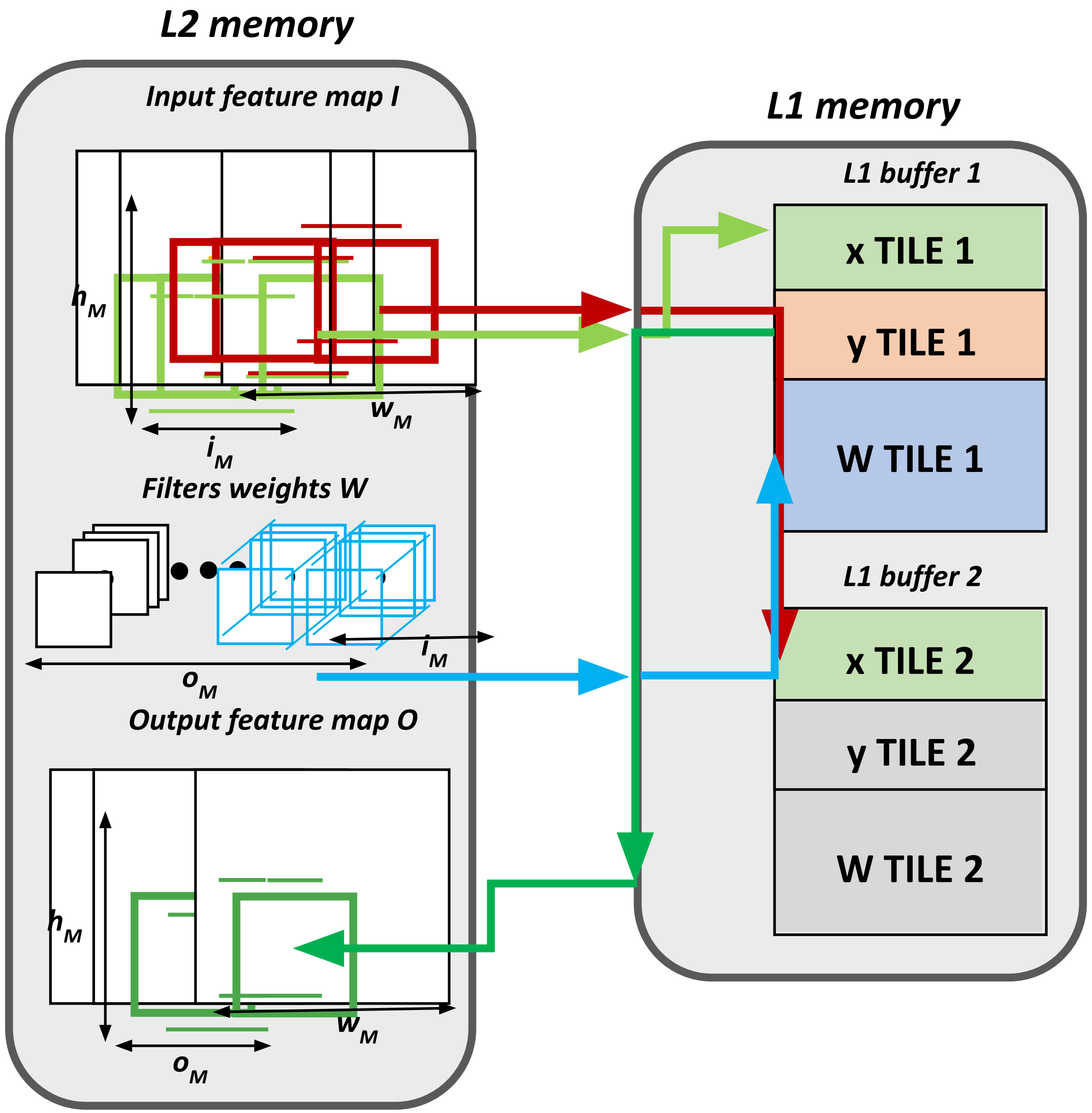


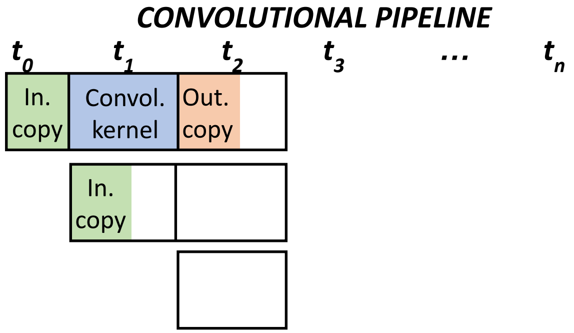
🟩 *DMA ch. 0-1*

🟧 *DMA ch. 2*

🟦 *Cluster computation*

LAB06: Tiling from L2 to L1 with double buffering



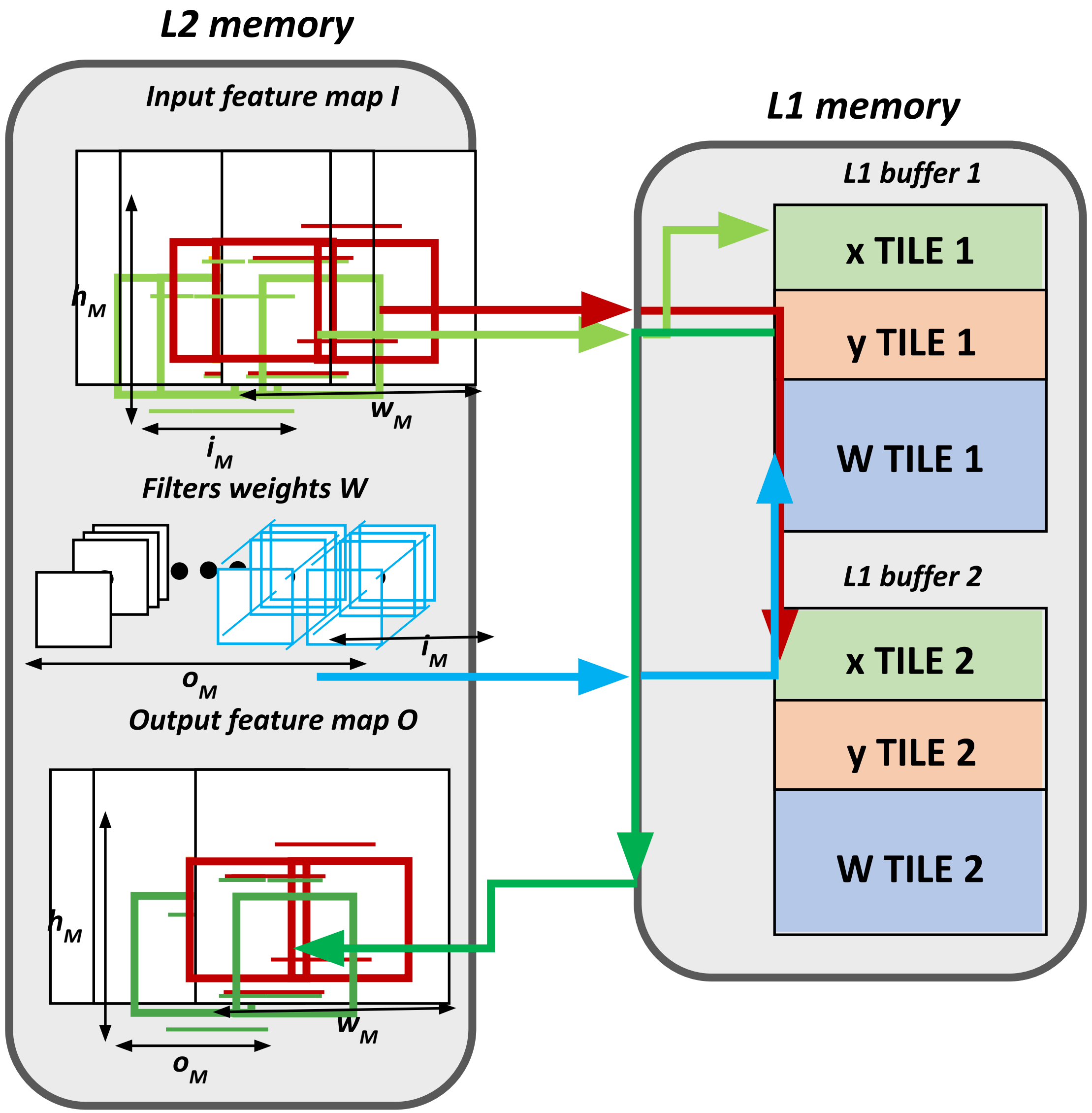


🟩 DMA ch. 0-1

🟧 DMA ch. 2

🟦 Cluster computation

LAB06: Tiling from L2 to L1 with double buffering



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*

EX4: find maximum dimensions of layers fitting L1 without tiling

Prerequisites:

```
module load pulp-sdk
module load dory-conda
```

Run the code:

```
 1.   python3 parameters_generate.py --kernel-shape=<add_here> --channels=<add_here> --output-spatial_dimensions=<add_here>
 2.   make clean all run
```

Follow the assignment document.

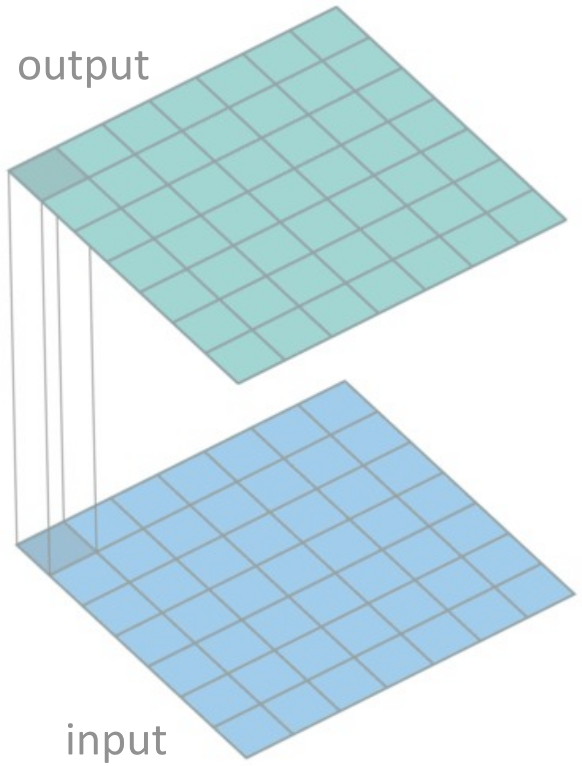
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

TASK5: conv3x3 and overlapping tiles

Case study: 3x3 conv2D

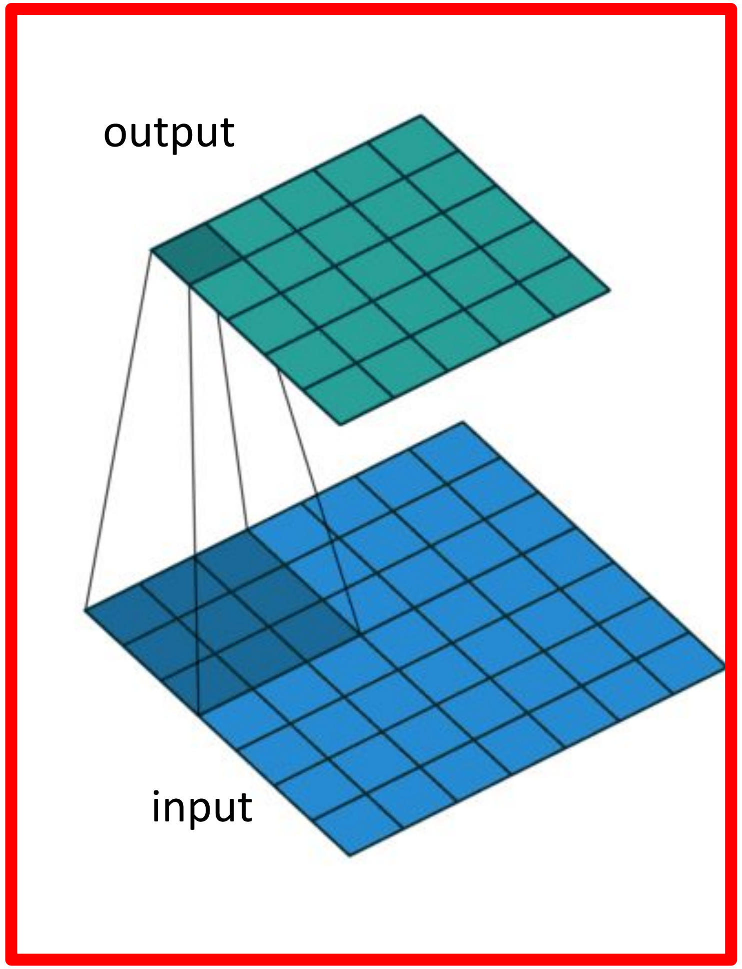
Task5.1: cov3x3

- conv1x1 the spatial size between input and output does not change!
- With conv3x3 it changes. Find out how!



1x1 Convolution

lab05!



3x3 Convolution

Used in lab06!

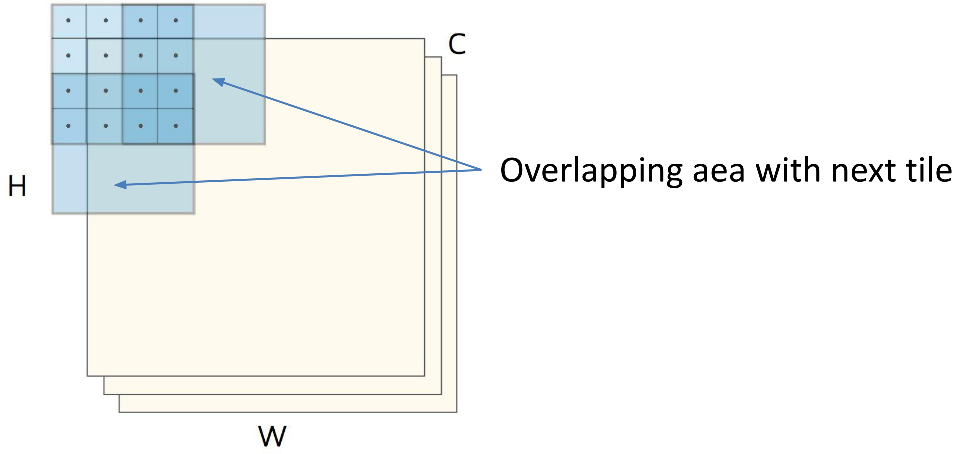
Case study: 3x3 conv2D

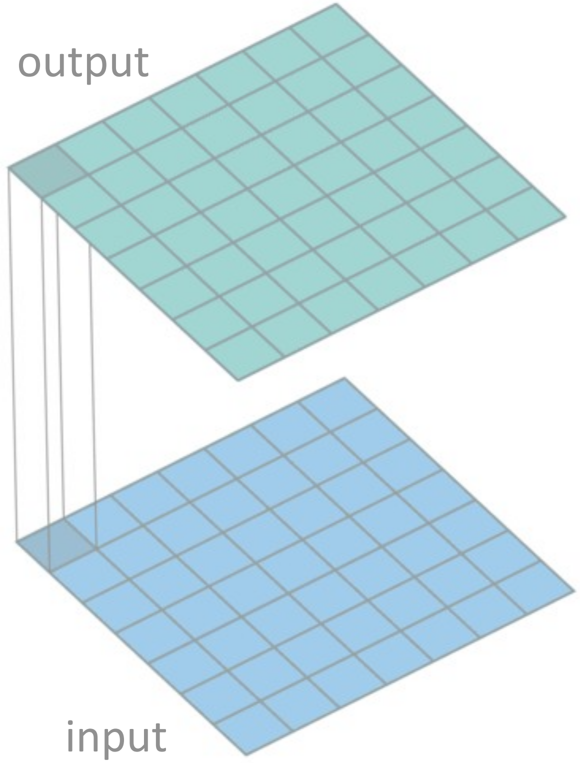
Task5.1: cov3x3

- conv1x1 the spatial size between input and output does not change!
- With conv3x3 it changes. Find out how!

Task5.2: overlapping tiles

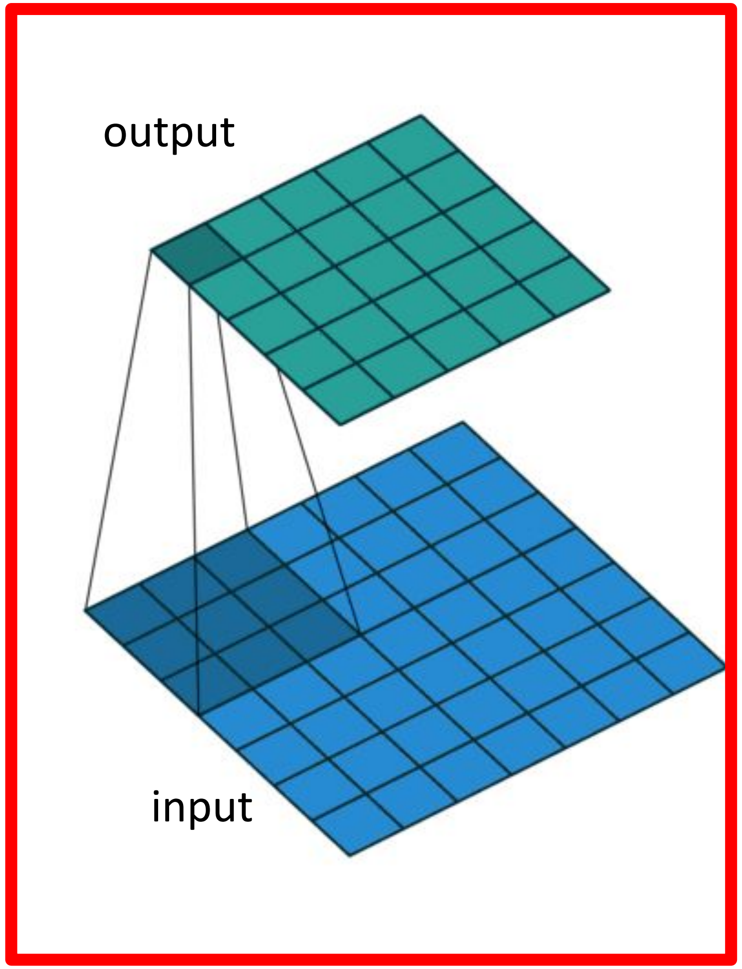
With 3x3 convolutions adjacent tiles overlap a bit, so we load the same piece of input twice. Implement the right overlapping factor!





1x1 Convolution

lab05!



3x3 Convolution

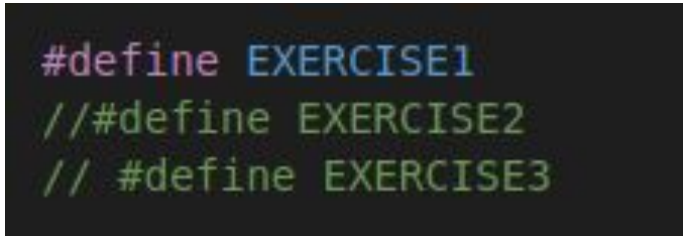
Used in lab06!

EX3: Tiling layer

```
Run the code:
$ python3 parameters_generate.py --channels=#### --spatial_dimension=####
$ make clean all run
```

Follow the assignment document.

NB: Choose the exercise by uncommenting one of the following defines in `main.h`:



```
#define EXERCISE1
//#define EXERCISE2
// #define EXERCISE3
```

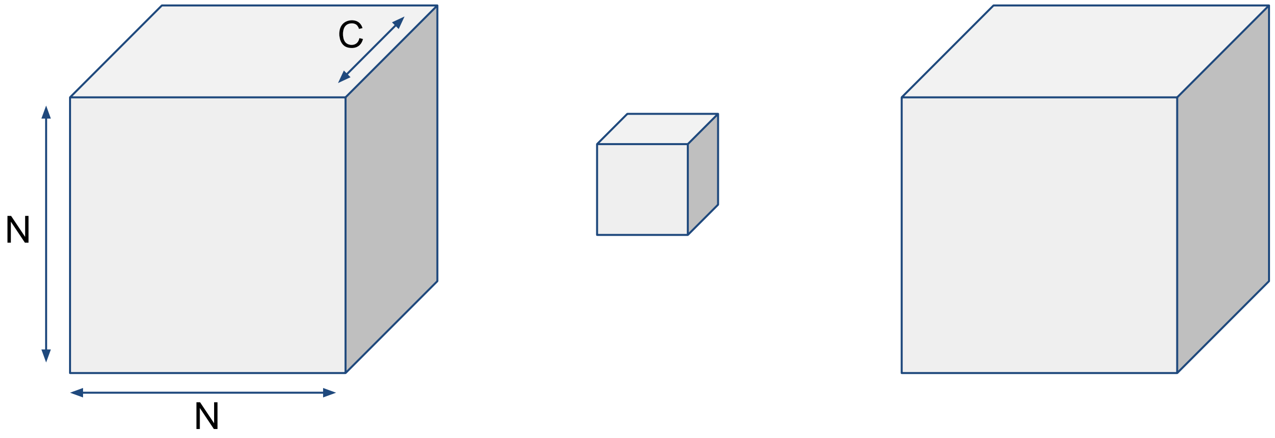
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

BACKUP

DEI – Università di Bologna

WHAT'S SPATIAL DIMENSION?

We assume width = height
Spatial dimension (N) = width (W) = Height (H)



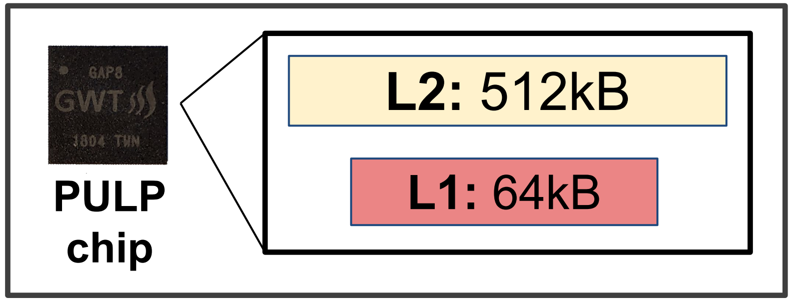
Case study: 1x1 conv2D

We tackle a 1x1 convolution with this sizes:

- Input = SPATIAL_DIM → defined by you
- Output = SPATIAL_DIM → defined by you
- Kernel = 1x1
- Stride = 1
- Padding = 0

NB: with conv1x1 the spatial size between input and output does not change!

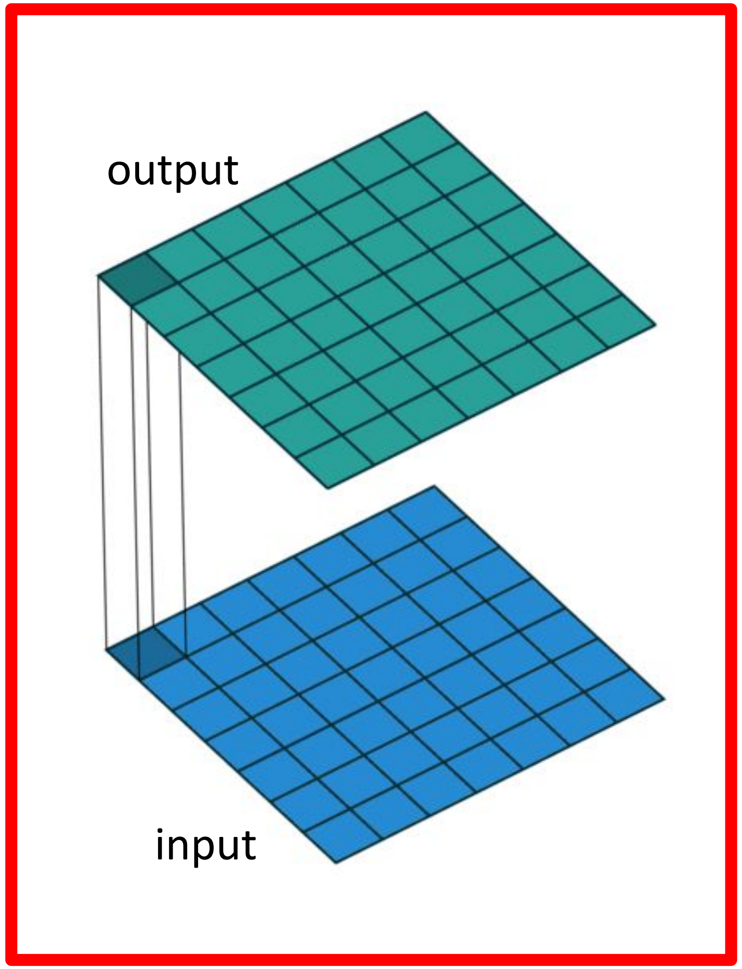
We want to fit into the L1 memory!



PULP chip

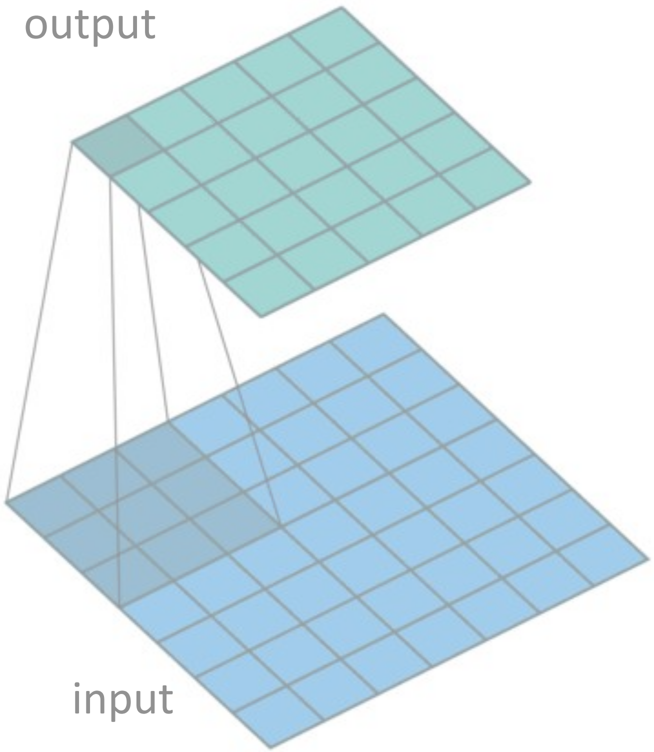
L2: 512kB

L1: 64kB



1x1 Convolution

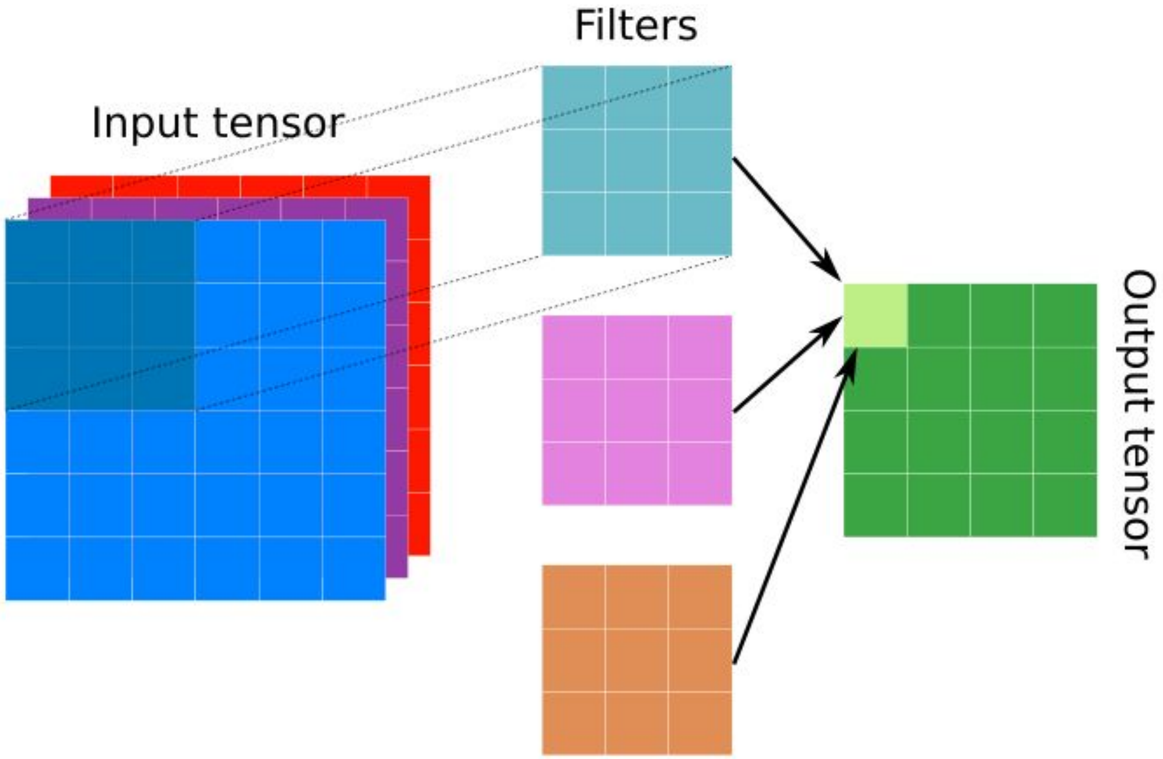
Used today!



3x3 Convolution

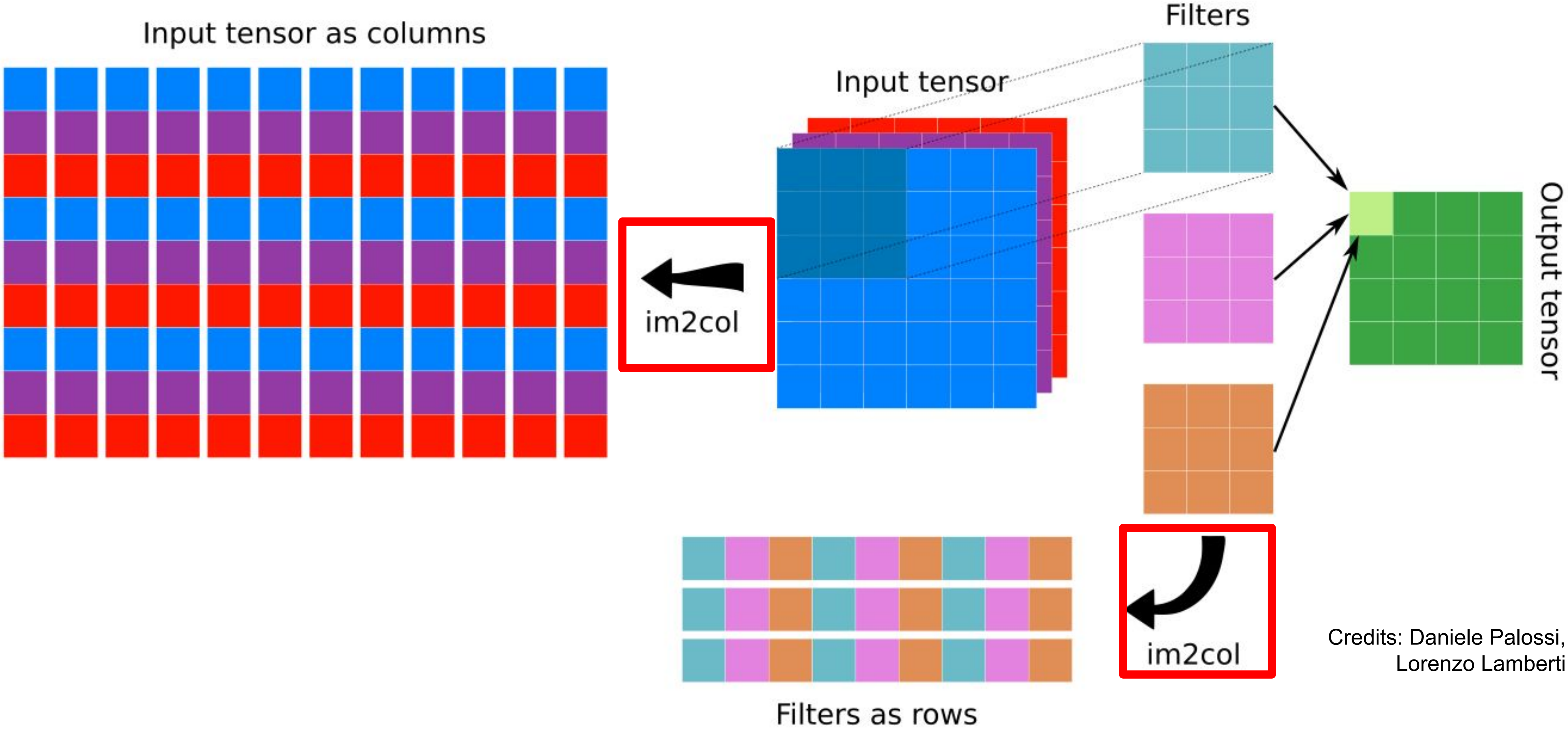
Used in lab04!

Convolution Operation: naive



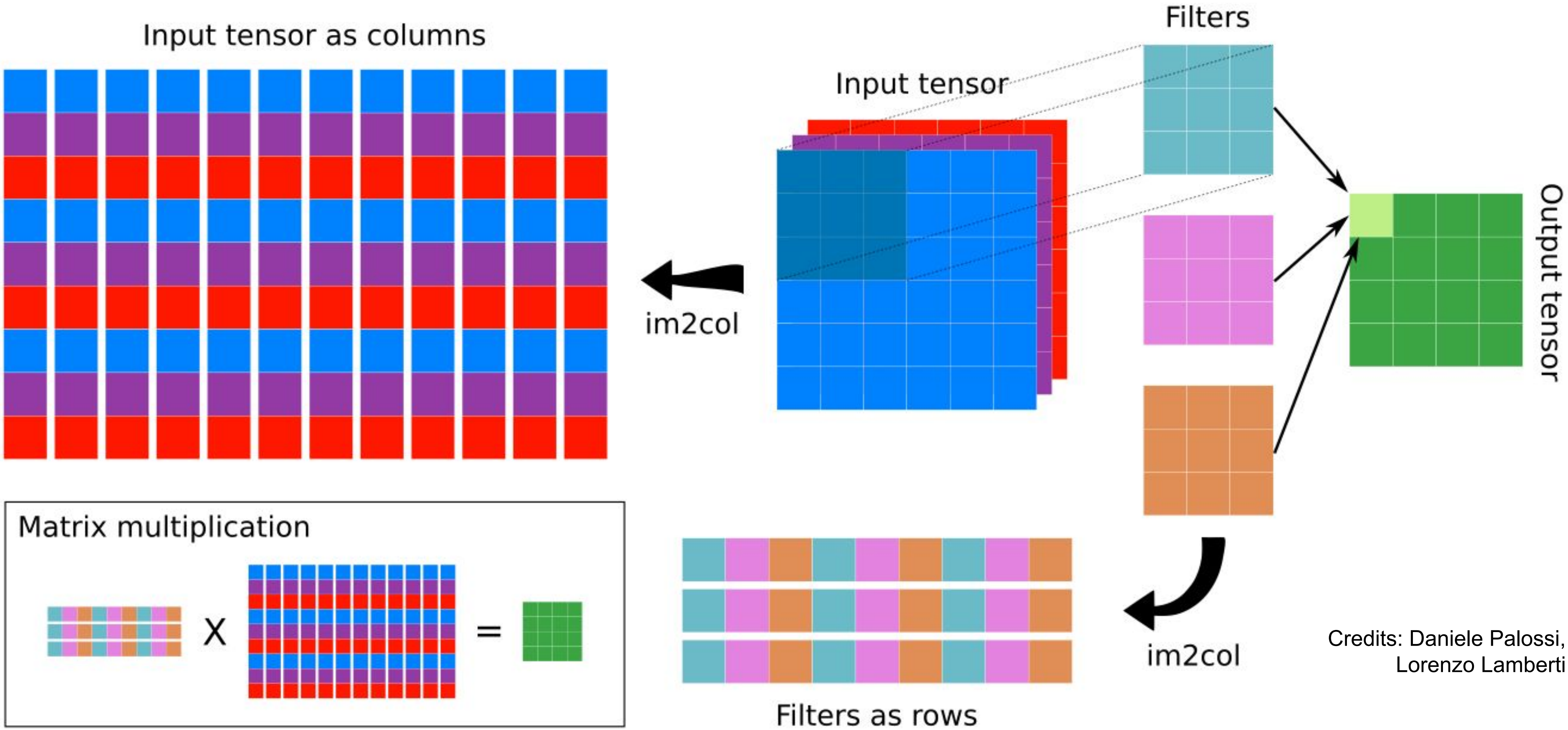
Credits: Daniele Palossi,
Lorenzo Lamberti

Convolution Operation: im2col and MatMul



Credits: Daniele Palossi, Lorenzo Lamberti

Convolution Operation: im2col and MatMul



Credits: Daniele Palossi,
Lorenzo Lamberti