Real time pattern recognition with artificial neural networks

G. Athanasiu^{1,2)}, P. Pavlopoulos^{1,3)}, S. Vlachos⁴⁾

Abstract

An artificial neural network (ANN) with local connectivities is proposed as a track identifier for high energy physics experiments. Realistic constraints from its possible hardware implementation as a first level trigger are taken into consideration. The performance of the ANN was evaluated with data of the CP-LEAR experiment, with very encouraging results. Such a ANN as a neural first level trigger (NFLT) may be relevant for LHC, SSC, HERA experiments.

(Submitted to Nuclear Instruments and Methods in Physics Research A)

1) CERN, Geneva, Switzerland

2) Univ. of Crete, Heraklion, Greece

3) Nuclear Physics Institute, NCSR Democritos Athens, Greece

4) Univ. of Liverpool, Liverpool, UK

1 Introduction

Several developments in the field of neurocomputing permit today reliable modeling of many physiological functions. The processes in the retina as well as the functionality of the first processing layers in the brain are of great interest for high energy physics (HEP) detectors. The formidable LHC luminosities make the use of neural first level trigger (NFLT) necessary. Given the complexity of the events at LHC we adopt an approach for the development of the NFLT which utilizes and abstracts functions from neurophysiology [1] and neurocomputing [2]. The simplest possible artificial neural network (ANN) is developed to identify the primary tracks in events from the CP-LEAR[3] experiment at CERN. This experiment was chosen because of its high luminosity and the required high on-line rejection of background events at the trigger level, two aspects relevant for LHC experimentation. Therefore an application developed for CP-LEAR should be easily upgraded to the next generation of experiments.

A single layer ANN was not capable of success rates higher than 80% on real events. Consequently a two layer ANN was developed with local connectivities. Notice that in a hardware implementation one ANN layer corresponds to one execution step (memory cycle) and our intention is to keep the number of such steps as low as possible. Even a reference frame coordinatization costs time and should be avoided if possible. The above mentioned ANN sees raw data and gives as an answer the position, the number of tracks and the association of the detector hits for each track. The constrained local connectivities accelerated the training phase and more importantly resulted in structures similar to the *on-centre-off-centre* cells. Extensive testing under several levels of noise proved the importance of the *off-centre* cells, whose contribution is observed to increase with noise. A similar effect is observed in the optical system of mammals as their dependence on night activity increases [4].

The above signify the possibility of duplicating certain functions of the visual system of mammals, at the complexity level of a HEP detector, with the characteristic noise robustness of distributed systems and at the speed of two processing steps. The whole network has 7.168 connections and 192 nodes. Its implementation in hardware is straightforward since the connectivity, per node, is low and the weights are integers. Moreover the ON-OFF formation makes the internal layer translationally invariant and therefore understandable and easy to implement. These results have shown the power of the algorithm and indicated that it can be used both for on-line triggering and subsequent analysis of the events.

2 Real time pattern recognition

The algorithms described in this article were developed taking into account their potential use in LHC experiments. The following constraints had to be taken into account:

1. The system has to be fast enough to be able to cope with the high LHC rates (15ns inter-bunch time, $10^{34} cm^{-2} s^{-1}$ luminosity) [5]
2. The detectors will be really complex, implying that there will be a large data set per event. Thus the number of connecting cables, between the front end electronics and any other system, should be minimized [6]

Therefore the track identification system has to be as simple as possible (to be fast and cheap) but on the other hand also very efficient and compact (to handle the high

track multiplicity and event rate). Only then it can be used as an effective data selection mechanism.

Because of the above mentioned restrictions, no pattern recognition system, based on conventional architectures, can be used efficiently in the LHC era. On the contrary, systems based on neural networks can perform adequately. This is because neural networks have an inherent capability of generalization, are not influenced by high levels of noise and execute their task in a few cycles. If in addition the pattern recognition method, to be used, has to be implemented in hardware, as a part of a fast trigger system, then neural networks are the best choice. Hardware performing the neural network algorithm can be easily built, and with the current VLSI technology it can be both fast and compact.

3 Simple perceptron

The simplest neural network configuration is the so-called *perceptron* [2, 7]. A *perceptron* is a network that consists actually of only one neuron connected directly to all the inputs.

The input of the network comes from the CP-LEAR data. The CP-LEAR detector [3] consists of 11 individual sub-detectors placed radially one outside the other. All the sub-detectors are divided in 64 angular sectors covering the whole 2π angle. Consequently an event can be represented by a 11×64 matrix called the hit-map (a row of the hit-map corresponds to a constant radius and a column to a constant angle).

In our actual model the *perceptron* had 55 input nodes. These 55 inputs corresponded to 5 columns of the event hit-map of the CP-LEAR detector. Each column consisted of 11 rows, as many as the number of radial sub-detectors of CP-LEAR. A typical event in this representation is shown in figure 1. To search the full hit-map for tracks, many *perceptrons* like that are needed apparently. The full network has 64 separate, but identical, *perceptrons*. Each one of these had its centre (the third out of the five columns) on a different sector of the hit-map. In this way maximum overlap between adjacent nodes was achieved, as each node had the first four, out of the five, input columns in common with the previous node and the last four with the following one. This overlapping is needed if one wants a track resolution of one sector from the system. All these *perceptrons* are identical as they have to do the same classification. Therefore only one *perceptron* is constructed and then repeated 64 times. In the learning phase that means there are 64 examples per event presented to the network. These 64 examples are seen by the 64 individual *perceptrons* from each hit-map.

The input-output of the system was digital having a value of -1 , for no hit (in the input) or no track (in the output), or 1 , for a hit or a track respectively. The weights themselves were allowed to take any real value, initialized by random numbers in the range -1.0 to 1.0 . That implies that the transfer function of the output node was $\text{sign}(x)$ (that is $+1$ if $x > 0$ and -1 if $x < 0$).

In the learning phase, the weights W_i were updated according to

$$\Delta W_i = \varepsilon \sum_{\mu} (T^{\mu} - O^{\mu}) x_i^{\mu} \quad (1)$$

where x_i^{μ} are the input hits of example μ , $O^{\mu} = \sum_i W_i^{\mu} x_i^{\mu}$, and T^{μ} the expected answer of the network (ε is a constant determining the learning speed). Note that the weights were changed once every time the whole example set was scanned.

Figure 1: The hit-map of a typical CP-LEAR event

The learning example set consisted of 400 events. The tracks, the objects the network had to identify, were identified by human experts. This was done in order to teach the network only with the best possible input, avoiding any imperfections of any existing pattern recognition algorithm.

A constant bias b was introduced so that $O^\mu = \sum_i W_i^\mu x_i^\mu - b$. With $b = 5.0$ and $\varepsilon = 0.001$ the network was exposed to the learning set 1200 times. The time evolution of a typical weight is plotted in figure 2. The time evolution of the cost function C , defined as

$$C = \sum_{\mu} (T^\mu - O^\mu)^2 \quad (2)$$

is plotted in figure 3. In figures 2 and 3 time was incremented when the full learning set had been presented to the network, and an update of the weights has taken place (note that the time evolution is plotted for the first 60 time steps, as later the variation of the weights is very small). The performance of the trained network with the learning and the test set was similar, resulting to 10% inefficiency of track detection and 3% probability of false track identification. Inefficiency of track detection is the probability that a real track is not detected by the neural network. Probability of false track identification is the probability that the network finds a track where there isn't any really.

4 Multi-layer networks

In order to improve the results obtained with the *perceptron* a two layer NN with local connectivities was introduced, with two hidden nodes and a single output for every 11×5 inputs. (see figure 4). The input to the system was the same as for the *perceptron*. Each individual network receives information from five consecutive sectors of an event hit-map again as in the case of the *perceptron*.

For the learning process the standard Back Propagation algorithm [8] was used. In a two layer NN the corresponding cost function C is again defined by equation 2. The difference is that in this case the output O^μ (for the example μ) is defined as

$$O^\mu = f \left(\sum_j w_j f \left(\sum_i w'_{ij} x_i^\mu \right) \right) \quad (3)$$

Figure 2: The time evolution of a typical *perceptron* weight

Figure 3: The time evolution of the cost function C of the *perceptron*

Figure 4: The structure of a three layer network with two hidden nodes

The weights from the hidden nodes to the output one are w_j and the ones from the input to the hidden nodes are w'_{ij} , where the indices i, j run over the number of input and of hidden nodes, respectively. The function f is the so-called response function of the neurons. The purpose of the learning algorithm is again to minimize the cost function C .

The algorithms was, though, slightly modified to take into account the non-equidistribution of the examples in the two desired classes (track, no track). Each event presented to the network consists of 4 track and, given the CP-LEAR detector segmentation, 60 non-track examples. This implies that most of the times the weights were pulled to the region where they force the system to respond that there was no track. To compensate for that particular distribution of examples the learning constant ε was substituted by formula 4

$$\varepsilon \leftarrow \left(1 + \frac{T^\mu - O^\mu}{|T^\mu - O^\mu|}\right) \varepsilon_R + \left(1 - \frac{T^\mu - O^\mu}{|T^\mu - O^\mu|}\right) \varepsilon_L \quad (4)$$

where $\varepsilon_R = 1.0$ and $\varepsilon_L = \varepsilon_R/20$. Formula 4 introduces different learning constants for the cases where the network answer should be -1 and those where it should be 1 .

The same learning and test example sets were used as for the *perceptron*, to facilitate the comparison. The weights were initialized with random numbers in the range -0.1 to $+0.1$. The response function of all the neurons was the typical sigmoid function $\tanh(x)$. A ‘temperature’ factor (β) was introduced, so the argument of $\tanh$ was $\beta \sum_i W_i x_i^\mu$ rather than $\sum_i W_i x_i^\mu$. Assuming $\sum_{i=1}^N W_i x_i^\mu \sim N$ with $-1 < x < 1$ (N being the total number of weights), a value of $1/50$ was choosen for β . The ‘temperature’ factor helps to accelerate the learning by allowing easier transition between classes for a misclassified example.

In order to find the optimal weight values, the network was exposed to the learning set 800 times. The time evolution of a typical weight, the cost function and the output distribution of the network are plotted in figures 5,6,7 respectively (notice that the time

Figure 5: Time evolution of a typical weight for the two layer network

axis in these plots goes only up to 80 as in later epochs the changes being much smaller).

In figures 8 and 9 the distribution of the activation of the output node is shown for the learning (with solid line) and the test (with dashed line) set for the actual track or non-track examples.

The weights derived from the learning phase were real numbers. They were later truncated to the closest integer numbers and no deterioration of the system's performance was noticed.

The final performance of the network is described by

- 0% inefficiency of track detection
- 6% probability of false track identification

for both the learning and test set. The fact that all real tracks are detected is very important for any practical use of this algorithm. The examples that were classified wrongly as tracks, fall in two categories. They were either noisy patterns that the network considered as tracks, or patterns of real tracks that also a few adjacent networks recognised. The later effect was observed in the case of inclined or curved tracks that span many sectors, and caused many tracks to be identified twice. However additional topological constraints, such as the requirement that several adjacent sectors should not have a track at the same time, reduce the false track identification. In addition further reduction can be achieved if a higher output activation is demanded when many sectors, close together, see a track.

Since the noise level of the CP-LEAR experiment is low, we implemented artificially random noise in order to study the networks noise robustness. The addition, in the test sample, of several noise levels in the range from 5% to 10% per channel, resulted in substantial deterioration of the performance of the network. This could be understood as the training was done only with clean events. After a training with noisy events the noise robustness improved but its performance never reached the one without noise. In particular for a test

Figure 6: The time evolution of the cost function for the two layer network

Figure 7: The activation of the output node for the learning set as a function of the number of learning cycles

Figure 8: The activation of the output node for the examples with tracks, with a solid line for the learning set and with a dashed one for the test set

Figure 9: The activation of the output node for the examples without tracks, with a solid line for the learning set and with a dashed one for the test set

Figure 10: Distribution of weights along a row for hidden node 1

set of events contaminated with 50% background noise over the signal, the following results were obtained:

- 0% inefficiency of track detection
- 20% probability of false track identification.

The addition of an extra node in the hidden layer did not result in any improvement of the performance of the network. By applying the above mentioned additional constraints the probability of false track identification was reduced to less than 3%.

5 On–Off cells and their interpretation

It is observed that the final weights, of the system, from the input to the hidden layer have a structure that can be seen in figures 10 and 11. In these plots (fig. 10, 11) the averages over the 11 weights for each of the 5 columns are shown for each of the hidden nodes.

As can be seen from figures 10 and 11 the two hidden nodes perform complementary functions. The first hidden neuron performs the function of the so-called *on-cell* filter. The second one performs the *off-cell* function. The *on-cells* manage to gather, in a way, scattered hits, lying close together, to form a single track out of them. On the other hand the *off-cells* pull apart slightly displaced hits to make two well separated tracks out of them.

The high number of double counts that was observed in the results was coming exactly from the strong influence of the *off-cell* on the decision. The *off-cells* increase very much the discriminating power of the system.

It is interesting to notice that the *on-cell*, *off-cell* structure is the one found actually in the first processing layer of cells in the retina of mammals (ganglion cells). As we know [4] this structure results in an important data reduction and facilitates feature extraction, before transmission to the back of the brain for further processing. It is encouraging that the proposed algorithm converged intrinsically to a solution that was chosen in nature by natural

Figure 11: Distribution of weights along a row for hidden node 2

selection.

6 Conclusions

It has been demonstrated that a two-layer neural network, using a back-propagation learning algorithm, is efficiently detecting tracks in the CP-LEAR experiment. The search for the optimal first level trigger architecture started from the simplest ones. The minimal working configuration consists of one hidden layer with two nodes. Some extended multi-layer networks were investigated but no real performance improvement could be seen.

It was also shown, by using separate learning and test example sets, that this network has a good generalization capability. This results in an excellent network for specific applications e.g. fast triggering.

The performance of the proposed neural network architecture was essentially the same when real or integer values for the weights were used. Therefore it can easily be realized in hardware using VLSI technology. This is so because the number of connections is small and only integer arithmetic is needed. As a consequence high parallelism and track identification in two clock cycles is achieved. This allows the definition of the track multiplicity of an event in less than 150ns, using current technology.

Without major modifications the same algorithm can be effectively used for on-line triggering and off-line analysis.

REFERENCES

- [1] H. B. Barlow and P. Foldiak, The computing neuron, Addison-Wesley, New York 1990.
- [2] D. E. Rumelhart, J. L. McClelland and the PDP research group, Parallel distributed processing, The MIT press, Massachusetts 1986.

- [3] CP-LEAR Collab., R. Adler et al., CP-LEAR proposal, CERN/PSCC/85-6 (1985).
- [4] D. C. VanEssen and C. H. Anderson, Information processing strategies and pathways in the primate retina and visual cortex, in: Introduction to neural and electronic networks, Academic Press, Orlando 1988.
- [5] The LHC design group, Design study of the large hardron collider, CERN 91-03, Geneva 1991.
- [6] N. Ellis, S. Cittolin and L. Mapelli, Signal processing, trigger and data acquisition, in: Large hardron collider workshop, Aachen 4-9 October 1990, eds. G. Jarlskog and D. Rein, CERN 90-10, Geneva 1990.
- [7] F. Rosenblatt, Principles of neurodynamics, Spartan Books, New York 1962.
- [8] B. Müller and J. Reinhardt, Neural networks, Springer-Verlag, Berlin 1990.