



CrossGrid User Manual Guide

ANN Application

WP 1.3.2

Document Filename:	CG-UserManual
Workpackage:	WP 1.3.2
Partner(s):	CSIC
Lead Partner:	CSIC
Config ID:	cg-usermanual-v0.2
Document classification:	PUBLIC

Abstract: This is the user documentation for the Artificial Neural Network (ANN) training produced in CrossGrid subtask 1.3.2. In this document the objective of this application is explained, along with the way to obtain, build and install it.

Delivery Slip

	Name	Partner	Date	Signature
From	David Rodríguez	CSIC	Oct 2004	
Verified By				
Approved By				

Document Log

Version	Date	Summary of changes	Author
0.2	Oct 27th, 2004	First draft version	David Rodríguez, Irma Díaz

Contents

Copyright Notice	4
1 Introduction	5
1.1 Product Usage	6
1.2 Running the Product	6
2 Interface Reference Guide	10
3 Troubleshooting Q and A	11
4 Contact Information and Credits	12
5 EDG License Agreement	13

Copyright Notice

Please include a copyright notice to the effect of:

Copyright (c) 2005 by **CSIC**. All rights reserved.

Use of this product is subject to the terms and licenses stated in the EDG license agreement (. Please refer to attached license for details.

If your software, or documentation thereof, makes use of any externally copyrighted products, please include the following notices for each such product:

ANN Application is a registered trademark of **CSIC**. All rights reserved.

You must also include the following notice:

This research is partly funded by the European Commission IST-2001-32243 Project CrossGrid.

Chapter 1

Introduction

The GRID [1] framework provides access to large shared computing resources distributed across many local facilities.

High-throughput computing suits this approach: as an example, in the High Energy Physics (HEP) field thousands of independent simulation batch jobs, corresponding to a physics channel, are processed distributed across computing sites in France, UK, Italy, Spain, etc., and the resulting datasets integrated in a centralized database at CERN. In contrast, we consider here another complex computing problem in HEP that does not allow independent data processing and typically seems better suited for a high-performance environment: interactive HEP data analysis using Artificial Neural Networks (ANN).

ANN have become the “de facto” standard in HEP to address complex multidimensional analysis problems. As a relevant example, the recent search for the Higgs Boson at the Large Electron Positron Collider (LEP) at CERN used this technique in most of the channels, obtaining a significative performance improvement compared to traditional “sequential cuts” or “likelihood-based” analysis. Typically, these networks are trained on large samples of simulated events, but the network structure itself is moderately complex: in the DELPHI collaboration the Higgs search [2] in the hadronic channel used an ANN with 16 inputs, two intermediate layers of 10 hidden nodes each, and one output node (noted 16-10-10-1 architecture, see figure 1.1).

Training typically requires 1000 epochs, running on samples of about 500000 events pre-selected from total simulated samples of around 5 Million events. Independent samples of the same size are used to test the ANN. As no specific rules exist to build an optimal ANN, different architectures and input variables are used in the process to get the optimal final function. This demands a large computing power, because each training process requires from hours to days of a current PIII 1GHz CPU. However, physicist would like to use them in an interactive way, without having to wait so many time to try out different possibilities for the analysis. A much faster training of an ANN can be achieved either by using a much more powerful computer, or by running in a distributed way, either in a parallel machine, in a local cluster, or, to get more resources at the same time, at GRID scale.

The possibility of parallelizing ANN (see [3] for a good survey), with several approaches, has been studied for a long time. We follow an approach based on data partition, and have implemented the program in commodity hardware. A similar approach in a Beowulf cluster can be found in [4]. This article presents, in the first place, the work done to implement a distributed technique to train ANN in a local cluster. The plans and first steps taken to move it into the GRID framework are then outlined. This final part of the work will take place inside the CrossGrid European Project [5], and will be used for HEP interactive applications and with a similar approach in the meteorology field using Self Organizing Maps [6].

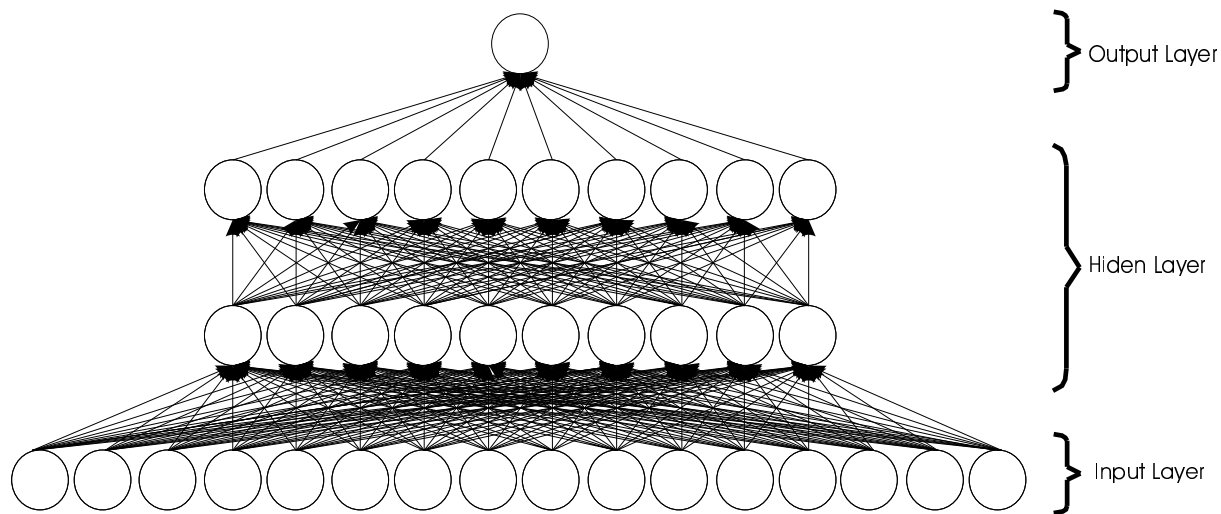


Figure 1.1: 16-10-10-1 ANN.

1.1 Product Usage

This module is one of the software products of CrossGrid Task 1.3. The main application in this module is an ANN training application. This application is meant to be executed in a distributed/parallel way. The application has been implemented using Message Passing Interface (MPI).

The objective of this software is to perform a typical neural network training such as the ones used in the search for the Higgs Boson in quasi-interactive way, i.e., reducing the physicists wait time so he can change the parameters and perform a new training in a few minutes instead having to wait hours/days as in the past. There are several auxiliary software products in the module. In particular, another application has been added. This application constructs an histogram on data files distributed in several locations.

1.2 Running the Product

You can get the application in two possible ways: getting the source code, building and installing it; or as a binary ready to install in RPM format.

1.2.1 Getting the code

The first thing to do is getting the code. You can download the source code from the CVS repository at FZK following the next steps:

```
$ export CVS_RSH=ssh
$ export CVSROOT=:ext:anoncvs@gridportal.fzk.de:/cvs/crossgrid
$ cvs -z3 co -d DESTDIR crossgrid/wp1/wp1.3-hep/wp1.3.1-ann
```

The `-d DESTDIR` argument is optional, but it could be more comfortable if you want just this module. It put the sources checked out in `DESTDIR` (you choose the name) instead of creating all the hierarchy.

Source code structure

The source code itself is in the `src` subdirectory. There several subdirectories contain the code for the different modules:

- `common`. Contains functions used by the application but clearly separable from it, and possibly reusable by other applications.
- `ann`. Contains the ANN training application. This is the main application of the subtask.
- `histogram`. Parallel histogram application.
- `ntuplefilter`. Ntuple filtering application.
- `util`. Contains utility programs.

1.2.2 Building

You will need autotools in your machine and you should execute:

```
$ ./autogen.sh
$ ./configure
$ gmake
```

The `configure` command accepts several arguments or parameters. For example, the `--with-rpm-dir=path-to-dir` optional argument lets you set the path for the RPM generation.

For the compilation, the first thing you should look at is the `compilation.mk` file in the `etc` directory. This file is included in all the Makefiles. You can put here common compilation definition and options. You should check that the definitions in it are correct for your installation. For example, by default the `mpi` compilation command is defined just as `mpicc`, assuming you have the executable in the path. If this is not the case, you should either, change the definition in `compilation.mk`, or add the `bin` directory of your `mpi` distribution to the path.

Furthermore, all the compilation stuff is fitted to run smoothly in CrossGrid “development workstations” and in the project autobuild machines. In other case you should carefully check 1.2.2.

In the `compilation.mk` file you will also see that there are two different compilations possibilities depending on the value of the environment variable `CG_MPICH_DEVICE`. This variable indicates the `mpich` device, and it can be either `“p4”` or `“g2”`.

If you just type, `gmake` at the top directory, it would compile everything first for the `p4` device and then for the `g2`. But, to have this working properly you should set correctly this stuff.

Build time dependencies

The following build time dependencies have been identified:

- `blas`
- `expat`
- `expat-devel`
- `gcc`

- gcc-g77
- lapack
- mpich (p4 or g2)
- w3c-libwww
- zlib
- zlib-devel

1.2.3 Installation

If you want to put the built executables in a particular directory (for example: `/opt/cg/bin`) you can type at the top directory:

```
$ gmake install prefix=/opt/cg
```

1.2.4 Getting RPM

Alternatively, you can also get an rpm from the CrossGrid autobuild system:

<http://gridportal.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp1/RPMS/>

The default installation directory is `/opt/cg`.

1.2.5 Running

To run the application with a local MPI:

- you need to have the MPI installed
 - check `mpi-run` path
- modify the `pgfile` to include the machines in your cluster, a 0 identifies the master, a 1 is used to identify slaves
- or use a `machines` file

First try in a local machine alone, using it as master and slave (two different processes will be open by MPI) and run the start-program script, `cg-ann.exe`, by typing:

```
$ mpirun -np 4 cg-ann.exe 4 16 10 10 1 100
```

It takes the data from the files specified in the dataset and uses it to train a neural network.

Now you can add more nodes to the `pgfile`, and you need copy executable to each slave machine. Be sure to have an account in all those machines. Scale the test gradually...

ANN application (`wp1_3_1-ann`) `cg-ann.exe` : The program needs the following arguments :

1. `a1`: number `n` of layers present in the ANN architecture (integer).

2. $a_2 \dots a_{n+1}$ number of neurons per layer (integer).
3. a_{n+2} number of epochs (integer).
4. a_{n+3} training
5. Number of epochs (integer).

It requires a compressed XML input file named `trainset.xml.gz`. This input file must be placed in the same directory as the executable and contains, in XML format, the list of input files that will be used to train the ANN as well as the number of events per file. Of course, the input files listed therein must be present also in the working directory. If they are not sented with the application, and so they are not there when the application starts, it would try to get them using CrossGrid's Replica Manager.

An example of JDL file used to commit the job is shown here:

```
Executable      = "cg-ann.exe";
JobType         = "MPICH";
NodeNumber      = 8;
StdOutput       = "std.out";
StdError        = "std.err";
InputSandbox    = {"cg-ann.exe", "trainset.xml.gz"};
OutputSandbox   = {"std.out", "std.err", "weights\_out.xml"};
Arguments       = "4 16 10 10 1 10";
```

The output of the program -as can be seen in the previous JDL file- is an XML file containing the final weights to be applied to the ANN. Furthermore, if graphical output has been chosen, two X11 windows will pop up, showing online the evolution of the ANN error versus the epoch number in one window and the separation power of the ANN applied over background (in red colour) and signal (in green colour) samples in the other window. The separation is shown for four different physical variables as can be seen in the plots below taken directly from the job at execution time.

Run time dependencies

The following run time dependencies have been identified:

- mpich (p4 or g2).
- gnuplot (if using X11 graphics version).
- EDG Replica Manager.

Chapter 2

Interface Reference Guide

Please provide an itemized description of the product user interface, if one exists. If the interface is a GUI, include screenshots for each interface element and describe each element (i.e. dialog box/window/menu) in a separate subsection. If the product provides a command-line interface, describe each command (preferably in an alphabetic manner), including syntax and usage examples.

Chapter 3

Troubleshooting Q and A

This chapter is intended as an in-depth explanation of potential problems and errors which might occur during the use of the software. Use the generally accepted Q&A layout, for example: Q: Upon submission of a job, the scheduler returns an error stating that no suitable computing element could be found for execution. A: Your job may require resources which are currently not available in the production testbed. Please revise your .jdl file to match the resources currently being offered (you may want to check <http://someaddress.net> for a list of currently available computing elements, along with their status).

Chapter 4

Contact Information and Credits

Good luck, and report back your experience to: drodrig@ifca.unican.es with CC to marco@ifca.unican.es and mrivero@ifca.unican.es.

Chapter 5

EDG License Agreement

This section should contain the EDG agreement, under which CrossGrid software is being licensed. If your software follows a different licensing pattern, replace this text with another license, appropriate for your software.

Copyright (c) 2005 CrossGrid. All rights reserved.

This software includes voluntary contributions made to the CrossGrid Project. For more information on CrossGrid, please see <http://www.eu-crossgrid.org>.

Installation, use, reproduction, display, modification and redistribution of this software, with or without modification, in source and binary forms, are permitted. Any exercise of rights under this license by you or your sub-licensees is subject to the following conditions:

1. Redistributions of this software, with or without modification, must reproduce the above copyright notice and the above license statement as well as this list of conditions, in the software, the user documentation and any other materials provided with the software.
2. The user documentation, if any, included with a redistribution, must include the following notice:

This product includes software developed by the CrossGrid Project (<http://www.eu-crossgrid.org>).

Alternatively, if that is where third-party acknowledgments normally appear, this acknowledgment must be reproduced in the software itself.

3. The names CrossGrid and CG may not be used to endorse or promote software, or products derived therefrom, except with prior written permission by cgoffice@cyfronet.krakow.pl.
4. You are under no obligation to provide anyone with any bug fixes, patches, upgrades or other modifications, enhancements or derivatives of the features, functionality or performance of this software that you may develop. However, if you publish or distribute your modifications, enhancements or derivative works without contemporaneously requiring users to enter into a separate written license agreement, then you are deemed to have granted participants in the CrossGrid Project a worldwide, non-exclusive, royalty-free, perpetual license to install, use, reproduce, display, modify, redistribute and sub-license your modifications, enhancements or derivative works, whether in binary or source code form, under the license conditions stated in this list of conditions.

5. DISCLAIMER

THIS SOFTWARE IS PROVIDED BY THE CROSSGRID PROJECT AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, OF SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE OR USE ARE DISCLAIMED. THE CROSSGRID PROJECT AND CONTRIBUTORS MAKE NO REPRESENTATION THAT THE SOFTWARE, MODIFICATIONS, ENHANCEMENTS OR DERIVATIVE WORKS THEREOF, WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADE SECRET OR OTHER PROPRIETARY RIGHT.

6. LIMITATION OF LIABILITY

THE CROSSGRID PROJECT AND CONTRIBUTORS SHALL HAVE NO LIABILITY TO LICENSEE OR OTHER PERSONS FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Bibliography

- [1] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infraestructure*. Morgan Kaufmann Publishers, 1999.
- [2] DELPHI Collaboration. *Search for the standard model Higgs boson at LEP in the year 2000*. Phys. Lett. B 499:23-37, 2001 [hep-ex/0102036]
- [3] Manavendra Misra. *Parallel Environments for Implementing Neural Networks*. Neural Computing Survey, vol. 1., 48-60, 1997.
- [4] D. Aberdeen, J. Baxter, R. Edwards. *98c/MFLOP Ultra-Large Neural Network Training on a PIII Cluster*. Proceedings of Supercomputing 2000, November 2000.
- [5] CrossGrid European Project (IST-2001-32243). <http://www.eu-crossgrid.org>
- [6] Kohonen, T. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 1995.