A Tool for Fast Development of Modular and Hierarchic Neural Network-based Systems

Francisco Reinaldo[1,2,3],Mauro Roisenberg[2],Rui Camacho[1], Luís Paulo Reis[1]

[1] FEUP - Faculdade de Engenharia da Universidade do Porto
LIACC - Laboratório de Inteligência Artificial e Ciência de Computadores
Rua Dr. Roberto Frias, sn, 4200-465
Porto, Portugal
reifeup@fe.up.pt
[2] UFSC - Universidade Federal de Santa Catarina
INE - Departamento de Informática e Estatística
Campus Universitário - CEP 88040-900
Florianópolis, Santa Catarina, Brasil
[3] UnilesteMG - Centro Universitário do Leste de Minas Gerais
GIC - Grupo de Inteligência Computacional
Av. Presidente Tancredo Neves n. 3500, Bairro Universitário, CEP 35170-056
Coronel Fabriciano, MG, Brasil

Abstract

This paper presents PyramidNet tool as a fast and easy way to develop Modular and Hierarchic Neural Network-based Systems. This tool facilitates the fast emergence of autonomous behaviors in agents because it uses a hierarchic and modular control methodology of heterogeneous learning modules: the pyramid. Using the graphical resources of PyramidNet the user is able to specify a behavior system even having little understanding of artificial neural networks. Experimental tests have shown that a very significant speedup is attained in the development of modular and hierarchic neural network-based systems by using this tool.
Key-words: Agent, Behavior, Architecture, PyramidNet.

1 Introduction

Cognitive processes are necessary to reach autonomous behaviour. A cognitive process is a decision-making route of beliefs or desires that triggers actions for emerging behaviours [6]. In order to be autonomous, an agent extracts information from dynamic and uncertain environments for reasoning and after acting on such environments. In addition, an agent must include cognitive reasoning to transform the gathered data in actions to achieve goals. Other processes like perception, learning, deduction and planning are important as well [12].

The quality of a cognitive process, which measures the best followed decision, determines a degree of autonomy in agents. This degree of autonomy is directly related to its own abstract capability to decide how sensors and actuators will work to achieve goals.

We define that an Artificial Neural System (ANS) is composed by cognitive processes working together to emerge reasoning. Several clusters of ANS constitute a Modular and Hierarchic Behaviour Architecture (MHBA).

Each autonomous agent [7] runs using an Executable Behaviour System (EBS) built-in. An EBS is a ready-to-use portable code that was developed based on Behaviour System Project (BSP). If the user does not have an appropriated tool for design this project, it is very difficult to design from BSP to EBS because a lot of programming time is needed. Also experts in behaviours, knowledge, system analysis, computer science and robotic engineering working full-time are necessary to compile a BSP into a compact EBS structure.

Some knowledge is needed to design a BSP and to implement EBS code because of the degree of autonomy can slowdown. In addition, many cognitive processes may become a scalability problem and may be unable to deliberate the best reasoning planning and reactive actions. A high degree of autonomy is primordial to an agent to reach its goals and survive in a dynamic environment.

The objective of this work was to develop a tool with features to handle heterogeneous colonies of cognitive processes/learning modules. The main focus looks for a MHBA can generate models of BSP. Each BSP can be independent or match with others to be an EBS. In addition, the tool is a valuable contribution to users without deep knowledge about Artificial Neural Networks (ANN) and program code because it offers high level of abstraction

and automatic creation of source-code.

This paper presents the PyramidNet Tool as a fast and easy way to develop a modular and hierarchic neural network-based system. Based on PyramidNet Architecture [9] the user can interact with PyramidNet Tool and create different models of BSP because the tool uses modularity and hierarchic organization of learning modules. The tool has resources to specify the number of layers, the configuration of each inner module and to choose what kind of ANN module can be used into a layer. It is possible to build interconnection among modules, connections to external data to feed a database for training of neural network and execution plan for automatic production of behaviors. Finally, the whole ANS is graphically produced by the user. After training of the whole system, a ready-to-use open-source code is produced like a final result to be applied in agents.

The rest of the paper is divided in six sections. Section 2 introduces the PyramidNet Architecture and features. Section 3 introduces the PyramidNet Framework and sub frameworks. Section 4 presents the PyramidNet Tool and some steps to obtain a full project of the modular and hierarchic connectionist architecture are in Section 5. Section 6 presents the experiment and discusses some results. In Section 7, we draw some conclusions and point future directions for this work.

2 PyramidNet Architecture

The PyramidNet Architecture [9] uses a modular and hierarchical approach of ANNs to emerge reasoning in order to produce behaviours in agents. The use of ANNs can represent many advantages over other proposed control architectures because it supports high noise immunity, fault tolerance and programming by examples [10].

The architecture was inspired in the organizational structure of mammalian nervous system. In fact, our nervous system has hierarchical structure. For example, local control of skin temperature does not depend on the central control, so the PyramidNet architecture follows the same principle, and is composed by multiple levels, each level with a function. These levels/layers of function represent subsequent clusters of ANS that are arranged in a hierarchical way, allowing increasingly behaviours like a cortex (neocortex) [2].

The PyramidNet Architecture is adequate to control behaviours because suggests ANN learning modules organised from reactive to deliberative layers as shown in Figure 1.

The structure of pyramid is arranged to provide a fast emergence of behaviours with high flexibility. The base of the pyramid runs reactions (activities) and the top of the pyramid has responsibility for producing of inner representations (cognitive reasoning). For instance, reflexive survival behaviours with real-time activities are implemented on the reactive level and elaborated behaviours are defined on the deliberative level.

In this way, a Feedforward neural network composes the base of pyramid that is responsible for simplest and

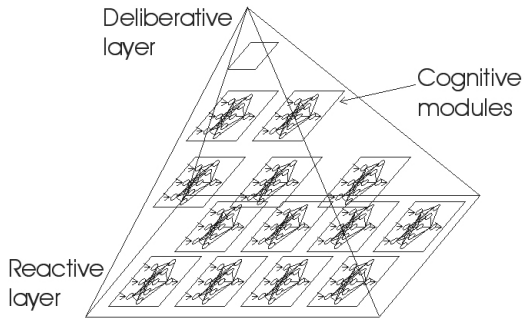

Figure 1: The PyramidNet Architecture

reactive behaviours, exploring the straightforward performance in the effector level. In addition, Recurrent neural networks are arranged in the upper layer of the pyramid representing more elaborated behaviours for controlling the reactive level [9]. Thus, layers and learning modules have communication through interconnections. The hierarchical structure has flexibility to extend and to implement more behaviour layers or learning modules in accordance with complexity to solve a problem.

3 PyramidNet Framework

PyramidNet Framework [6] is an open-source C++ platform that offers a robust group of reusable classes for developing ANN tools. The objective of the framework is to develop and standardize a fast reliable procedure for tools that support the layered organization of learning modules.

The main point of the framework is to offer objects/classes that were definite in framework structure (skeleton) to create future flexible and extensible tools detailing only particularities [12]. The framework enables the construction of a tool through of union/extension of classes and communication between objects to create solutions to similar problems.

The framework has three main sub frameworks: the Graphic User Interface Framework (GUIF), the Artificial Neural Networks Framework (ANNF) and the Automatic Open-Source Code Generator Framework (ACF). GUIF is a set of main classes for handling and modelling learning process using graphic elements that will interact with users. It makes available some items: desktop constructor, sensor, actuator, line links, skins of ANN, scheduler of ANN training, menus, dialog box and others. ANNF offers classes of learning modules. Each ANN class is composed of algorithms, layers, neurons and its respective particularities to be worked. In addition, it is implemented to be extensible for receiving more classes. ACF offers a set of classes able to interpreter a CBS. Also, it can be able to interpret an automatic open-source code of a CBS design.

4 PyramidNet Tool

PyramidNet Tool [5] is a utility that helps users from Artificial Intelligence, Psychology and Robotic areas to design the whole structure of fine behaviours. In order to maintain standard and collaborative tasks, the engine of the Tool was built over some PyramidNet Framework classes.

The main point of PyramidNet Tool is to facilitate for designing and assembling of ANS in a layered but hierarchic way, in the same way as proposed in the PyramidNet Architecture. In addition, the Tool has support to translate ANS project to code. The Tool allows users to handle learning modules in hierarchic layers, design and implement new features to agents can run in dynamic environment for accomplishing elaborated tasks with essentials concepts of ANN and minimum knowledge of programming language. These functionalities relieve the simplest, lightest and fastest way to produce behavior projects to agents.

PyramidNet Tool provides three ANN learning modules that are Feedforward networks [1], Recurrent networks [4] and SOM - Kohonen's maps [8] anything less is like developing behaviours to an agent but letting fine movements and elaborated decisions in through the side. These modules can reach the behaviour levels in order: Stereotyped (reflexive and taxies), Reactive and Deliberative. The Tool permits users to understand the emergence of reasoning in agents within classroom setting. This is one of the many reasons why the PyramidNet Tool has been used as curricular component on the undergraduate courses of computer science in the Federal University of Santa Catarina (UFSC) - Brazil, and Laboratory of Connectionism and Cognitive Computing (L3C) - Brazil.

With PyramidNet Tool users have a full compact and legible code of the ANS project. In a few easy steps, the PyramidNet Tool provides a clean ready-to-use ANSI C open-source core of information fusion, planning and coordination. This process uses a high-performance interpretation algorithm to create the functional and compact core. For user convenience, it is possible to reuse a project and its code because it is easy to extend the code if required.

The graphic environment, called Desktop, is used for drawing ANS Projects. Desktop offers sensors, actuators, lines of connection and learning modules. Each graphical element has its features, such as colour, label, format and size. Heterogeneous learning modules are standard boxes with input and output neurons and have additional features that are specific to ANN that are number of neurons, kind of function to train and follow. Instantly the user has a group of options to configure that all graphic items in desktop. In desktop, sensors send signals to actuators/learning modules and actuators receive signals from sensors/learning modules. Some constraints were used to preserve system's harmony, such as: an input neuron and an actuator can receive only one connection that can be coming from sensors or learning modules; and an output neuron and a sensor can send many connections to actuators or learning modules. Therefore, a pyramidal format is achieved when is used learning modules of which specific layer. PyramidNet Architecture offers levels of functionality to do a project. The learning modules can be organised in the hierarchic form to take out all of PyramidNet Architecture resources for emerging behaviours. The hierarchic structure is used to separate levels of complexity into incremental functionality. This approach can adequately select a large variety of learning modules without decreasing the performance. It is possible to mount ANS or CBS and even creates different kinds of architectures.

5 Constructing an ANS with PyramidNet Tool

In this section we briefly analyse all simple sequential steps to construct, train and obtain a finished ANS project using PyramidNet Tool. Firstly it is necessary to decide about the agent's body, intentions to solve specific tasks and methods to perform them. Next, it is necessary to draw a project in desktop. The project must have sensors; learning modules and actuators. Inspired in PyramidNet methodology, the whole project is organised in a pyramidal way. After designing the project, every sensor must receive a database file. This database file previously collects data around the environment by using the respective agent's sensor. It is then used to feed the learning modules for training.

Afterwards, interconnections are established among sensors, learning modules and actuators making a complete ANS. Each learning module must be adjusted to experiment for training if needed. Careful is needed in this step because a badly planned set up may led to the emergence of wrong reasoning processes. This step is the only that requires a little more knowledge about ANN. Finally, the last step is to set up a training sequence. Training sequence use a tabular training order (scheduler) to trigger the first and the last learning modules. Every learning module is scheduled for training based on hierarchical data flux. Maybe if the ANS project has a Stereotyped layer, it is possible to follow the training, how a teacher mode, by a short result train square. Certainly, user needs translate ANS to source-code to put in agents. The source-code is a copy from graphic modelling. Inside this code, user will find neural connection weights, an activation function, a compact main core, and some sensor and actuator matrixes. The code generated can be compiled using different c compilers because it uses ANSI C. In conclusion, these quick steps help the user to substitute large programs, time consuming projects and confusing line codes.

More details about the development of the Tool can be seen in [6][5]. The next section presents experiments which were created by using PyramidNet tool.

6 PyramidNet Tool Experiment

We present two different experiments to demonstrate fast development of ANS, feasibility and efficiency of the

Tool:

- "Container Capturer": emergence of behaviour in a dynamic environment;
- "Follow Wall - Search Recharging Point": heterogeneous learning modules and emergence of behavior.

6.1 Fist Experiment: "Follow Wall - Search Recharging Point"

The first experiment takes place into a rectangular arena with a white surface bounded by a black ribbon. The arena has several containers that can be on moving. Each container has a particular colour. The intention of the agent is to remove green containers out of arena. The background knowledge of the agent is to recognize a container, a green colour and a black ribbon. The actions of the agent are walking inside of the arena, searching for containers, identifying and removing green containers. The motivation of the agent is to continuously remove green containers if it founds different objects. The arena has unpredicted events and thus: there are some containers moving to different positions on the bounded arena; room light (brightness) may be changed; containers can be horizontal or vertical; and states and objects that are not containers can be used to confuse the agent.

We choose a Lego MindStorm robot [3] (Hitachi H8 processor) to run this experiment because it has all sensors and actuators needed to run this experiment. Its body has two traction motors for moving inside the arena, one pressure sensor for sensing containers and two light sensors to detect containers and the black ribbon. Next step, we draw the diagram with actions to be executed by the Lego robot, as seen in Figure 2. Based on Figure 2, the PyramidNet Tool was used to design an ANS project that is shown in Figure 3. The ANS project has four learning modules, three sensors, two actuators and respective interconnections.

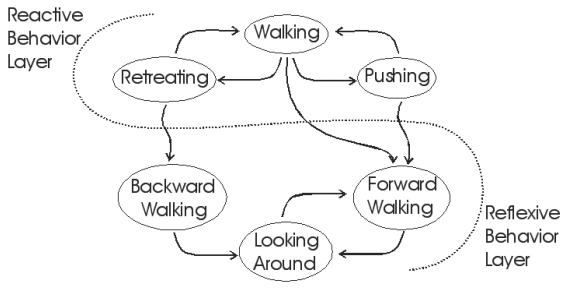

Figure 2: The Behaviour Task Plan Diagram

The Figure 2 shows a draft structure composed of two layers. The lower layer performs basic tasks, such as backward movement, looking around and forward movement. The upper layer performs complex decisions about continuously searching for containers, retreating for wrong containers and pushing the specified container out of arena.

In the Figure 3, the first learning module, Stereotyped Network, uses FeedForward topology trained with

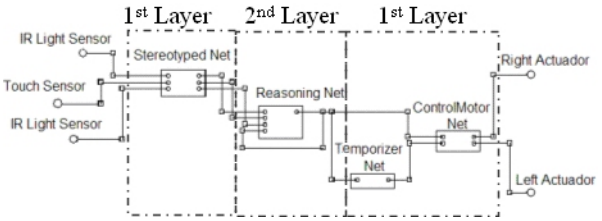

Figure 3: The ANS Project

the Backpropagation algorithm for reflexive behaviors. It receives signals from sensors and sends data to second layer. It receives signals from three sensors: two infrared sensors and one touch sensor. The first sensor, infrared, recognizes the container and colour. The second sensor uses pressure to detect a container and to control the velocity of access. The third sensor, infrared, detects the black ribbon on the floor. The second learning module, Reasoning Net, uses Recurrent networks to make reasoning decisions about detected events in the first layer and transmit them to first layer overlapping it. In this case, decisions consist for walking on, retreating on and pushing the green container out of arena. The third learning module, Temporizer Net, uses Feedforward ANN to simulate a simple temporizer that triggers turn around movements and back in first layer when a black ribbon is detected. Finally, the fourth learning module, Control Motor Net, uses Feedforward ANN to control the tracking motors for obeying the orders that come from superior layer. Concluding, a source-code was automatically produced to be applied to the robot. Figure 4 shows the Lego robot selecting a green container and moving it out of arena.

Figure 4: LEGO Arena

6.2 Second Experiment: "Follow Wall - Search Recharging Point"

This second experiment propose to build a complex behaviour architecture to perform actions in an agent in order to achieve a goal. Based on Silva's work [11], the intention of the agent is to walk inside the labyrinth and search a path to come near of luminous lamp. The background knowledge of the agent is to recognize a luminous lamp that is a recharging energy point. The actions of the agent are walking until find a luminous point and do not collide with walls or other objects. The motivation of the agent is continuously to go to a luminous point case its energy is coming down.

We choose this more ambitious project because it needs more elaborated behaviors in agents. Also, it is important to demonstrate the facility and quickly development to transform a handle project into a standardized ANS project. Careful and much attention must be taken into account to design an earlier project. Silva's project uses Khepera robot (Motorola 68331 processor) [5] to perform a more elaborated sequence of actions about Follow Wall and Search an Energy Point in a labyrinth room. Kephera is equipped with sensors of proximity/identification and two traction motors. About behavior in agents, Silva had settled two layers that are Stereotyped and Reactive, respectively. The new model of an ANS project is shown in Figure 5. ANS project has ten learning modules, eight sensors, two actuators and respective interconnections.

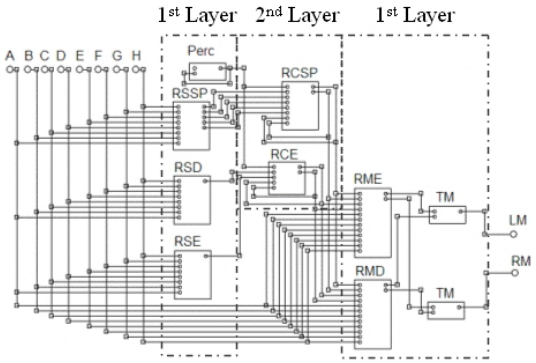

Figure 5: ANS project for "Follow Wall"and "Search Recharging Point".

The referred above diagram is composed by eight sensors (A, B, C, D, E, F, G, H), a Perceptron (Perc), an ANN of Sensory Follow Wall (RSSP), an ANN of Sensory Distance (RSD), an ANN of Sensory Energy (RSE), , an ANN of Control Follow Wall (RCSP), an ANN of Control Energy (RCE), an ANN of Walk Energy (RME) an ANN of Walk Distance (RMD), two ANN of Motor Controller (TM), and two actuators (LM, RM).

In this experiment 5, Khepera robot has default behaviour of walking in an environment, searching and following a wall while his energy is high. A central energy checker performs constant verifications about his consumed energy. If the energy falls to a minimum level, the default behaviour is inhibited and the behaviour of seeking for a bright spot while avoiding obstacles is triggered. Once the robot reaches the bright spot and recharges his energy, the default behaviour is enabled again. When these behaviors were implemented in the ANS, it used one Feedforward ANN to sense wall position relative to the robot's body, one Feedforward ANN to sense obstacles and another one to sense light direction. Two recurrent ANN receive their inputs from these Feedforward networks and from the recurrent perceptron that simulates the lowering of battery and is above them. The interesting point is that the whole behavioural code was produced using a few lines of code. These figures show the line code reduction in the project that was constructed by PyramidNet Tool.

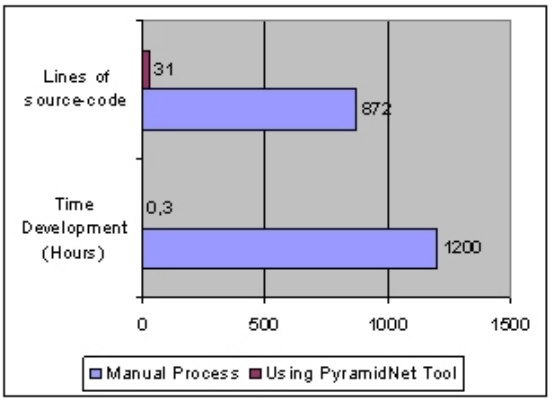

Figure 6: Development of an ANS

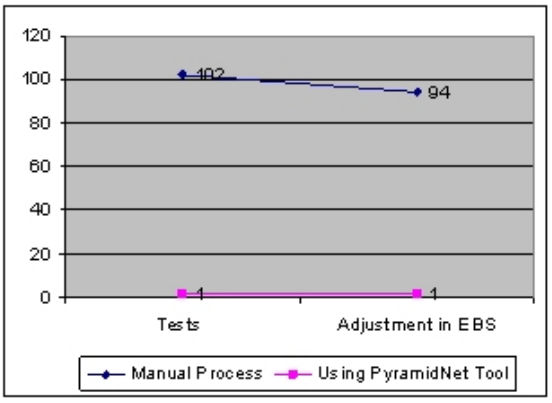

Figure 7: Tests and Adjustments in a final EBS code

7 Conclusão

This paper presented an connectionist approach for building Autonomous Agents using neural networks as the base to accomplish the reasoning process. The proposed tool - PyramidNet Tool - may be easily used in institutional educations because it is focused on common users, without professional knowledge on neural networks and a short agent development time.

PyramidNet Tool solves the strategy and behaviour problems of among agents in unpredictable environments

with the graphic emerging behaviour design in AA. The tool includes including graphical simulation among heterogeneous artificial neural network, sensors and actuators, generation of ready-to-use open-source code and support of several well-known ANN models - Backpropagation for Feedforward networks; Recurrent networks and SOM - Kohonen's maps.

PyramidNet Tool enables its users to understand the emergence of reasoning in Autonomous Agents within classroom settings. Thus, it provides opportunities for developing significant prototypes using computational paradigms in classroom, through which students can learn many of the trailing techniques of neural network-based system and behavior-based system development.

References

[1] FINE, T. L. *Feedforward Neural Network Methodology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[2] KOLB, B. . W. *An Introduction to Brain and Behavior*, 2nd ed. Worth Publishers Inc, New York, 2005.

[3] LEGO. Lego mindstorm hitachi h8: 3804 robotic invention system 2.0. `http://mindstorms.lego.com`, 2002.

[4] MANDIC, D. P. & CHAMBERS, J. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[5] REINALDO, F. A. F. Pyramidnet tool project. `http://www.inf.ufsc.br/~rei`, 2002.

[6] REINALDO, F. A. F. Projecting a framework and programming a system for development of modular and heterogeneous artificial neural networks. Dept. of computer science, Federal Univ. of Santa Catarina, Florianopolis, Brazil, Feb 2003.

[7] REIS, L. P. *Coordination in Multi-Agent Systems: Applications in University Management and Robotic Soccer*. Tese de Doutorado, University of Porto, 2003.

[8] RITTER, H. & KOHONEN, T. Self-organizing semantic maps. *Biol. Cyb. 61*, 4 (1989), 241–254.

[9] ROISENBERG, M., B. J. M. S. F. D. A. E. & VIEIRA, R. C. Pyramidnet: A modular and hierarchical neural network architecture for behavior based robotics. In *Proceedings of International Symposium on Robotics and Automation - ISRA 2004*, Queretaro, Mexico, August 25-27 2004, IEEE, p. 6.

[10] ROISENBERG, M. *Emergency of the intelligence in autonomous agents through inspired models in the nature*. Electrical engineering, Federal University of Santa Catarina, Florianopolis, Brazil, 1998.

[11] SILVA, F. A. Hierarchic neural nets for behavioural implementation in autonomous agents. Dept. of computer science, Federal University of Santa Catarina, Florianopolis, Brazil, 2001.

[12] WIRFS-BROCK, R. J. & JOHNSON, R. E. Surveying current research in object-oriented design. *Commun. ACM 33*, 9 (1990), 104–124.