NeuFlow: A Runtime Reconfigurable Dataflow Architecture for Vision

Clément Farabet[1,2] Yann LeCun[1]
Eugenio Culurciello[2]
[1] Courant Institute of Mathematical Sciences, New York University
715 Broadway, New York, NY 10003
[2] Electrical Engineering Department, Yale University
10 Hillhouse Avenue, New Haven, CT 06511
`cfarabet@nyu.edu`
`http://www.neuflow.org`

Computer vision is the task of extracting high-level information from raw images. Generic, or general-purpose synthetic vision systems have for ultimate goal the ellaboration of a model that captures the relationship between high-dimensional data (images, videos) into a low-dimensional decision space, where arbitrary information can be retrieved easily. The exploration of such models has been an active field of research for the past decades, with a particular focus on biologically-inspired visual models [9, 7, 6].

Biologically inspired vision models, and more generally image processing algorithms are usually expressed as sequences, or more generally graphs of operations or transformations. They can be well described by a modular approach, in which each module processes an input image bank and produces a new bank.

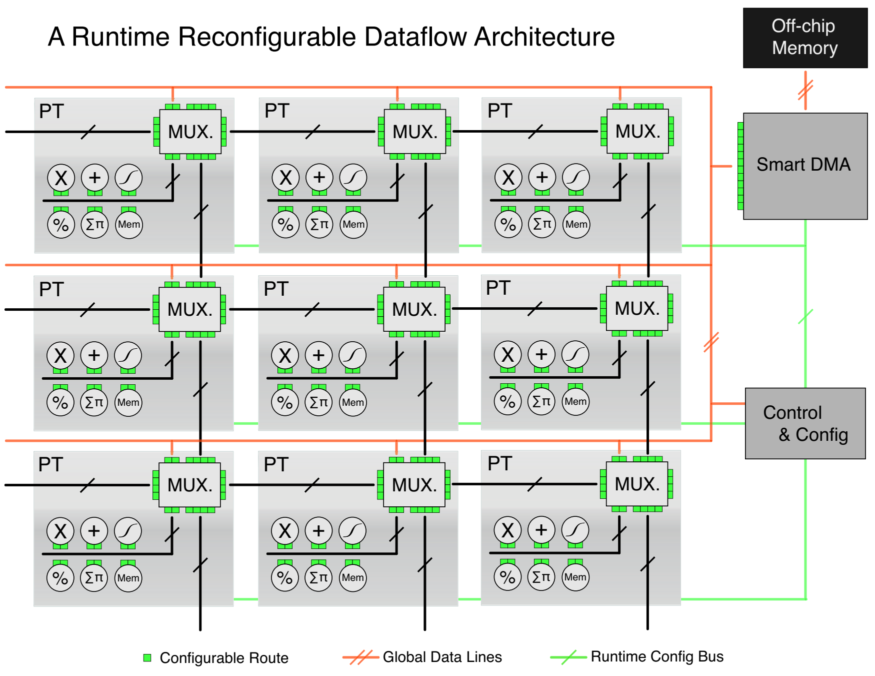


Figure 1: A runtime reconfigurable dataflow grid of processing tiles (PTs).

We present a scalable hardware architecture for large-scale multi-layered synthetic vision systems based on filter banks—*neuFlow*—and a dataflow compiler—*luaFlow*—that transforms a high-level flow-graph representation of an algorithm into machine code for neuFlow. This vision processor is a dataflow engine that can perform real-time detection, recognition and localization in mega-pixel images processed as pipelined streams. The system was designed with the goal of providing categorization of an arbitrary number of objects, while consuming ten times less than a bench-top or laptop computer (on the order of 10W for an FPGA implementation).

First dataflow hardware architectures were introduced in [1, 3]. Our work builds on these ideas, and extends the concept of dataflow grid, and runtime configurability. A schematic summary of the *neuFlow* system is presented in Figure 1. The main components are: (1) a *Control Unit* (a custom 64bit CPU), (2) a grid of *Processing Tiles (PTs)*, and (3) a *Smart DMA* interfacing external memory via a standard controller.

The *PTs* are independent processing tiles laid out on a two-dimensional grid. The PTs contain a routing multiplexer (MUX) and local operators. Local operators can range from simple, generic adders, multipliers, dividers to nested operators such as multiply-and-accumulate arrays (to efficiently compute convolutions) or non-linear mapping engines (to efficiently compute non-linear functions). These operators are fully pipelined to produce one result per clock cycle.

A key aspect of this grid is its configuration capabilities. Many systems have been proposed which are based on two-dimensional arrays of processing elements interconnected by a routing fabric that is reconfigurable. Field Programmable Gate Arrays (FPGAs) for instance, offer one of the most versatile grid of processing elements. Each of these processing elements can be connected to any of the other elements of the grid, which provides with the most generic routing fabric one can think of. The main drawback is the reconfiguration time, which takes in the order of milliseconds.

The grid presented here offers restricted connectivity (each tile can only be connected to its neighbors and a few global N-to-N data lines), and relies on a network-on-chip (NoC) to efficiently broadcast configuration packets. Each module (PT and DMA ports) in the design has a set of configurable parameters, routes or settings (depicted as squares on Figure 1), and possesses a unique address on the network. The *control unit* interfaces the NoC to efficiently reconfigure the grid at runtime. Reconfiguration times are in the order of 10^{-6} to 10^{-8} seconds, allowing the grid to be reconfigured thousands of times a second.

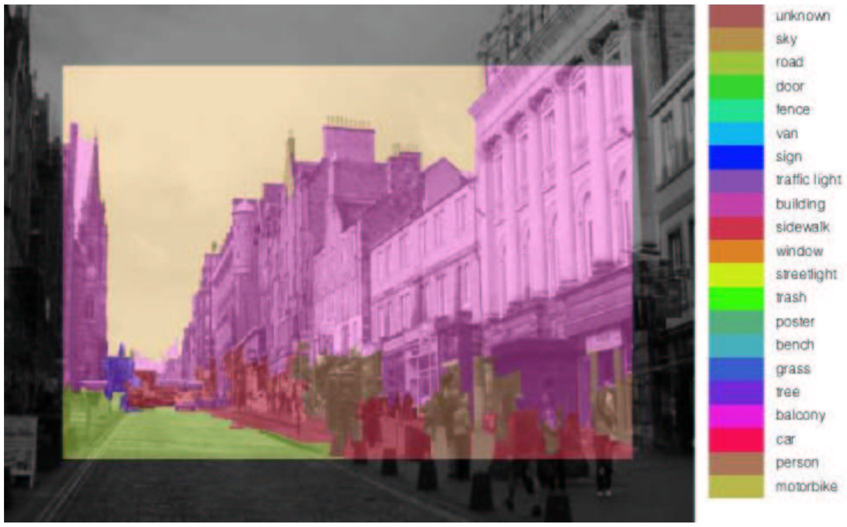
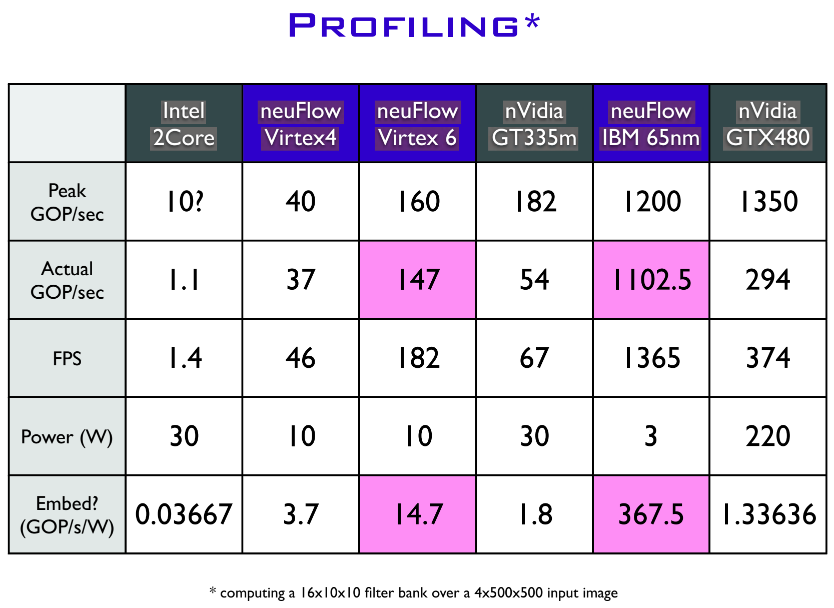


Figure 2: Street scene parsing: a convolutional network was trained on the LabelMe spanish dataset [8] with a method similar to [5]. The training set only contains photos from spanish cities; the image above is a picture taken in Edinburgh. The convolutional network is fully computed on neuFlow, achieving a speedup of about 100x (500x380 images are processed in 80ms, as opposed to 8s on a laptop).

Several applications were implemented: from a simple face detector [4] to a pixel-wise obstacle classifier [2] and a complete street scene parser, as shown on Figure 2. Others can be seen at `www.neuflow.org`. Figure 3 reports performance comparisons between several platforms.



PROFILING*

	Intel 2Core	neuFlow Virtex4	neuFlow Virtex 6	nVidia GT335m	neuFlow IBM 65nm	nVidia GTX480
Peak GOP/sec	10?	40	160	182	1200	1350
Actual GOP/sec	1.1	37	147	54	1102.5	294
FPS	1.4	46	182	67	1365	374
Power (W)	30	10	10	30	3	220
Embed? (GOP/s/W)	0.03667	3.7	14.7	1.8	367.5	1.33636

* computing a 16x10x10 filter bank over a 4x500x500 input image

Figure 3: Performance comparison. 1- Intel DuoCore: laptop-class CPU, 2.7GHz, optimized C code, 2- neuFlow on Xilinx Virtex4/6: two implementations of neuFlow—actual measurements; 3- neuFlow on IBM 65nm process: simulated results, the design was fully placed and routed; 4- two GPU implementations: low power GT335m and high-end GTX480.

References

[1] Duane Albert Adams. *A computation model with data flow sequencing.* PhD thesis, Stanford, CA, USA, 1969.

[2] Benoit Corda, Clément Farabet, Marco Scoffier, and Yann LeCun. Building heterogeneous platforms for end-to-end online learning based on dataflow computing design. Dec 2010.

[3] Jack B. Dennis and David P. Misunas. A preliminary architecture for a basic data-flow processor. *SIGARCH Comput. Archit. News*, 3(4):126–132, 1974.

[4] Clément Farabet, Cyril Poulet, Jefferson Y. Han, and Yann LeCun. Cnp: An fpga-based processor for convolutional networks. In *International Conference on Field Programmable Logic and Applications (FPL'09)*, Prague, September 2009. IEEE.

[5] David Grangier, Léon Bottou, and Ronan Collobert. Deep convolutional networks for scene parsing. ICML 2009 Deep Learning Workshop, June 2009.

[6] Koray Kavukcuoglu, Marc'Aurelio Ranzato, Rob Fergus, and Yann LeCun. Learning invariant features through topographic filter maps. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'09)*. IEEE, 2009.

[7] Nicolas Pinto, David D Cox, and James J DiCarlo. Why is real-world visual object recognition hard? *PLoS Comput Biol*, 4(1):e27, 01 2008.

[8] B Russell, A Torralba, K Murphy, and W T Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 2007.

[9] Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005.