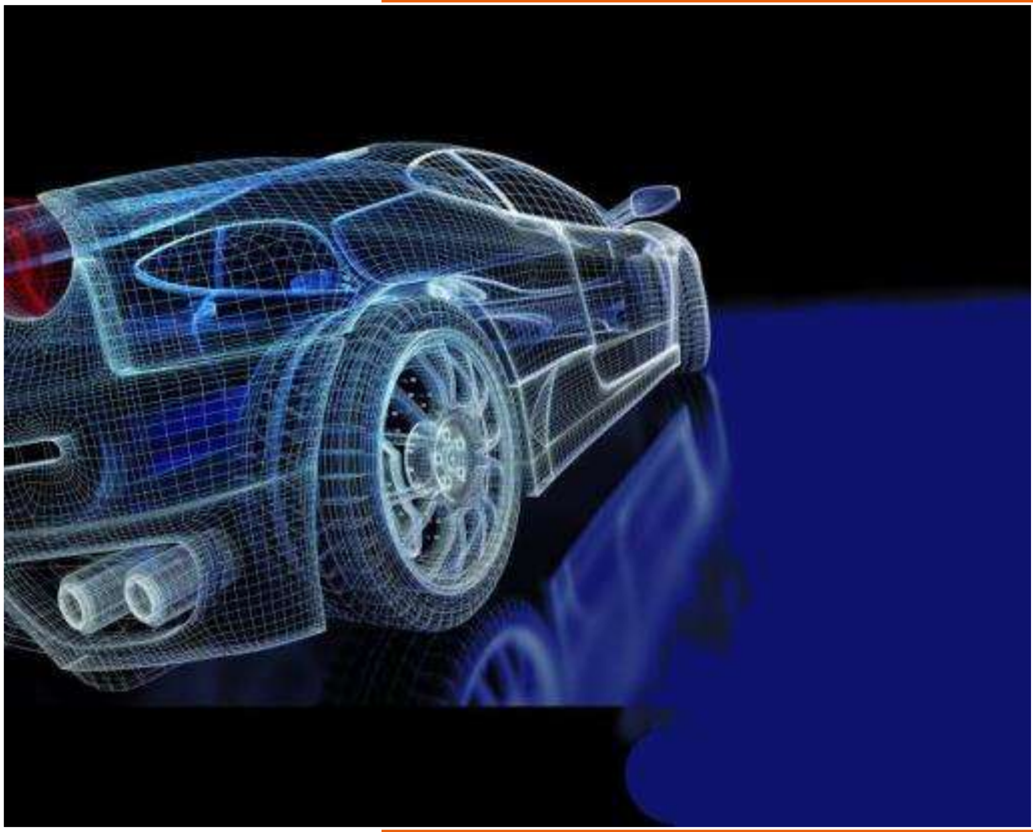
WHITE PAPER

Deep Learning on MPPA® manycore processor



Kalray

http://www.kalrayinc.com

Table of Contents

1 Introduction

Speech, face and handwriting recognition are straightforward tasks that humans are able to quickly fix, while for a machine represent a real challenge.

The human brain structure is like a massive parallel network of processing units and its representation on a computer is called neural network. A neural network is based on a set of algorithms named "Deep Learning".

Deep Learning provides to the machine the ability to make the right decisions, by modeling high-level abstractions in data through self-training.

Various deep learning architectures such as deep neural networks, convolutional deep neural networks, and deep belief networks have been applied to fields such as computer vision, automatic speech recognition, natural language processing, and music/audio signal recognition.

The convolutional neural network (CNN), the most well-known deep learning algorithm, has been extensively adopted in various applications.

Deep Learning algorithms are very demanding in terms of computing power and are very challenging for most existing processors. The purpose of this document is to present an implementation of Deep Learning on Kalray's new processor family and to provide comparison data of Kalray's solutions versus the latest generation of GPUs.

In our particular case, we propose to study Convolutional deep Neural Networks (CNN).

1.1 What is a Convolutional deep Neural Network?

The CNN is inspired by biological processes and is a powerful tool for vision problems such as image classification[1], scene parsing[2], video analytics, and pattern (speech, image, handwriting) recognition used by large industries such as robotics, video surveillance and data centers.

Computer vision and pattern recognition are essential for web search engines with image and video recognition, autonomous cars or medical applications such as cancer detection.

For example, scene parsing is used for advanced driver assistance systems (ADAS) while image classification is widely used in cloud computing applications by many of the largest web and social media companies. Speech (or word or speaker) recognition can be useful for security applications.

Due to the specific computation pattern of CNNs, general-purpose processors are not efficient for CNN implementation and can hardly meet the performance. Thus, massively parallel architectures such as the MPPA® are well adapted for CNNs because of the huge number of floating point operations, the power efficiency and on chip memory size.

Thanks to its of high power computing ability and low energy consumption, the MPPA® processor is interesting both for big data centers where the MPPA® acceleration card (i.e. TurboCard2: 1024-core PCIe acceleration card) offloads the main CPU or for embedded systems like robots, drones or ADAS systems in cars where the MPPA® is used as a standalone system.

[1] http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

[2] http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Socher_125.pdf

2 Implementation on the MPPA®

2.1 The MPPA® product family

2.1.1 The MPPA® manycore processor

The MPPA®-256 manycore processor brings 700Gops of computing power on a single chip typically consuming 11W, with high programming flexibility using the Kalray development suite based on C/C++ or OpenCL programming languages.

This chip is processed using 28nm CMOS technology. It is composed of 16 clusters connected by a network on chip. Each cluster is composed of 16 VLIW processors sharing 2MB of memory. High throughput 2x64bit DDR3 and PCIe Gen3 16 lane interfaces are available for data transfers and storage.

Connected to a host processor through the PCIe, the MPPA® is a low power massively parallel coprocessor well adapted for image or video processing in the cloud infrastructure but it can be used as a standalone system for embedded systems when it runs a rich operating system.

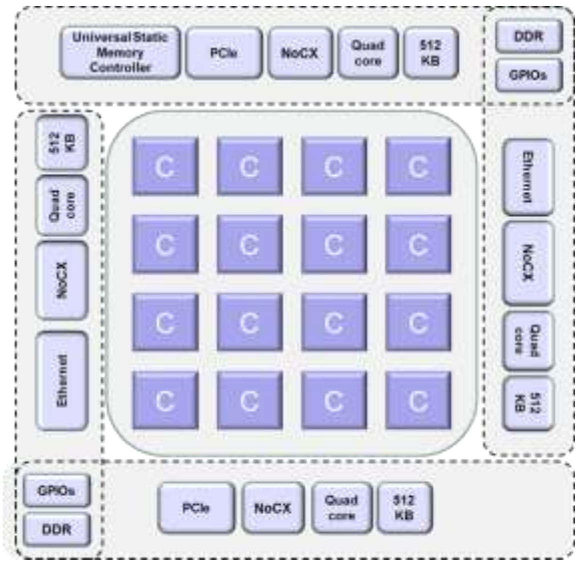


Figure 1 - MPPA®-256 block diagram

2.1.2 TurboCard2 accelerators

The TurboCard2 is a PCIe accelerator for an x86 or an ARM host. It embeds four MPPA® manycore processors on a standard PCIe form-factor in order to be used in server environments with high density requirements. It is programmed using the same tools and languages as the MPPA® manycore processor while providing x4 performance.

2.2 Selected Algorithm

CNN are built from two main basic blocks, critical for performance:

- The 1st part (convolution part) is characterized by a high number of neurons for each layer with the same coefficients. The number of coefficients is low (few hundreds).
- The 2nd part (full connected part) is used to classify an unknown input into different classes.

There are fewer neurons inside each layer but coefficients are not shared and a large number of coefficients are required to compute each layer (the number of input neurons multiplied by the number of output neurons).

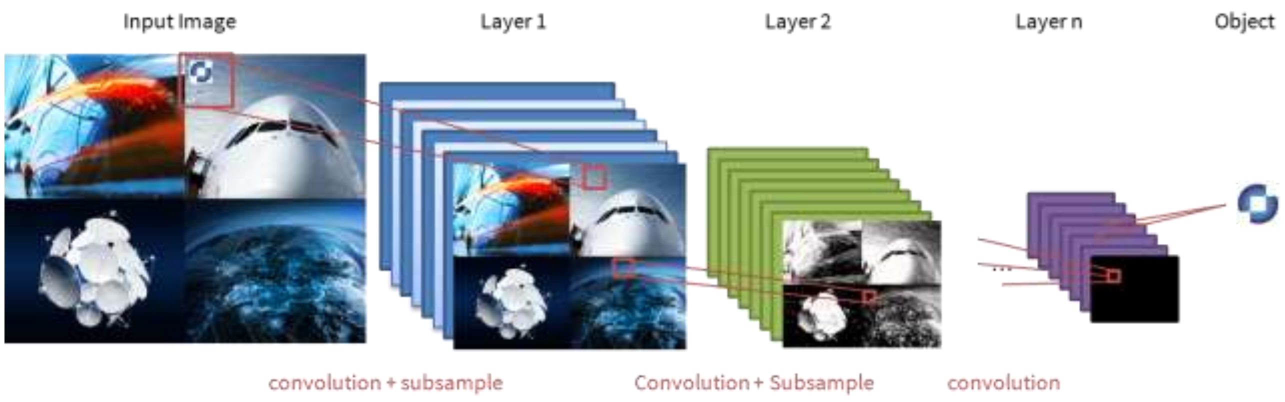


Figure 2 - Convolutional Neural Networks

The full-connected parts dominate when the batch size is small (typically 1 or 2). The convolution parts dominate when the batch size increase[3].

2.3 CNN implementation on the MPPA®

The implementation of the CNN on the MPPA® for high throughput systems takes advantage of the ability of the MPPA® to store a large amount of neurons inside the shared memory of each cluster.

Convolution and full-connected parts are implemented in different ways to maximize the MPPA efficiency.

2.3.1 Convolution kernels

The convolution part is critical in the performance of CNNs.

The convolutions are performed by using a direct convolution kernel implementation with blocking, in order to maximize the reuse of data. The parallelization is then done on 2 levels:

- 1st between different input images for each cluster
- 2nd by splitting each feature map between cores in a cluster

This allows to process feature maps entirely in local-memory.

2.3.2 Full-connected kernels

Full-connected kernels have still a significant impact on performance, especially in the absence of batching (eg. in production, once the learning phase is completed).

Full-connected kernels are performed by using the direct full-connected kernels implementation with blocking, in order to maximize the reuse of data. Each individual core performs the full computation with different input data, entirely in local memory.

[3] "Learning Semantic Image Representations at a Large Scale", Yangqing Jia, http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-93.html

2.3.3 Other functions

Other functions such as neuron activation functions, max-pooling, etc. are less critical for performances, and are implemented in their simplest C/C++ form. The user can then easily extend those set of functions through standard C/C++ programming.

2.4 Considered Use Cases

The use of the Kalray MPPA® and the TurboCard2 for CNN computations can be separated into two main use-cases: training and production, and into two main operation models: acceleration or micro-server.

2.4.1 Production

The main characteristics of CNN workload "in production" are no intermediate layers results savings. In that case each layer output can be directly reused locally to compute the next layer. This is possible on the MPPA® thanks to the high amount of on-chip memory.

Batching is hardly be possible in production if on-line classification is a must.

We separate convolutional layer processing and full-connected layer processing to maximize the available space. With this architecture, very few communications are needed: only the data for the initial layer plus each layer's parameter (eg. convolutions coefficients). The other layers (max-pooling…) can be directly merged with the previous layer to further increase efficiency.

2.4.2 Training

CNN training requires a special architecture in order to save intermediate results for each layer. In that case input data and kernel parameters are streamed to local memory, and results are streamed back for later reuse. The bandwidth needed is low in any case and below the various components' bandwidth (PCIe, DDR and NoC) and does not limit the computations.

2.4.3 Acceleration

In acceleration mode, the MPPA® is used as a PCIe accelerator for the host CPU (x86, ARM...). In that case all exchanges go through the PCIe, and the PCIe on the MPPA® supports RDMA up to the local memory of each cluster, eliminating the need for on-board DDR and improving latency. The PCIe is not a bottleneck in the case of the MPPA® as the convolution input feature maps do not need to be linearized. Moreover, the communication model is RDMA with double-buffering, allowing communication latency hiding.

2.4.4 Micro-server

The MPPA® can run as a standalone processor running Linux. In that case, the data can be exchanged either over a PCIe backplane or directly through Ethernet. Multiple MPPAs® can also be natively interconnected together for direct RDMA between MPPA® local memories.

2.5 Transparency to developers

The low level libraries implementing the above functions can easily be integrated into widely used software frameworks such as Caffe[4] or Torch[5]. In this way, deep learning acceleration is fully transparent to developers.

[4] http://caffe.berkeleyvision.org/

[5] http://torch.ch/

3 Measures and Comparisons

GPUs are currently the most popular computing solution.

As a reference, to compare the MPPA® solution and other processors, we use the de-facto standard benchmark convnet-benchmark[6]: This benchmark implements 5 convolutional layers with different parameters in term of input features maps size and number, kernel size and output feature maps number.

In table1 and table2, we compare the achieved GFLOPS averaged over those 5 layers with a batch size of 1 to measure and compare the efficiency of a CNN implementation. A comparison of the energy efficiency is also given. In table1, we compare the Kalray processor generations while in the table2, we compare Kalray PCIe acceleration board (TurboCard) and Nvidia K40 PCie acceleration board.

Architecture	Power (W)	GFLOPS (higher is better)	GFLOPS/W
Kalray MPPA®-256 Andey	15	165	11
Kalray MPPA®-256 Bostan[7]	25	402	16

Table 1 - MPPA® convnet-benchmark layers averaged, no batching

Architecture	Power (W)	GFLOPS (higher is better)	GFLOPS/W	Ratio vs. K40
Kalray TurboCard2 (Andey)	70	660	9.4	x2.6
Kalray TurboCard3 (Bostan)[7]	110	1608	14.6	x4
NVIDIA K40 (Kepler)[8]	235	850	3.6	1

Table 2 - Acceleration card, convnet-benchmark layers averaged, no batching

There is no such benchmark for fully-connected layers, but in table3 we show the performance of a 4096x4096 full-connected layer on MPPA® as used in the well-known AlexNet CNN architecture[9].

Architecture	Power (W)	GFLOPS (higher is better)	GFLOPS/W
Kalray MPPA®-256 Andey	15	170	11.3
Kalray MPPA®-256 Bostan[7]	25	410	16.4

Table 3 - MPPA® full-connected benchmark

[6] convnet-benchmark, "Easy benchmarking of all public open-source implementations of convnets", https://github.com/soumith/convnet-benchmarks

[7] Based on simulation results – Bostan will be available in Q3 2015

[8] "cuDNN: Efficient Primitives for Deep Learning", http://arxiv.org/pdf/1410.0759.pdf

[9] "ImageNet Classification with Deep Convolutional Neural Networks", http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

4 Conclusion

Deep Learning is changing the way computers and applications react. Deep Learning is being adopted in a growing number of functions and applications, both in server environments and embedded systems.

Kalray's unique MPPA® architecture is extremely well suited for running algorithms such as Deep Learning. Kalray's deep learning solutions are significantly better than the standard alternative in terms of performance and cost. In addition, the MPPA® delivers a performance per watt ratio much higher than CPUs or GPUs while offering a standard C/C++ programming environment.

For large deployment of CNN-based applications, such as feed-forward for image/video classification, the Kalray MPPA® is the perfect solution.