Exploration and design of embedded systems including neural algorithms

Jean-Marc Philippe, Alexandre Carbon
CEA, LIST
Embedded Computing Lab
F-91191 Gif-sur-Yvette, France
Email: jean-marc.philippe@cea.fr

Olivier Brousse, Michel Paindavoine
GlobalSensing Technologies
14 rue Pierre de Coubertin
F21000 Dijon, France
Email: olivier.brousse@globalsensing.eu

Abstract—The current trend in embedded systems is to make them surrounding the users, providing services thanks to a knowledge of their environment. These self-awareness and context-awareness properties are provided by numerous sensors, from different types. Using the provided information causes at least two problems: the fusion of data from different sources, and the noise induced by sensors which are closer from the processing unit than ever. Additionally, the needed applications that use these information are based on different recognition processings, sometimes not easy to formalize with conventional algorithms. Processing chains using neural-based algorithms are promising approaches for solving these kinds of issues. Unfortunately, embedding bio-inspired algorithms in an embedded system is not so easy since there is no exploration environment for this specific task. Moreover, neural networks often need pre- or post-processing of data for optimal operation.

In fact, there is a balance to find between pre-processing and neural network processing: for example, adding more filtering to clean or to transform data (like convolution filters or FFT) enables to have smaller neural networks, leading to less number of neurons, less learning time and finally more efficient applications. This paper presents early results of a collaboration towards the design of such an exploration environment coming from a joint laboratory between an SME and a Research Institute. The main object coming from the current collaboration is the coupling of a rich exploration environment of embedded systems (including multi/manycore) with a neural network exploration tool.

The combination of the two enables us to have feedbacks concerning both algorithm efficiency and performances and other non-functional metrics regarding the target system for driving the co-design cycle of industrial embedded systems.

I. INTRODUCTION

The growing interest for pervasive systems that seamlessly interact with their environment motivates research in the area of smart objects. These deeply embedded and aware systems are intended to show smart behaviors and to solve complex issues such as pedestrian counting, human postures and elderly survey, bulk objects sorting, etc. From urban safety to industrial applications (Figure 1) (such as default detectors) passing via human posture recognition (as presented by Figure 2a) and worker detection in dangerous areas or perimeter security (illustrated by Figure 2b), a lot of those applications rely on the ability of the system to perform recognition. This kind of ability is not easy to achieve with legacy algorithms. Based on their properties, Artificial Neural Networks (ANN) are serious candidates [1] to implement these smart applications on embedded systems and are nowadays found in applications at home and both in office and industry environments.

Nevertheless, ANN also take benefits from raw data pre-/post-processing regarding the quality of the recognition. In fact, there is a balance to find between pre-processing and neural network processing: for example, adding more filtering to clean or to transform data (like convolution filters or FFT - Fast Fourier Transform) enables to have smaller neural networks, leading to less numbers of neurons, less learning time and finally more efficient applications. For example, decreasing the number of neurons or layers in a neural network can be
very interesting in fully connected networks such as MLP (Multi-Layer Perceptron) or RBF (Radial-Basis Functions) [2] networks since the number of multiplication-accumulation (computation) as well as the number of synapses to store is reduced (memory optimisation).

A good example of pre-processing usefulness is for example dealing with human body whose shape is moving and changing over different parameters (weight, posture, clothes, age and so on). The first common operation to perform in human body-related applications is the segmentation (shape recognition) of the body versus the background. Therefore algorithms such as Mixture of Gaussian (MOG) [3], [4] or Sigma-Delta (SD) as presented in Figure 3 maybe used as first processing stages.

Those algorithms allows us to deal with the background cancelling and therefore to put the focus on human shapes on a black background rather than to deal with the much more complex scenes with a huge amount of textures that induce to use more complex ANNs. With the same goal, oriented-edge filters such as Gabor filters are really useful to get relevant data from images and are generally used as first stages of more evolved processing chains such as Hierarchical Maximum (HMAX) [10].

Another point is that reducing the number of neurons also reduces the number of non-linear functions to execute which are typically time consuming especially when non-simplified models are used (use of exponential or gaussian functions, etc.). This rises an issue: how to split work between legacy data processing and ANN. This is especially true when dealing with systems embedding heterogeneous resources (e.g. general-purpose processors, or hardware accelerators) like on nowadays System on Chips (SoCs) or the well-known Xilinx Zynq [5], or even dedicated co-processors like the one depicted in Figure 4, composed of a Xilinx Spartan 6 FPGA together with a first generation neural network accelerator.

This partitioning requires exploration tools to find the optimal split to cope with the low computing power or power efficiency versus application complexity trade-off which is the traditional issue in embedded systems. In fact, taking advantage of co-design opportunities for neuro-inspired algorithms on embedded systems in one of the goal of this platform. This paper presents our approach based on co-design using an ISS-based simulator with approximate timing for system design space exploration and an ANN exploration tool for the application point of view.

The remainder of this paper is as follows. Section II presents the two tools (i.e. simulation framework for embedded systems and ANN exploration) composing the exploration framework whereas Section III presents early results coming from use cases. Then, the last section concludes the paper.

II. EXPLORATION FRAMEWORK FOR NEURAL-BASED EMBEDDED SYSTEMS

One of the hardest part of the exploration of systems including neural algorithms is to perform the co-design between the system itself, including non functional characteristics, and the processing chain, including the neural algorithm. For this purpose, the study was divided into two levels: the system one, using simulation platforms and the processing chain one using neural network exploration simulator.

A. System level exploration

A simulation platform was designed using the SystemC-based SESAM environment [6] to perform system design space exploration. Besides being able to simulate complete systems, it is compatible with co-simulation and co-emulation to keep the same framework during the design flow, easing verification and debugging. Apart from pure functional simulation, SESAM allows to perform system exploration including timing, power consumption and even thermal and ageing [7] characteristics.

The target system is a classic embedded system platform (depicted in Figure 5) comprised of:

- an applicative processor for system management, application execution and user interaction,
- a generic interface for sensors (images, audio, etc.),
- a system memory for data storage (filter coefficients, neural weights, etc.),
- a DMA (Direct Memory Access) transferring data from the sensors (peripheral mode) to the system memory,
- and finally, a block was dedicated to the implementation of the neural part of the application (software on a processor, an IP on an FPGA or even a complete chip).
SESAM allowed quick system modeling, illustrated by Figure 6: the processor is simulated using a MIPS Instruction Set Simulator (ISS) with cache memories and a translation look aside buffer (TLB). The sensor interface is connected to an "External bus" responsible for communications to and from the workstation executing the simulation platform. This infrastructure separates the simulated part of the system from external host accesses to the simulator.

Neural algorithm part was modeled using a dedicated block in the system to enable exploration. Apart from the neural algorithm function itself, the model can be annotated to explore timing or power consumption at system level by executing target applications. This block is composed of interfaces for both control and data and for linking dynamic libraries of processing or neural functions. These functions can be configured with a text file, result of neural network exploration, allowing parameterized exploration without recompiling.

B. Processing chain level

Most of the considered applications (from vision domain, etc.) rely on the ability to discriminate objects in scenes or events in signals. Hence, the flexibility offered by ANNs in a unified framework allows us to easily deal with various applications and contexts.

Two different types of processing chains are promoted here. The first one is based on legacy and arithmetical pre-processing followed by an ANN-based classification or interpretation of data. For example, CMOS sensor-based human detection relies on an adaptive movement detector that filters the scene and isolates moving objects by getting rid of the background. Once this stage is done we apply a data transformation that allows data classification through an ANN chain.

The second processing chain is based on complete neuro-inspired processing chains such as Convolution Neural Networks (CNN) and HMAX. Those chains transform grabbed data and process them in a neuro-inspired way until the classification.

The unified framework is based around parameterizable macros that implement legacy image processing as well as several ANN implementations such as RBF [8], CNN [9] and HMAX [10]. Those macros allow the user to setup its processing scheme, simulate results as shown in Figure 7 and once the chain is established, to export this offline configuration to a parameter file, aimed to be read by the embedded system simulator. The GST Neuro Platform (GNP) software is here used to simulate the neuro-inspired chain for both learning and recall phases. This program is intensively used at GlobalSensing Technologies (GST) to build configuration files based on learning base for both our and client applications. It really makes the building of our ANN solutions easier to produce.

The ability to manage those two processing chains within a unified framework helps us to determine what are the optimal solutions to solve issues of applications implementation. As those applications are more and more in the embedded world, we design our framework to deliver offline configuration of processing chains (both pre-processing/ANN and neuro-
III. EXPERIMENTAL USE CASES

This section presents the first two use cases of the exploration framework: pure image processing and neural network implementation.

A. Image filtering algorithm

For the first case, Gabor filters are demonstrated as being the first step of HMAX processing chain, as presented in Figure 8. The HMAX chain is seen to be particularly efficient for vision-based applications. It is composed of several layers of pre-processing: Gabor filters from different scales and with different orientations (S1 layer in Figure 8) which are then combined using a pooling layer (often the local maximum function, named max pooling) to reduce the number of information and to extract the main features of the input image which were highlighted by the Gabor filters.

Then there is a comparison with patches from a dictionary, containing already known features and finally, the maximum of these comparisons are extracted (S2 and C2 layers). The final step of the complete chain is a classifier, which can be an RBF or an MLP neural network.

Figure 9 presents the global simulator user interface including an input image and the output image with a normalized Gabor filtered image as the output.

From the simulator point of view, taking into account the modeled system described in Figure 6, the process is composed of the following steps, all implemented using a real MIPS binary since ISS is used:

1) the application processor triggers the transfer of an input image from the IO interface emulator to the system memory thanks to an acquisition API,
2) once finished, the application processor sends a signal to the Neural Algorithm block (thanks to a memory mapped register) to ask it to acquire the image from the memory,
3) the Neural Algorithm block then transfers the entire image into its internal memory and executes the Gabor filter. It then writes the resulting image to a dedicated area of the memory,
4) once this processus is finished, the Neural Algorithm block sends an interrupt to the application processor that reads the memory and sends the resulting image to the IO emulation module, and especially to the memory mapped interface that emulates a display.

Executing such a filter on the simulation platform enables us to measure its performance using approximate timing: by modelling timing of the different components of the architecture (processing time on application processor is given by the ISS, for example), the designer can evaluate very early in the design process the performance of the system.

Another way of thinking is that the designer can have a maximum time for neural network execution if real-time processing is a constraint. Then, the sizing of the network can be done so that real-time execution is ensured (number of layers versus number of neurons per layer, including their activation function).

B. Parameterized neural network

The other use case illustrates the use of a combination of system platform simulation and the neural algorithm exploration by executing a processing chain recognizing different patterns available on a draughtboard. For this purpose, a generic RBF network was implemented as a dynamic library of the system simulator. From the neural algorithm point of view, the RBF network was trained on different samples to recognize an input vector of 1024 coordinates (32x32 pixels), as presented in figure 10. This input vector is meant to be a Region of Interest for this test application (ROI).

The network is architectured around an hidden layer of four neurons and the output layer is also four neurons (each one corresponding to a decision on the four classes of interesting patterns - pure white, pure black and two diagonals), as shown in Figure 10. The learning phase was performed using the neural network exploration tool resulting in a configuration file
Musawi optimized RBF cells

Output layer

Fig. 10: Illustration of the simple RBF network of the draughtboard use case. The four neurons of the hidden layer are trained to recognize the different patterns which can be found on a typical draughtboard.

This file was then used with the system exploration platform and the application execution lead to Figure 12, illustrating the original image and the classified one on the simulator GUI (Graphical User Interface).

Fig. 11: Sample of the parameter file generated by the neural network exploration environment.

These use cases allowed us to test the different functionalities of the platform, as well as system management exploration: comparison of different synchronization schemes and programming low-level APIs (Application Programming Interface) between the components of the system. Other applications are under design while writing those lines.

IV. Conclusion

The work presented in this paper is partly done within the NeuroDSP project\(^1\). The proposed coupling between two tools for co-design-based exploration of the integration of neuro-inspired components in embedded systems is one of the first step to propose innovative smart products in various application domains from home to perimeter security, industrial environments, etc. The easier integration of ANN-based classifications and detections in embedded systems will enable

---

\(^1\)Acknowledgment: Part of this work was funded by French FUI under the NeuroDSP project with the participation of BPI France, région Bourgogne, conseil général de Savoie, Grand Dijon, Arve Indusries and Vitagora.
to get an efficient matching between architectures (possibly heterogeneous ones) and algorithms. The efficiency of the ANN being tested offline, it allows the simulator to focus on the application matching toward targeted embedded systems without the simulation of several complex contexts as it is often the case in ANN-based solution development and by including non functional properties such as power consumption and even thermal and ageing characteristics. Relying on the powerful SESAM exploration environment as simulation tool, our work will enable to have a complete exploration environment which will be able to span the entire design process from early virtual prototyping to co-simulation, co-emulation and even validation. Studies related to complete industrial applications for high-value markets are currently performed in the joint laboratory between GST and Commissariat à l’énergie atomique et aux énergies alternatives (CEA).

REFERENCES


