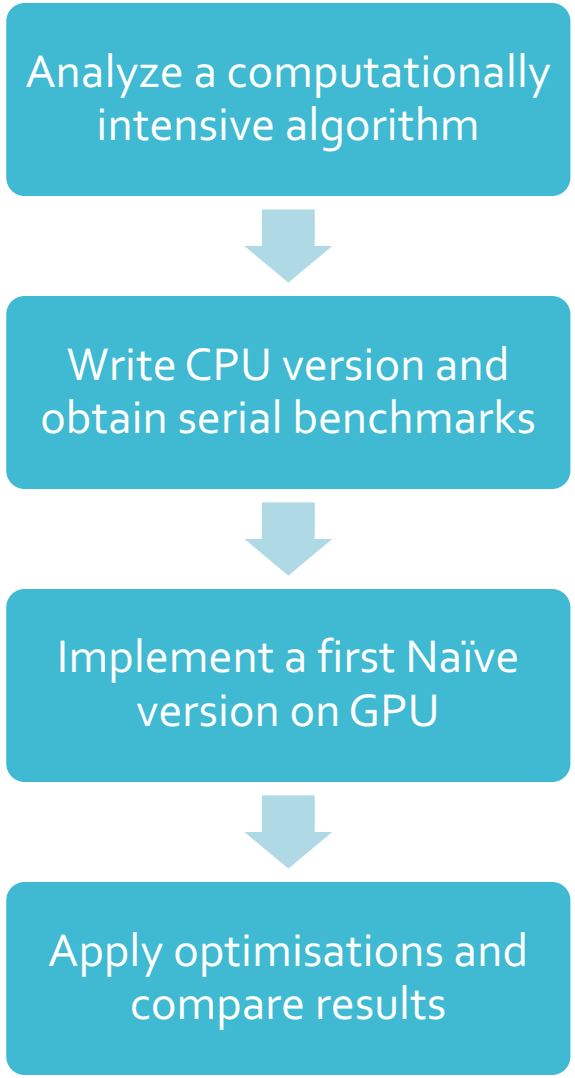
LeNet-1 Convolutional Neural Network acceleration in CUDA

Computer Engineering master degree, Politecnico di Torino, GPU programming course.

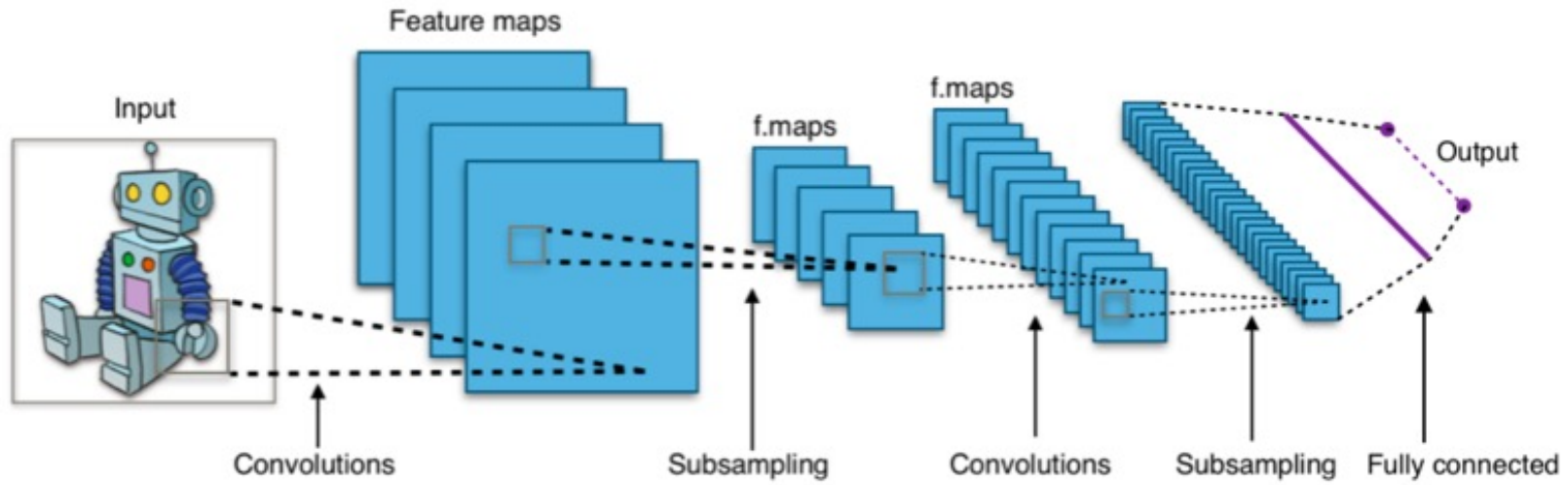
Presenter: Dalmasso Luca

Project objectives

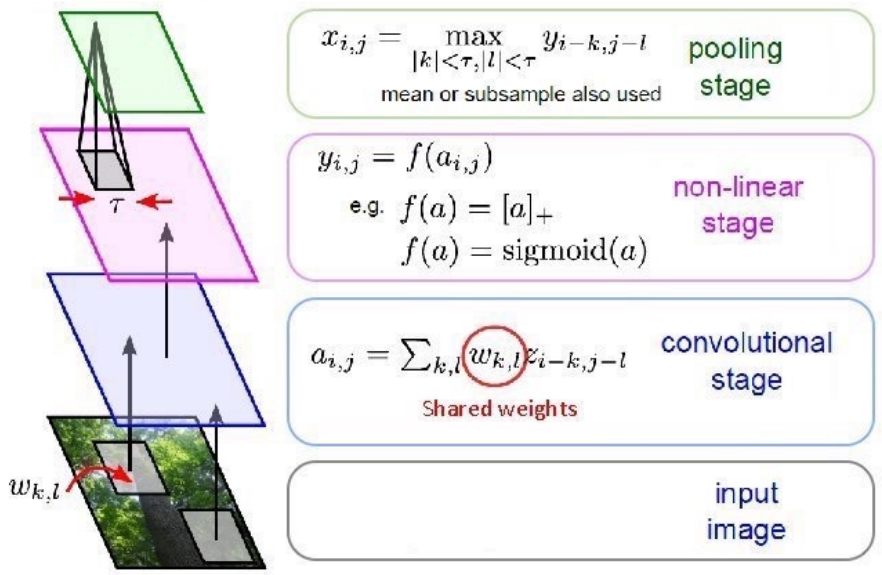


CNNs in a nutshell

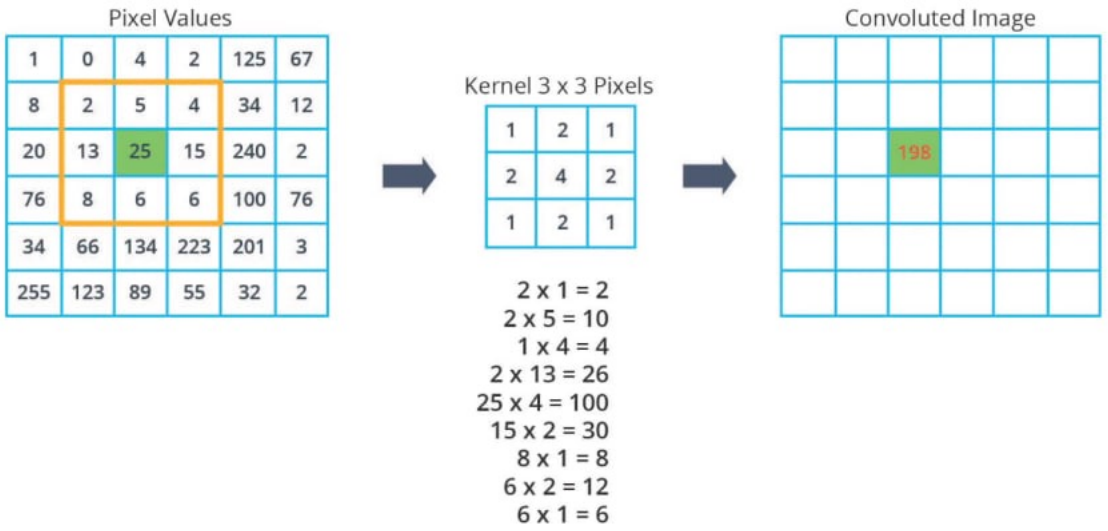
- CNNs are a class of Deep Learning algorithms widely used for image classification.
- A CNN is a list of layers that transform the input data into an output prediction (classification)
- Layers can be of 3 types: Convolutional layer, Non-linear layer and Pooling layer. These are the building-blocks of any CNN



Building-block of a CNN

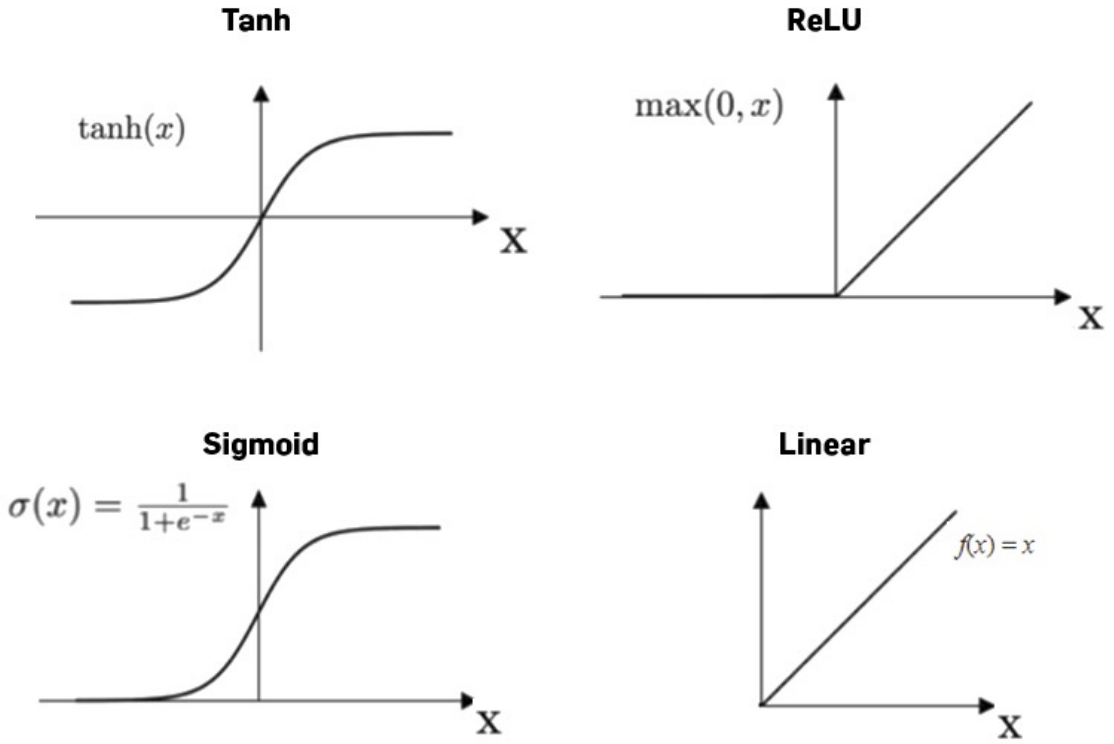


Convolutional Stage

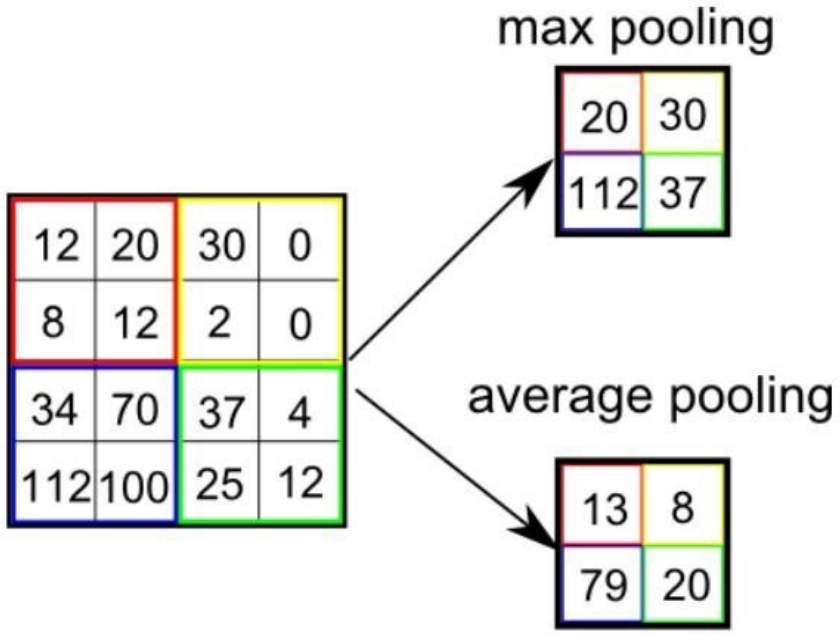


2 + 10 + 4 + 26 + 100 + 30 + 8 + 12 + 6 = 198

Activation functions in the Non-linear layer



Pooling or Subsampling layer



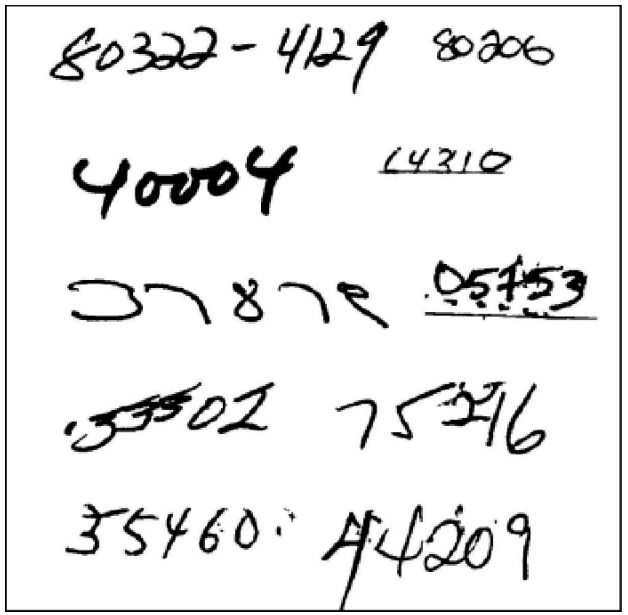
Repeating the mathematical steps presented in the building block for each neuron (pixel) in each feature maps in the network is a computationally intensive task.

On the CPU the neurons are processed in series

On the GPU all neurons in a given layer can be processed in parallel by independent threads!

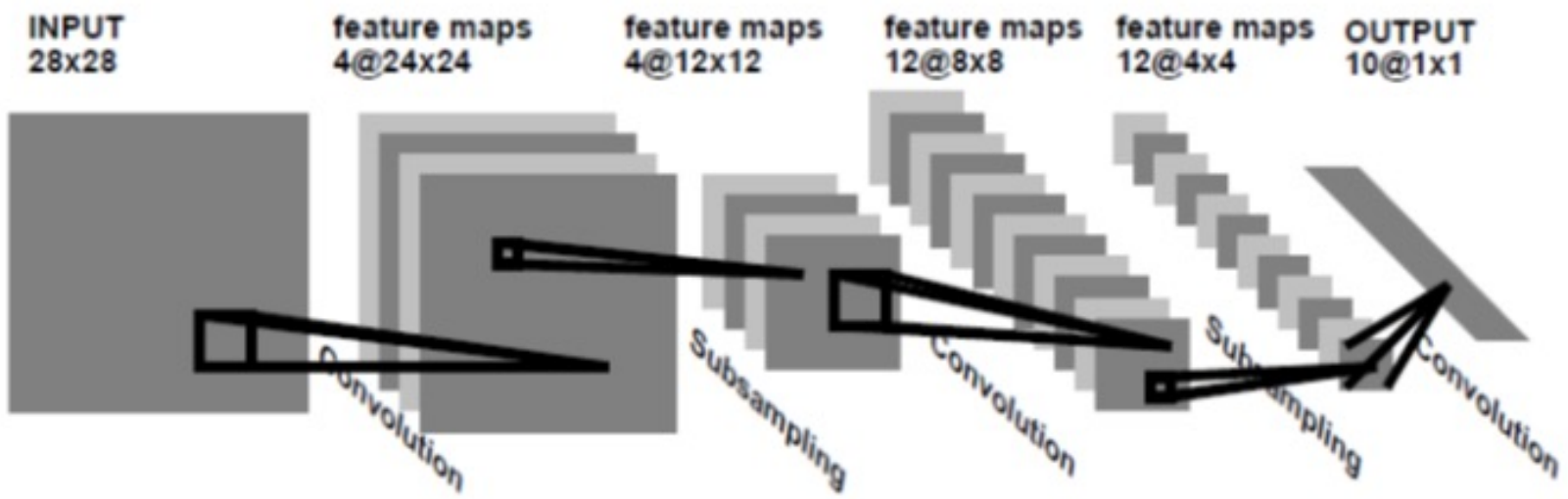
The LeNet-1 CNN, a pioneering work in Deep Learning

- LeNet-1 is one of the earliest Convolutional Neural Network, proposed in 1998 by Yann LeCun.
- Attempt to classify and recognise ZIP code images



LeNet-1 Architecture

LeNet-1 CNN

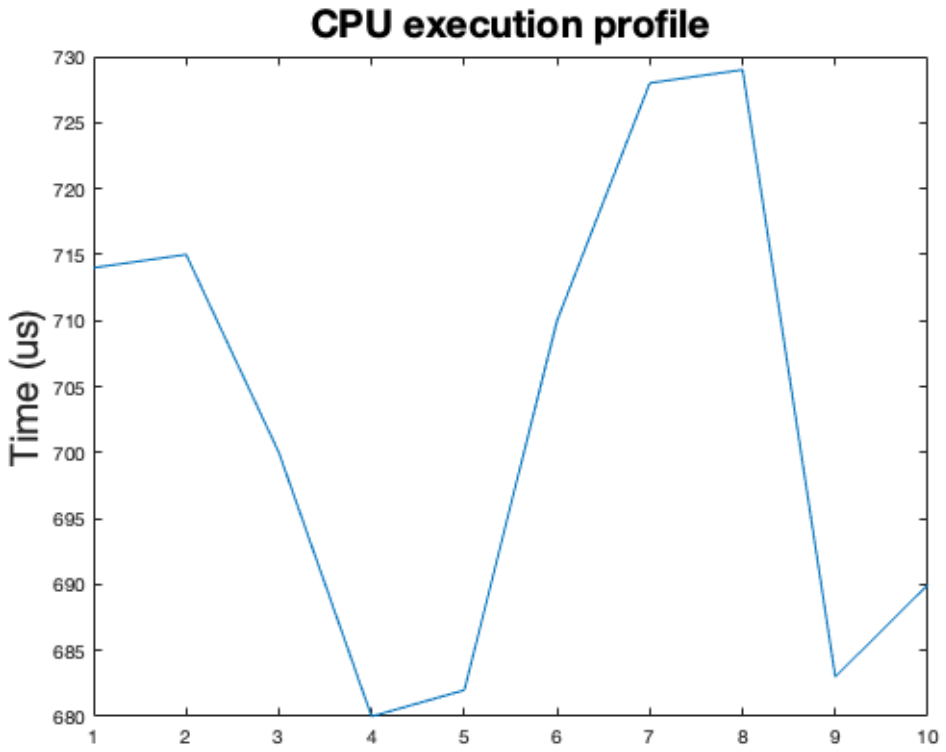


LeNet-1 Architecture

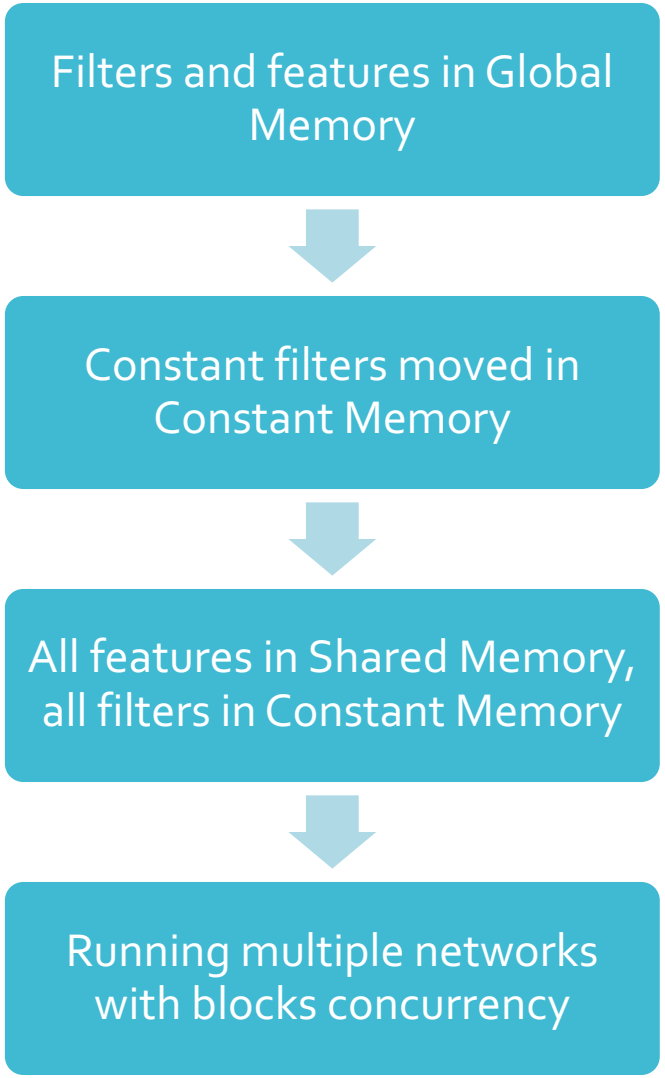
CPU serial version benchmarks

- Random images and random weights have been used for the benchmarks
- The CPU computes each neuron value in series, so 1 single thread doing all the job.
- Sampled execution profile shows an average around 700 us for each forward propagation

CPU performances sampled from 10 runs of the algorithm on different images and weights



GPU versions

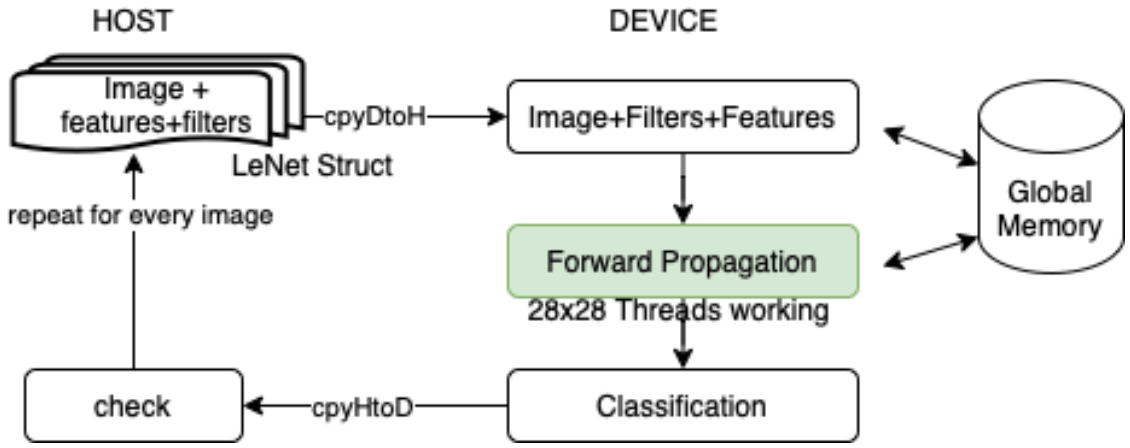


1)

All features and filters are moved and filled in global memory

1 block with 28x28 threads running the CNN

Heavy usage of global memory is the bottleneck!



1)

Load and store efficiency is low because of strided access from convolutions and poolings

Load efficiency: 49.17%

Store efficiency: 76.15%

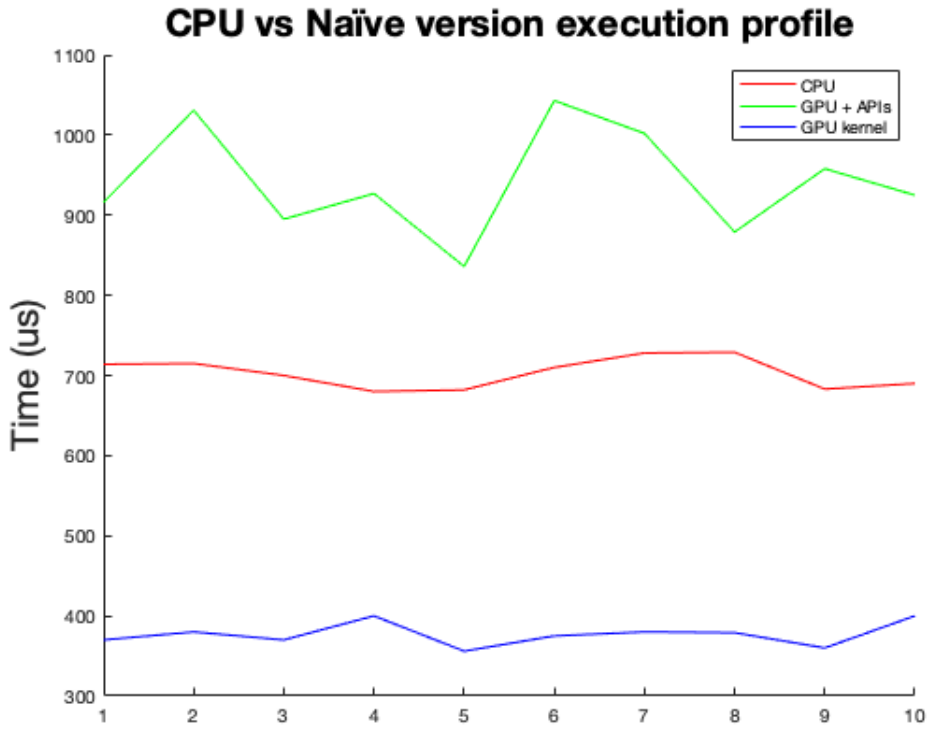
Load transactions: 124610

Store transactions: 632

API overhead between host and device

1)

The GPU executes faster but the data exchange between host and device leads to bad performances



2)

After a CNN is trained, all filters are constant!

Device's contant memory is big enough to contain all filters.

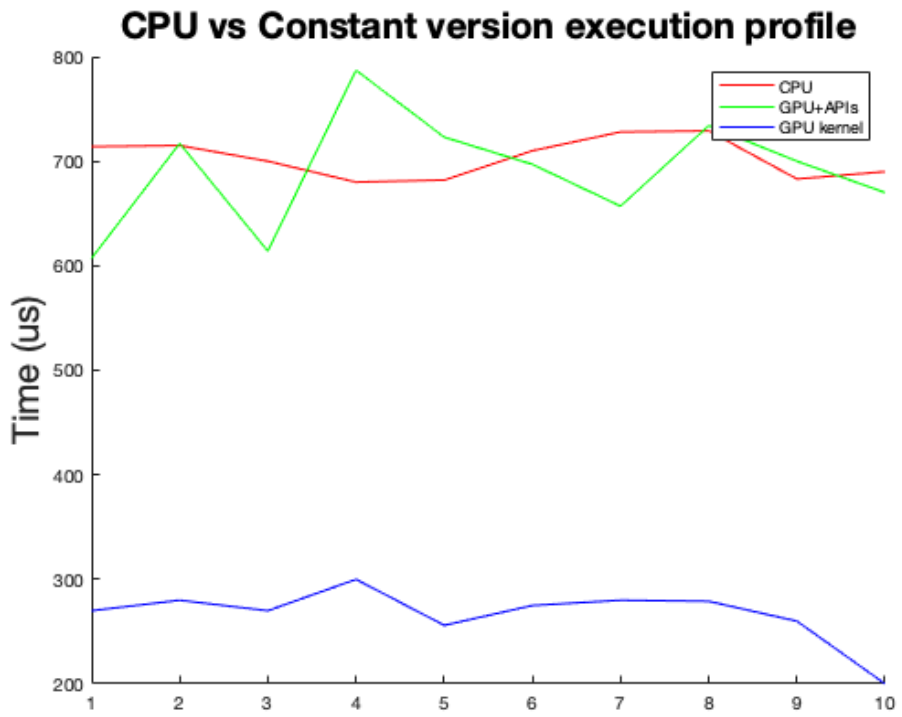
Load transactions: 12000+ → 64610

Load efficiency: 49% → 60%

Overall better usage of memory and less global load requests.

2)

GPU performances getting better

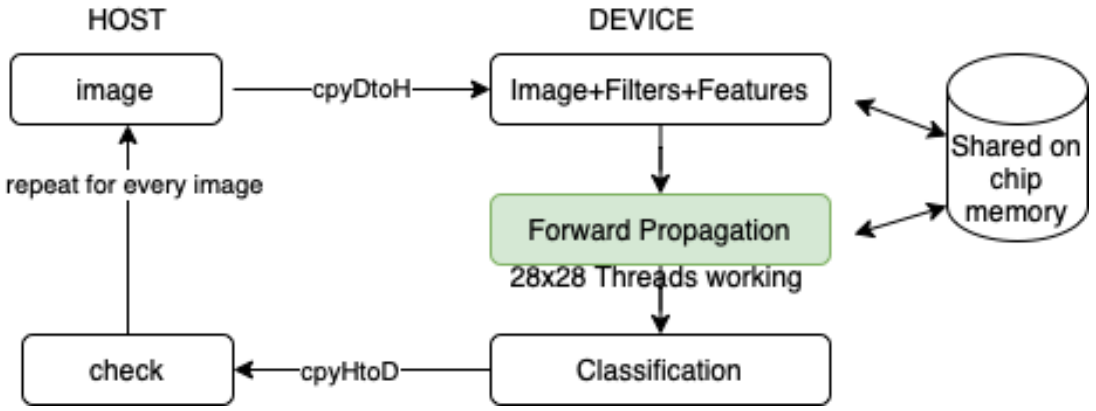


3)

All features can be allocated and computed in Shared Memory

The kernel just needs the image

Global memory usage is negligible because the image and the classification are the only one moved off-chip



3)

All 28x28 threads can load the image in a coalesced way!

Load efficiency: 60% → 100%

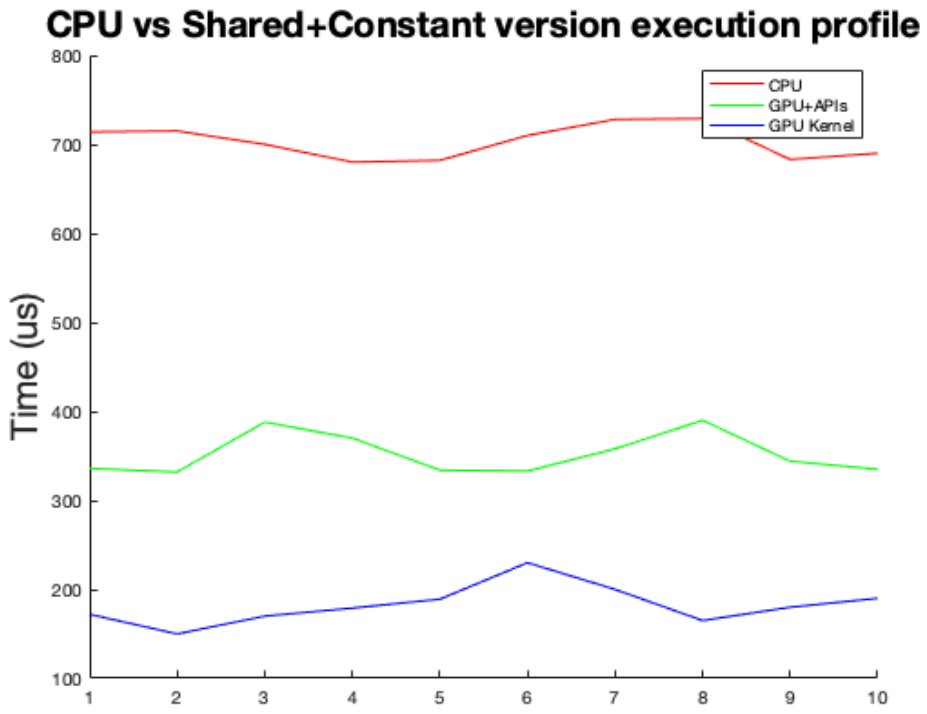
Load transactions: 64000+ → 394

Store transactions: 632 → 2

API overhead between host and device higly reduced

3)

GPU can now run the forward propagation almost 2x faster than the CPU



Blocks concurrency

- Previous implementations make use of a single block
- It is possible to run the algorithm on different images in parallel on the gpu because each block can work on his own allocated shared memory!
- So now is possoble to run N instances of the CNN on N different images with N concurrent blocks!

GPU acceleration can move further now