Architectures and Platforms for Artificial Intelligence—module 1

Programming assignment 2024/2025

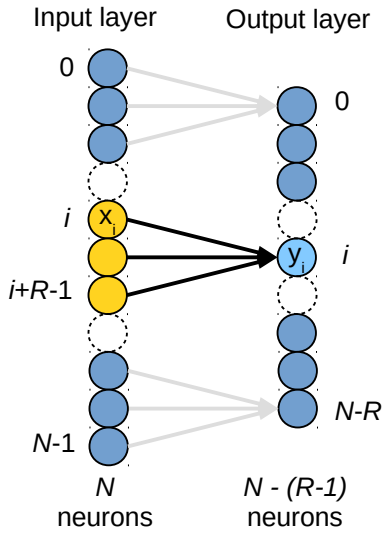
Moreno Marzolla

Version 1.0 2024-11-11

First version of this document

1. Specification

In this assignment you will implement a simple multi-layer Neural Network (NN). First of all, let us focus on the structure of two adjacent layers, shown in the following figure.



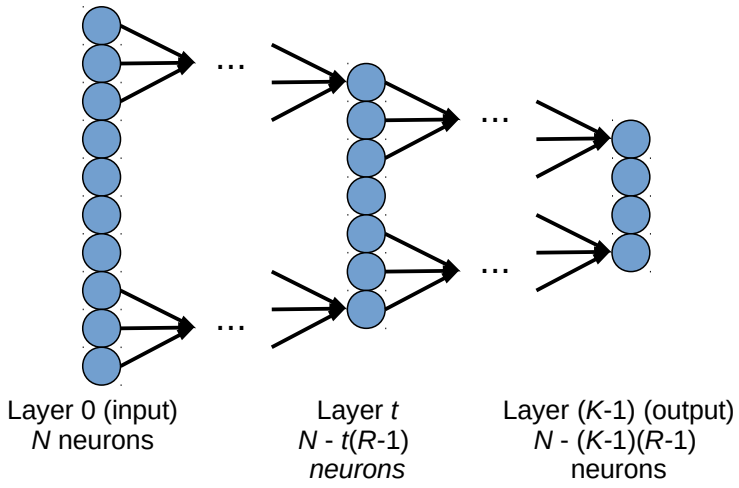
Let N be the number of nodes (neurons) in the input layer; each neuron holds a real value x_i, $i = 0, \ldots, (N - 1)$. The output layer has $N - (R - 1)$ neurons, each one holding a real value y_j, $j = 0, \ldots, (N - R)$. R is a compile-time constant, meaning that the value of R can be defined to some hardcoded constant or symbol in your source program. You may assume that R is small.

For each $i = 0, \ldots, (N - R)$, y_i depends only on the R values $x_i, \ldots x_{i+R-1}$ according to the following formula:

$$y_i = f\left(\sum_{r=0}^{R-1} x_{i+r} \times W_{i,r} + b\right)$$

where $f(x)$ is the sigmoid function $f(x) = 1 / (1 + e^{-x})$, $W_{i,r}$ and b are real number. The value of b is the same for all neurons across all layers. Note that our NN is sparsely connected, since the value of each neuron of the output layer only depends on R adjacent neurons on the input layer.

It is possible to build a K layer NN by taking K layers connected as described above. If layer 0 has N nodes, then layer t has $[N - t(R - 1)]$. The NN we consider in this assignment is a simple feed-forward network where the values on layer t, $1 \leq t \leq K - 1$, only depend on (a subset of) the values on layer $(t - 1)$.



2. What to do

The goal of this assignment is to write two programs, one using OpenMP and one using CUDA, that can efficiently evaluate a multi-layer NN as the one described above. Specifically, your program should contain a function/kernel that, given:

- the input values x_i and the length N of the input layer;
- the weights $W_{i,r}$;
- the constant b

computes the values y_i of the output layer. The function/kernel should then be called repeatedly to evaluate all K layers of a multi-layer NN. Note that the weights $W_{i,r}$ and the constant b might differ across layers; R is a compile-time constant and is the same for all layers.

To simplify testing and performance evaluation of the program, the size N of layer 0 and the number K of layers should be passed as command line arguments. The values on the input layer, the weights and the constant b can be defined as you wish (e.g., randomly generated, or read from input file(s), or something else).

I suggest that you start with the OpenMP version, which should be quite easy to implement. You can then base the CUDA version on the OpenMP one.

You are required to provide a written report (in English) at most six pages long, in PDF format. The report should provide a concise description of the parallelization strategies that have been used, and analyze the performance of the programs using the techniques described in the class. Note that the report should **not** describe the code line-by-line: I will look at the source files for that.

3. General rules

- This assignment must be completed individually by each student. In particular, sharing portions of the code and/or report with others is strictly forbidden.
- You can use code from the examples provided on the module Web page.
- The OpenMP program will be compiled with the gcc compiler on the department's HPC cluster using the `-std=c99 -Wall -Wpedantic` flags, and therefore must be C99 compliant; the CUDA program will be compiled with NVidia proprietary compiler without any special flag. You are not required to use the HPC cluster for development or testing, but you should make sure that your program at least compiles and runs there.

- I am available to provide clarifications on the project specification (this document); however, I can not (and will not) debug your code. This assignment is part of the final exam of this course, and is therefore important that you show a reasonable ability to program using the techniques discussed during the lectures.

4. Handing in the assignment

This assignment must be handed in using https://virtuale.unibo.it/; the system has been configured to accept a `.zip` or `.tar.gz` archive up to 20MB (this is a hard limit imposed by the "Virtuale" platform; should you have issues with that limit please let me know). The archive must contain the full source code, and the report in PDF format.

Please include a README file with compilation and execution instructions. Should your programs process an input file, please include a sample. You are not required to include all input files that have been used, e.g., for the performance evaluation described in the report.

The archive submitted through "Virtuale" should be named with your last and first name (e.g., `MarzollaMoreno.zip` or `MarzollaMoreno.tar.gz`); it should uncompress in a single folder with the same name (e.g., `MarzollaMoreno/`) containing all files submitted for evaluation, i.e., all, source files, sample input files (if any), and written report in PDF format.

The assignment can be handed in only once. Should you be unsatisfied with the final grade, you will be given a new project with new specifications.

Completed assignments will be graded as soon as possible, but that might take up to a few weeks in the worst case; please be patient. If you have deadlines (e.g., for renewing residence permits, for submitting your thesis and so on), please let me know; note however that I require at least ten working days to evaluate an assignment, so plan your submission accordingly.

5. Grading the assignment

The final grade will depend on the quality of both the code and the the written report. The code will be evaluated across different dimensions; in decreasing order of importance: correctness, clarity, efficiency. The source code should be properly formatted, and commented where appropriate.

The report will be evaluated by taking into consideration both its technical correctness, its clarity and proper use of the English language. The report will be as important as the code, meaning that a good implementation accompanied by a poor report will likely receive a mediocre grade.

You are free to provide extensions and/or improvements to the minimum requirements described above in this document. Note however that a good implementation of the basic features might receive better grades than a poor implementation of an extended set of features.

The final grade of the "Architectures and Platforms for Artificial Intelligence" will be the average of modules 1 and 2, rounded rounded to the nearest integer. Honors (*lode*) will be awarded at the discretion of the instructors of both modules for outstanding results.