Level-Two Triggering via a Neural Net with Linear Discriminant Functions

Hakl, František
1997

Level-Two Triggering via a Neural Net with Linear Discriminant Functions

František Hakl and Marcel Jiřina [*]

E-mail: hakl@uivt.cas.cz, marcel@uivt.cas.cz

July 31, 1997

Abstract

The electromagnetic cluster algorithm based on the method of neural net for second level trigger is considered. The target is to separate two kinds of events (em showers and jet showers) as good as possible. For useful data (electrons) the "accept" signal is generated. Within a concept of region of interest quidance the level two trigger uses full granularity information from calorimeter and preshower inside one or more regions defined at first level. So each event is represented by one or more matrices of integers corresponding to the energy release. These data are concentrated and then classified by a neural classifier. There is need of rather high speed processing as the data come with frequency of 100 kHz. The neural net with layered architecture and step nonlinearity is described. The results showing the efficiency of neural net approach are presented.

You can obtain programs and data sets mentioned in this paper in an electronic form on the following ftp address: ftp://ftp.uivt.cas.cz/pub/atlas .

[*]Institute of Computer Science, Academy of Sciences of the Czech Republic

Contents

List of Figures

List of Tables

1 Introduction

In both low and high luminosity running of LHC the selection of events containing at least one or two electromagnetic (em) clusters is proposed [atl94]. High-transverse-momenta electrons or photons indicate rare processes and give a possibility to distinguish them from other physics processes. Em-cluster trigger use selection cuts with transverse momenta ($p_\perp$) lowest limit at 20 or 30 GeV, which is much lower than the $p_\perp$ limit for jet triggers (150 or 300 GeV). At $p_\perp \sim 20$ GeV the cross section is dominated by jets and the aim of em-trigger is to recognize em showers from jet showers.

The information from calorimeters may be treated as signals or as pictures. Signals are rather complicated multichannel ones. If the information is considered to be a picture, it has rather small number of pixels (20×20) of a gray scale.

Electrons are about one million times more rare events than jets. Therefore, jets must be filtered out with a high quality, while the electron loss should be as small as possible. The LHC will provide a bunch-crossing every 25 nanoseconds (i.e. with frequency 40 MHz). Then triggering must be performed just with this frequency. It proceeds in two steps. First, the Level 1 trigger must reduce the frequency of events from 40 MHz to 100 KHz, i.e. 400 times. Then the Level 2 trigger reduces frequency 50 times.

2 Level-Two Triggering

Electromagnetic clusters selection in region of interest (RoI) concept proposed at ATLAS detector [atl94] is done at two levels. The first level trigger (LVL1) is proposed to operate globally in the whole calorimeter, while the second level (LVL2) will operate with restricted data from one or more RoIs defined by LVL1. While LVL1 works with granularity ($\triangle\phi \times \triangle\eta$) 0.1×0.1, the LVL2 use the full granularity of calorimeter cells (which is for em calorimeter 0.025×0.025). The ROI can be described by a pair of matrices; their elements correspond to individual sensors in the electromagnetic and hadronic calorimeters and their values are given by energies measured by sensors. Every ROI forms something like two gray-scale pictures consisting of (currently) 20×20 points (pixels) each. These pictures are to be classified as jets or electrons.

To achieve a rate reduction of a factor ~ 100 at LVL2, it is necessary to operate on full-granularity data from several subdetectors and to combine their information. It is not sufficient to repeat LVL1 cluster algorithm with improved resolution [atl94].

LVL2 data processing is divided into three steps with, in each case, a reduction in the amount of data to be passed to the following step:

1. Feature extraction.

2. Features for one region of interest from several subdetectors are combined to form an object (e.g. the particle type).
3. Objects for all regions of interest are combined to form a global event decision.

The use of neural algorithm has been proposed in [BvK$^+$95b] as an alternative for step 2 above. In this study a suitable feature extraction — preprocessing in combination with neural net is considered.

Neural nets are capable of learning and then of identifying of rather complex pictures or signals [Hak93], [LG19]. The essential problem is a high required speed. The neural algorithm must respect both complexity and speed of the process.

3 Data Concentration and Feature Generation

Electromagnetic detector provides input data in the fine granularity 0.025×0.025. These data are arranged in a matrix of 20×20 energy values. The original picture 20 times 20 pixels contains too many parameters to be classified in a short time. For neural net it would be $20 \times 20 = 400$ input values. This is principally possible but net would be too complex and slow. For three-layer net with one neuron in the first layer and two neurons in the second and the third layer for each input has total 2000 neurons, i.e. 25 times more than 16 inputs net. Hence, either a part of the picture, or some its features must be used. There are many other ways how to concentrate the original data. They can be e.g.:

- row, column, or diagonal sums that form directional histograms of the event,
- concentric squares or semi-circles around the event
- an extracted part of the picture with the prescribed shape.
- to split the original matrix of 20×20 values to 4×4 (16) submatrices of 5×5 values each. The data from a submatrix is summed up to a single value (subsum). Thus one can form a new matrix of 4×4 subsums.

Most features require that the event must be first located. It implies finding the maximal value, the center of gravity, or the mean of the event. This is an easy procedure for modelling on a computer, but repetitive computations must be avoided in a practical realization. The problem may be solved by using massive parallel electronic structures that perform all required summations and comparisons simultaneously. Centering of the integral features around the mean or maximum may essentially simplify the following classification.

4 Triggering Algorithm Efficiency

In this chapter we first show efficiency of triggering algorithm not based on neural nets, then the behavior of the neural net. Neural approache has been tested using simulated data generated in CERN by ATLAS physics simulation group. The data are the data after LVL1 triggering and we use them to study the behavior of the neural net designed for LVL2 triggering.

Let us consider that $U = E + J$ events arises in a time unit T. Here E denotes electron events considered to be useful and should not be lost; while jet events J should be suppressed as much as possible. Let the portion of E-events in the mixture be

$$P_0 = \frac{E}{E + J}.$$

It is supposed that this portion is very small, typically less than one millionth.

The next step of event classification in ATLAS is designed to treat data with some limited frequency. According to [atl94], p.140, the output frequency of LVL1 (the LVL1 rate) for isolated cluster or two isolated clusters is $\lesssim$ 24 kHz and $\lesssim$ 12.5 kHz for high and low luminosity, respectively. The LVL2 rate is $\lesssim$ 0.6 kHz (0.3 kHz for 1e, 0.1 kHz for γ , 0.2 kHz for 2e) for high luminosity and $\lesssim$ 0.28 kHz (0.2 kHz for 1e, 0.2 kHz for γ , 0.01 kHz for 2e , and 0.01 kHz for 2γ) for low luminosity. Then the triggering system should reduce the original number of events by at least reduction factor K. This factor is approximately $K = \frac{25}{0.6} \doteq 42$ for high luminosity and $K = \frac{12.5}{0.28} \doteq 48$ for low luminosity.

Let the triggering process have the following parameters: j_{err} is the J-events error, i.e. the portion of J-events going through the trigger misinterpreted as E-events. It is the first kind of an error. e_{eff} is the E-events efficiency, i.e. the portion of E-events going through the trigger correctly recognized as E-events. New number of events after the trigger corresponding to the original number of events U is now $U_n = e_{e_{eff}} \cdot E + j_{err} \cdot J$. The arrangement with trigger is shown in the schema:

$$\begin{array}{ccr} E + J & \longrightarrow & e_{eff} \cdot E + j_{err} \cdot J \\ \text{(source} & & \text{trigger)} \end{array}$$

The event frequency reduction ratio due to triggering is

$$r \overset{def}{=} \frac{U_n}{U} = \frac{e_{eff} \cdot E + j_{err} \cdot J}{E + J} \overset{def}{=} \frac{1}{R}.$$

The frequency of events is then reduced by frequency reduction ratio r and it must hold that

$$\frac{1}{r} \geq K.$$

The amount of data after triggering must be less or equal to the amount which can be recorded in the next level. All E-events will pass into the next level (number of them is $e_{eff} \cdot E$). This portion of E-events in the mixture after triggering is

$$P_1 \overset{def}{=} \frac{e_{eff} \cdot E}{e_{eff} \cdot E + j_{err} \cdot J} > P_0.$$

This is Y-times more than in the input of trigger,

$$Y \overset{def}{=} \frac{P_1}{P_0} = \frac{e_{eff}}{j_{err}}. \tag{1}$$

We call Y as electrons enrichment factor, or efficiency.

In the summary, the triggering is characterized by two parameters:

- event frequency reduction ratio $r \simeq j_{err}$, $\frac{1}{r} \geq K$,
- electrons enrichment factor Y which should be as large as posible.

The first tests of the trigger efficiency of the neural net method are based on 17 GeV Level-1 filtered dijet events data.

4.1 Neural algorithm efficiency

The neural net used for triggering has a layered architecture (see figure 3). Every layer consists of a set of switches and modulators and one summator and comparator. The number of switches and modulators in every layer is equal to the number of input data, i.e. to the number of processed signals or their features. The modulators perform a multiplication of input signals by weight coefficients. Switches are used for choosing the proper set of values of weight coefficients according to the results from the previous layer. They ensure a step nonlinearity of the network by switching among several sets of weight coefficients.

The original data are of two kinds: without pile-up and with pile-up (see [atl94], section 2.7.1.1.). In the next the symbol starting with 2- denotes original data with pile-up, otherwise the original data without pile-up has been used. The original data of 20x20 numbers from the RoI in the EM calorimeter for each event (pattern) does not suit as direct input for neural net. The data must be, first, preprocessed in a suitable way. The efficiency of neural algorithm depends on several parameters:

1. The kind of data concentration (and then the number of inputs). There are the possibilities as follows:

   KR 10 concentric squares. For each square the sum of corresponding points is computed. Thus 10 values for the input of the neural net are obtained.

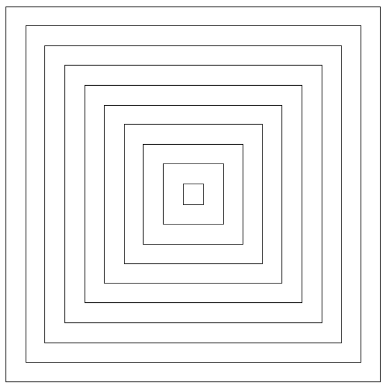


Figure 1:

CL The clusters as in the cluster algorithm. The sums over six areas as shown in the figure are taken as the inputs of the neural net.

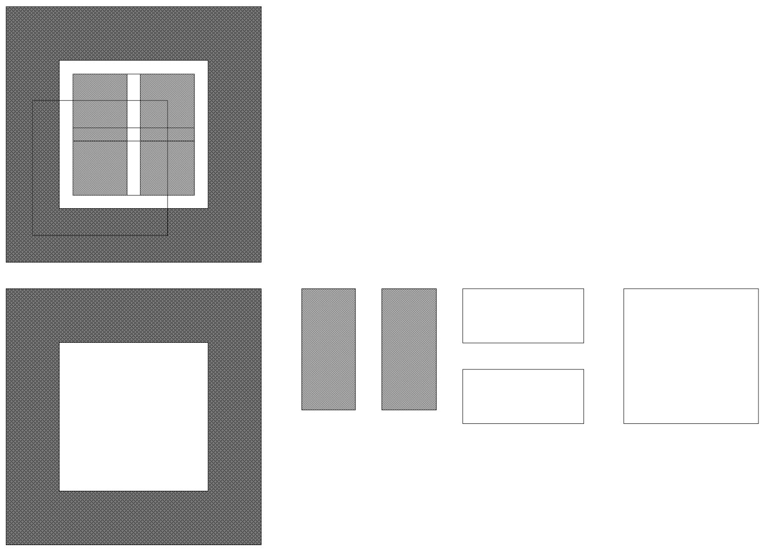


Figure 2:

CT 16 values each arising as a sum over 5×5 subsquare of the 20×20 initial data.

COL The sums over 5 columns of the square 5×5 centered around the maximum are taken as the inputs of the neural net.

ROW The sums over 5 rows of the square 5×5 centered around the maximum are taken as the inputs of the neural net.

CIR The sums over five concentric squares are taken as the inputs of the neural net. The center is situated in in the point with the maximal signal value.

PIX 25 pixels of the square 5×5 centered around the maximum are taken as the inputs of the neural net.

2. The number of neurons in each layer, i.e. number of positions of the switch S_i (see Figure 3). This parameter together with the number of layers describe the architecture of the net used.

3. The number of layers of the net.

When testing the net with given set of data, it depends also on splitting of the original set into the learning (training) set and the testing set. Three methods has ben used for the splitting:

LT The learning set and the testing set have the same size. These sets are generated so, that the first, third and other odd samples belong to the learning set and the even samples belong to the testing set.

LTTT The learning set is $\frac{1}{4}$ and the testing set is $\frac{3}{4}$ of the entire set of the samples. These sets are generated so, that the first sample belongs to the learning set, the second, third and fourth sample belong to the testing set. The following sample belong to the learning set and next three samples belong to the testing set and so on.

TTLTTT The learning set is $\frac{1}{6}$ and the testing set is $\frac{5}{6}$ of the entire set of the samples. These sets are generated so, that the third sample belongs to the testing set and the first, second, fourth, fifth and sixth sample belong to the learning set. The next six samples are split in the same way and so on.

It is easily seen that there are lot of different combination of preprocessing methods, the neural net architectures and ways of splitting of the original set of data. All these combinations has been tested. In Table 1 a part of results is shown.

In the first column of Table 1 the kind of original data, the way of the preprocessing and the architecture of the neural net are given as follows:

Symbol starting with "2-" denotes original data with pile-up, otherwise the original data without pile-up has been used. The following two or three letters (KR, CL, CT, COL ...) denote the kind of the data preprocessing.

The architecture of the neural net is given mostly with combination of numbers:

notation	The number of neurons in the second, third, fourth,. . . layer
2	22222
3	33333
4	44444
8	88888
1TO5	234567
7TO1	765432
424	43234
626	626262
242	23432
4242	424242

The number of really used layers is given in the second column of Table 1. The next column pairs give the values of compression and of the efficiency for three methods which has ben used for the splitting the data set (LT, LTTT and TTLTTT). The minimal value of the efficiency for three methods of the splitting in the row is given in the last column. According this value the rows in Table 1 are sorted starting from the largest value.

Table 1: Results on testing data – maximal enrichment factor

Explanation: Strings in the first column of the table are names of corresponding result files and denotes data and net architecture used. The names starting with 2- denotes that the data used was the data with pile-up, other are without pile-up. Next part of names denotes the kind of preprocessing ("KR","CL","CT","CIR" etc.). The last part of the name denotes the architecture of the neural network. Column pairs LT, LTTT and TTLTTT denote choice of learning and testing data.

data set splitting		LT		LTTT		TTLTTT		
data & nets	layers	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	$min\{\frac{e_{eff}}{j_{err}}\}$
2-KR	0.002	48.1	0.001	51.6	0.001	71.4	48.1	
2-KR242	3	0.002	61.0	0.001	56.3	0.001	48.0	48.0
2-KR2	3	0.002	61.0	0.001	56.3	0.001	48.0	48.0
2-KR1TO5	3	0.002	61.0	0.001	56.3	0.001	48.0	48.0
2-KR4242	3	0.002	45.7	0.001	61.0	0.001	60.9	45.7
2-KR424	3	0.002	45.7	0.001	61.0	0.001	60.9	45.7
2-KR4	3	0.002	45.7	0.001	61.0	0.001	60.9	45.7
2-KR2	5	0.002	48.1	0.003	44.5	0.001	51.5	44.5
CL2	4	0.002	47.1	0.001	57.1	0.001	43.8	43.8
CL242	3	0.002	41.6	0.001	60.4	0.001	43.8	41.6
CL2	3	0.002	41.6	0.001	60.4	0.001	43.8	41.6

continued on next page

continuation of table 1

		LT		LTTT		TTLTTT		
data & nets	layer	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	$min\{\frac{e_{eff}}{j_{err}}\}$
CL1TO5	3	0.002	41.6	0.001	60.4	0.001	43.8	41.6
2-KR4242	4	0.002	46.9	0.003	47.5	0.001	38.6	38.6
2-KR2	7	0.002	51.6	0.004	36.7	0.001	55.0	36.7
CL8	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL7TO2	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL626	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL4242	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL424	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL4	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL3	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL242	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL2	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL1TO5	2	0.004	36.7	0.001	40.6	0.001	42.7	36.7
CL2	6	0.004	35.1	0.001	53.8	0.001	40.5	35.1
CL2	5	0.004	35.1	0.001	52.7	0.001	42.7	35.1
2-KR2	6	0.002	51.6	0.001	34.0	0.001	55.0	34.0
CL2	7	0.004	32.9	0.001	52.7	0.001	40.5	32.9
2-KR242	4	0.002	46.9	0.001	30.5	0.001	73.8	30.5
2-KR1TO5	4	0.002	46.9	0.001	30.5	0.001	73.8	30.5

Table 2: Results on testing data – reduction ratio 0.02

Explanation: Strings in the first column of the table are names of corresponding result files and denotes data and net architecture used. The names starting with 2- denotes that the data used was the data with pile-up, other are without pile-up. Next part of names denotes the kind of preprocessing ("KR","CL","CT","CIR" etc.). The last part of the name denotes the architecture of the neural network. Column pairs LT, LTTT and TTLTTT denote choice of learning and testing data.

data set splitting		LT		LTTT		TTLTTT		
data & nets	layers	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	$min\{\frac{e_{eff}}{j_{err}}\}$
2-KR2	5	0.020	18.3	0.020	18.8	0.020	20.0	18.3
2-CT2	5	0.020	18.3	0.020	19.0	0.020	21.6	18.3
2-KR2	7	0.020	18.0	0.020	19.0	0.020	19.0	18.0
2-CL2	5	0.020	18.3	0.020	18.0	0.020	26.7	18.0
2-CL626	4	0.020	18.7	0.020	17.9	0.020	22.8	17.9
COL242	3	0.020	18.0	0.020	18.0	0.020	17.8	17.8
COL2	3	0.020	18.0	0.020	18.0	0.020	17.8	17.8
COL1TO5	3	0.020	18.0	0.020	18.0	0.020	17.8	17.8
2-CL2	7	0.020	19.0	0.020	17.8	0.020	23.0	17.8

continued on next page

continuation of table 2

		LT		LTTT		TTLTTT		
data & nets	layer	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	$min\{\frac{e_{eff}}{j_{err}}\}$
2-CL2	6	0.020	19.1	0.020	17.8	0.020	24.7	17.8
2-CT242	3	0.020	17.7	0.020	18.1	0.020	20.5	17.7
2-CT2	3	0.020	17.7	0.020	18.1	0.020	20.5	17.7
2-CT1TO5	3	0.020	17.7	0.020	18.1	0.020	20.5	17.7
2-CL2	4	0.020	18.8	0.020	17.6	0.020	25.7	17.6
2-CL4242	4	0.020	17.6	0.020	18.7	0.020	25.7	17.6
2-CL626	3	0.020	24.1	0.020	17.4	0.020	18.7	17.4
2-CL424	5	0.020	18.5	0.020	17.4	0.020	17.5	17.4
2-CL4242	5	0.020	18.4	0.020	17.3	0.020	24.9	17.3
2-CL4242	3	0.020	17.9	0.020	17.2	0.020	23.5	17.2
2-CL424	3	0.020	17.9	0.020	17.2	0.020	23.5	17.2
2-CL4	3	0.020	17.9	0.020	17.2	0.020	23.5	17.2
2-CL242	3	0.020	17.0	0.020	17.5	0.020	22.0	17.0
2-CL2	3	0.020	17.0	0.020	17.5	0.020	22.0	17.0

Let us discuss the results in Table 1. In fact, the last column of this table shows the worst case of Y, i.e. the efficiency for the neural net trigger system. This worst case arose from the procedure described in the preceding paragraph. This value can be used as a measure of the neural algorithm efficiency considering the robustness of the results to the so called future (unknown) data - the data which are expected to be similar (but different) from the data used as the learning set.
Point out here some facts as follows:

- For data with and without the pile-up a different preprocessing is suitable - the KR method (concentric squares) for pile-up and the CL (clusters) for the data without pile-up.
- The minimal value of Y may arise for different ways of the data set splitting depending on the preprocessing method and the architecture of the neural net.
- The neural algorithm efficiency depends on the neural net architecture, i.e. the number of layers and the number of neurons on each layer, but is not too sensitive on it.

The neural algorithm using the EM calorimeter data only gives the value of Y 48.11 or 43.87 just as needed for LVL2 triggering. To introduce the safety margin the hadron veto and the information from other detectors can be used. (Compare with the factor K introduced in the third paragraph in Chapter 4.)

Table 3: Results on learning data – maximal enrichment factor

Explanation: Strings in the first column of the table are names of corresponding result files and denotes data and net architecture used. The names starting with 2- denotes that the data used was the data with pile-up, other are without pile-up. Next part of names denotes the kind of preprocessing ("KR","CL","CT","CIR" etc.). The last part of the name denotes the architecture of the neural network. Column pairs LT, LTTT and TTLTTT denote choice of learning and testing data.

data set splitting		LT		LTTT		TTLTTT		
data & nets	layers	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	$min\{\frac{e_{eff}}{j_{err}}\}$
2-KR3	3	0.002	86.6	0.004	150.6	0.006	64.5	64.5
2-KR4242	3	0.002	93.6	0.004	149.5	0.006	61.0	61.0
2-KR424	3	0.002	93.6	0.004	149.5	0.006	61.0	61.0
2-KR4	3	0.002	93.6	0.004	149.5	0.006	61.0	61.0
2-KR2	5	0.004	104.7	0.004	124.9	0.006	79.7	79.7
2-KR4242	4	0.002	100.6	0.004	139.0	0.006	80.9	80.9
2-KR2	7	0.004	100.1	0.004	133.1	0.006	61.0	61.0
2-KR2	6	0.004	100.6	0.004	133.1	0.006	63.3	63.3
2-KR242	4	0.006	94.8	0.004	164.7	0.006	65.7	65.7
2-KR1TO5	4	0.006	94.8	0.004	164.7	0.006	65.7	65.7
2-CL626	4	0.002	98.3	0.004	53.7	0.006	73.9	53.7
2-KR424	5	0.002	151.0	0.004	127.3	0.006	64.5	64.5
2-KR3	4	0.002	100.6	0.004	136.6	0.006	97.3	97.3
2-CT242	3	0.002	111.2	0.008	77.6	0.006	59.8	59.8
2-CT2	3	0.002	111.2	0.008	77.6	0.006	59.8	59.8
2-CT1TO5	3	0.002	111.2	0.008	77.6	0.006	59.8	59.8
COL242	5	0.004	67.0	0.008	63.0	0.012	59.7	59.7
COL1TO5	5	0.004	67.0	0.008	63.0	0.012	59.7	59.7
CIR7TO2	4	0.004	56.0	0.008	75.0	0.012	59.7	56.0
2-KR7TO2	3	0.002	93.6	0.004	172.8	0.006	55.1	55.1
2-CT2	5	0.004	55.6	0.004	89.9	0.006	89.1	55.6
2-KR424	4	0.002	87.8	0.004	110.9	0.006	72.7	72.7
2-CT2	4	0.002	120.5	0.008	91.7	0.006	90.3	90.3
2-KR626	3	0.006	96.0	0.004	193.9	0.006	97.3	96.0
2-CT3	3	0.002	96.0	0.004	105.1	0.006	79.7	79.7
2-CL8	3	0.002	85.4	0.004	73.5	0.006	55.1	55.1
2-KR8	3	0.002	96.0	0.004	142.5	0.006	109.1	96.0
2-CT242	4	0.002	81.9	0.008	83.5	0.006	92.6	81.9
2-CT1TO5	4	0.002	81.9	0.008	83.5	0.006	92.6	81.9
2-CL1TO5	6	0.002	108.8	0.004	120.3	0.006	66.8	66.8
COL242	4	0.004	79.0	0.008	64.0	0.012	60.7	60.7

continued on next page

continuation of table 3

		LT		LTTT		TTLTTT		
data & nets	layer	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	$min\{\frac{e_{eff}}{j_{err}}\}$
COL1TO5	4	0.004	79.0	0.008	64.0	0.012	60.7	60.7
2-KR242	5	0.002	264.5	0.008	92.8	0.006	130.2	92.8
2-KR1TO5	5	0.002	264.5	0.008	92.8	0.006	130.2	92.8
2-KR3	5	0.002	94.8	0.004	120.3	0.006	105.6	94.8
2-CT3	4	0.004	69.6	0.004	130.8	0.006	91.5	69.6
COL3	3	0.004	55.0	0.008	69.0	0.012	52.6	52.6
2-CT4242	3	0.002	97.1	0.004	93.4	0.006	86.8	86.8
2-CT424	3	0.002	97.1	0.004	93.4	0.006	86.8	86.8
2-CT4	3	0.002	97.1	0.004	93.4	0.006	86.8	86.8
2-CL4242	7	0.002	69.0	0.004	128.5	0.006	52.8	52.8

Table 4: Results on learning data – reduction ratio 0.02

Explanation: Strings in the first column of the table are names of corresponding result files and denotes data and net architecture used. The names starting with 2- denotes that the data used was the data with pile-up, other are without pile-up. Next part of names denotes the kind of preprocessing ("KR","CL","CT","CIR" etc.). The last part of the name denotes the architecture of the neural network. Column pairs LT, LTTT and TTLTTT denote choice of learning and testing data.

data set splitting		LT		LTTT		TTLTTT		
data & nets	layers	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	j_{err}	$\frac{e_{eff}}{j_{err}}$	$min\{\frac{e_{eff}}{j_{err}}\}$
2-KR2	5	0.020	37.5	0.020	38.3	0.020	44.9	37.5
2-CT2	5	0.020	28.6	0.020	43.2	0.020	55.1	28.6
2-KR2	7	0.020	35.7	0.020	42.7	0.020	44.2	35.7
COL242	3	0.020	29.7	0.020	32.0	0.020	45.5	29.7
COL2	3	0.020	29.7	0.020	32.0	0.020	45.5	29.7
COL1TO5	3	0.020	29.7	0.020	32.0	0.020	45.5	29.7
2-CT242	3	0.020	30.5	0.020	39.7	0.020	38.3	30.5
2-CT2	3	0.020	30.5	0.020	39.7	0.020	38.3	30.5
2-CT1TO5	3	0.020	30.5	0.020	39.7	0.020	38.3	30.5
2-CL2	4	0.020	28.5	0.020	28.2	0.020	31.0	28.2
2-KR424	5	0.020	39.1	0.020	46.7	0.020	37.1	37.1
2-KR4242	4	0.020	34.6	0.020	42.9	0.020	38.3	34.6
2-KR242	4	0.020	41.7	0.020	43.2	0.020	36.7	36.7
2-KR1TO5	4	0.020	41.7	0.020	43.2	0.020	36.7	36.7
2-CT4242	3	0.020	32.4	0.020	44.8	0.020	45.3	32.4
2-CT424	3	0.020	32.4	0.020	44.8	0.020	45.3	32.4
2-CT4	3	0.020	32.4	0.020	44.8	0.020	45.3	32.4

There is a question of the nature of the data which has been used for this analysis. Tables 3, 4 similar to Tables 1, 2 has been formed using the learning data. Again, some facts can be found as follows:

- Generally the values of Y are necessarily larger than in the Tables 1 and 2.
- The jets error (compression) is different from zero. It means that the neural net recognizes some jets from the learning set as electrons. It is not an error! This is the generalization ability of the neural net. A neural net is able to find even in the learning set the patterns of wrong class. Because em showers and jet showers have a statistical nature, in the learning set appear patterns from the opposite class, i.e. electrons very similar to jets and jets very similar to electrons. From this it follows that during learning the neural net adapts itself to optimal splitting of two classes of data even if these classes are partially overlapped or there are small errors in the learning set. Several wrong patterns in the learning set cannot cause any erroneous function of the neural net. Again, as in Tables 1, 2, the results are not too sensitive to the architecture of the neural net.

5 The Neural Net

An original paradigm of a neural net for electron/jet classification was proposed by Bitzan and Šmejkalová [BvK^{+}95b] and [BvK95a]. The neural net has a layered architecture (see figure 3) and all its layers have the same structure (with the exception of the input layer that is simplified). Every layer consists of a set of switches and modulators and one summator and comparator. The number of switches and modulators in every layer is equal to the number of input data, i.e. to the number of processed signals or their features. The modulators N perform a multiplication of input signals by weight coefficients. Switches S are used for choosing the proper set of values of weight coefficients according to the results from the previous layer. They ensure a step nonlinearity of the network by switching among several sets of weight coefficients. The net has been derived from rather common perceptron type neural net [Kra93].

The operation of the neural net is relatively simple. The input signals come from calorimeter sensors after preprocessing. Then they are processed in the modulators that perform their multiplication by weights. All modulated signals in one layer are summed up and compared with the threshold of that layer. According to the result of the comparison, the appropriate set of weights for the next layer of modulators is selected. After k steps (i.e. signal processing in k layers), the last sum compared to the threshold gives the final result that can identify the presence of either jet or electron.

In spite of that this neural net has been derived from layered perceptron-type neural net, it uses a different approach for its learning. The algorithm developed

[BvK+95b] is based on learning one layer after another one by sophisticated optimization algorithm [Luk94]. There is no feedback iterative procedure used.

From the principle of neural nets, it may be easily seen that what is considered as jet or as electron depends on the samples in the training set. The problem can be generalized and other events can be looked for. It would suppose a more precise analysis and relevant training sets of those events. The neural algorithm offers better facilities of the classification, but it has no direct physical interpretation similar to cluster algorithm where the thresholds are easily understandable as levels of energy in the regions specified.

Originally, an electrooptical implementation of the neural net has been proposed [BvK+95b]. Electrooptical implementation of a neural net can provide the speed required. The optical or electrooptical elements have some favourable features: fast operation, summation of optical signals is trivial, multiplication in electrooptic modulators is relatively simple and fast, optical signals can be easily compared. On the other hand, electrooptic devices have some disadvantages: optical signals cannot be substracted, and there is an essential signal degradation in electrooptical modulators.

Current designs for the LVL2 trigger system propose the use of commercial circuits and processors and do not rely on the development of specialized devices to achieve the performance required [atl94]. The same is true for the neural net.

5.1 Switching neural net architecture:

Switching neural net consists from three types of units which form neural net consisting from layers. Each layer contain one switching unit and at least two (except the input layer, which is formed by one neural unit only) neural units per one input. The third type of unit perform output mapping from last layer only. The formal description of this three types of units is the following:

Neural unit

$$(1) \qquad N : R^d \times W \Rightarrow R^d, \quad \mathbf{N} = \mathbf{N}(\mathbf{x}, \mathbf{w}),\ \mathbf{x} \in R^d, \quad \mathbf{w} \in W,$$

where $\mathbf{x}$ is an input vector and $\mathbf{w}$ is a parameter vector from a parameter space W.

Switching unit

$$(2) \qquad S : R^d \times U \Rightarrow \{1, 2, \ldots, m\},\ S = S(\mathbf{x}, \mathbf{u}),\ \ \mathbf{u} \in U,$$

where $\mathbf{u}$ is a parameter vector from parameter space U, and $m > 1$ is an integer equal to the number of units in the next layer.

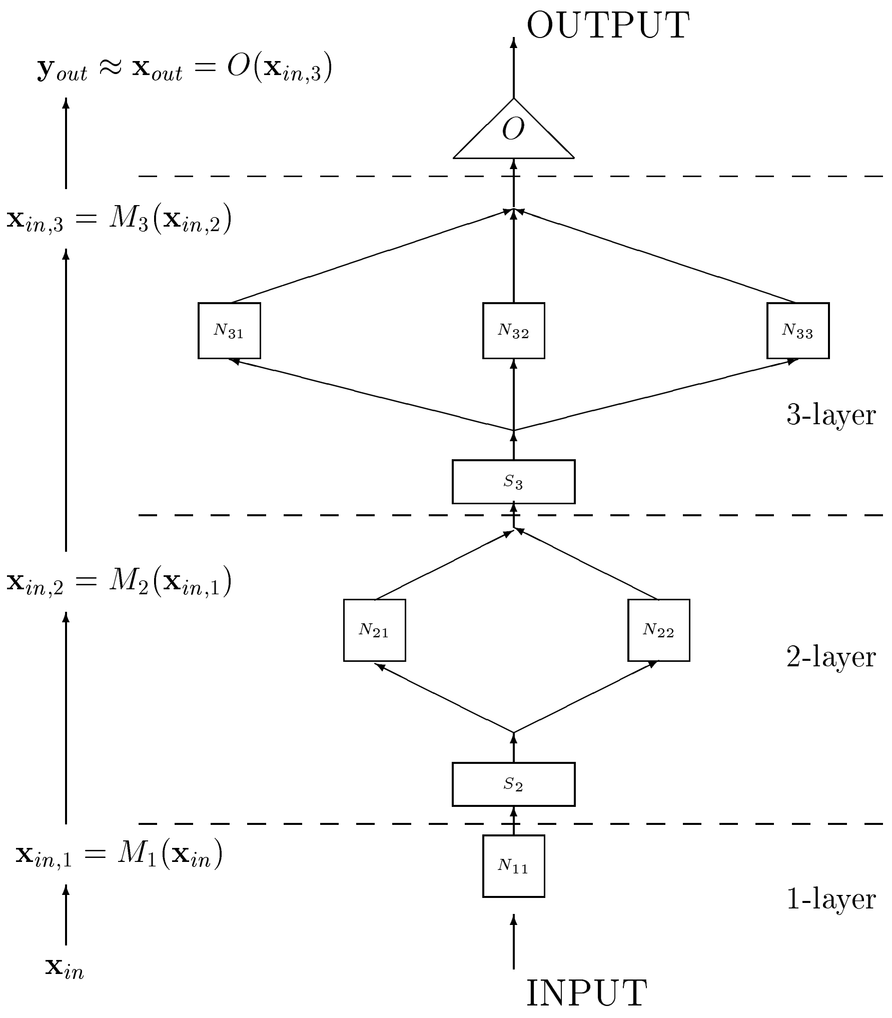

OUTPUT
$\mathbf{y}_{out} \approx \mathbf{x}_{out} = O(\mathbf{x}_{in,3})$
O
$\mathbf{x}_{in,3} = M_3(\mathbf{x}_{in,2})$
N_{31}
N_{32}
N_{33}
3-layer
S_3
$\mathbf{x}_{in,2} = M_2(\mathbf{x}_{in,1})$
N_{21}
N_{22}
2-layer
S_2
$\mathbf{x}_{in,1} = M_1(\mathbf{x}_{in})$
N_{11}
1-layer
$\mathbf{x}_{in}$
INPUT

Figure 3: Scheme of a 3-layer network with switching units. An input sample $\mathbf{x} \in R^d$ is mapped into $\mathbf{x}_{out} = O(x_{in,3})$ approximating $\mathbf{y}_{out} \in R^b$ (each node N_{ij} in this scheme performs multiplication of input signals (of dimension d) by diagonal $d \times d$ matrix of weights).

Output unit is represented by an output function O, which projects R^d into lower or equal dimension

$$O : R^d \Rightarrow R^b, \quad O = O(\mathbf{x}), \quad b \leq d. \tag{3}$$

Architecture of switching neural net and connections between units is described in the 3.

5.2 Description of the net response:

Switching net maps input $\mathbf{x}_{in} \in R^d$ into output

$$\mathbf{x}_{out} = M(\mathbf{x}_{in}) \in R^b,$$

where the mapping M is a composition of mappings performed by each layer

$$M = S \odot M_{nl} \odot M_{nl-1} \odot \cdots \odot M_2 \odot M_1,$$

n_l is number of layers, $M_1 = N_{11}$, and

$$\mathbf{x}_{in,k} = M_k \odot \cdots \odot M_1(\mathbf{x}_{in})$$

$$M_k = N_{kS_k(\mathbf{x}_{in,k-1})}, \quad \text{for} \quad k > 1.$$

5.2.1 Case of electro-optic implementation:

To reach maximum performance speed we decide to use an opto-electronic implementation of switching neural net. Unfortunately, this leads us to simplify the mappings performed by each neural and switching unit. Let m is the number of neurons in a layer, and

$$N_{ij} = \mathbf{w} \cdot \mathbf{x} + \mathbf{s}, \mathbf{w} \in R^{d,d}, \mathbf{s} \in R^d,$$

$$S_k = \begin{array}{llcll} 1 & \text{iff} & \mathbf{u} \cdot \mathbf{x} & < & t_1 \\ 2 & & \mathbf{u} \cdot \mathbf{x} & \in & \langle t_1, t_2) \quad t_2 > t_1 \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ m & & \mathbf{u} \cdot \mathbf{x} & \geq & t_{m-1} \quad , \end{array}$$

$$O(\mathbf{x}) = \mathbf{b} \cdot \mathbf{x}, \ \mathbf{b} \in R^d.$$

This type of net is implementable by opto-electronic hardware.

5.2.2 Learning process of the net:

Most of neural nets work in two phases. First is the learning procedure of the net, during which the parameters of the nets are adjusted (such as weights of connections, thresholds for units, parameters of threshold functions, etc.). The second phase use this parameters to perform a mapping, which could approximate mapping, used in the learning phase. Let us now to turn attention to the learning process. The learning process is performed on each layer, independently on parameters in other layers. The results on the last learned layer are grouped in n_l clusters (here n_l is the number of neurons in the layer). This clustering based on Jancey's cluster algorithm, which is briefly explained in the following way (see also page 31):

let $O(\mathbf{x}) = \mathbf{b} \cdot \mathbf{x}$, $p_i = O(M_k(\mathbf{x}^i_{in}))$, n_p = number of patterns and $q = 1, \cdots, m$.

```
1. for randomly chosen sequence $1 \leq j_1 < j_2 < \cdots < j_m \leq n_p$
   set
   $\mathbf{c}_q^{new} = \mathbf{c}_q^{old} = \mathbf{p}_{j_q}$
   and
   $S_q^{new} = S_q^{old} = \{\mathbf{p}_{j_q}\}$,

2. let $r_1, \cdots, r_{n_p}$ is random permutation of the $1, \cdots, n_p$,

3. for all $k = r_1, \cdots, r_{n_p}$
     DO
          $i = \min\left\{v \middle| \|\mathbf{c}_v^{old} - \mathbf{p}_k\| = \min\{\|\mathbf{c}_q^{old} - \mathbf{p}_k\|\}\right\}$,
          $c_q^{old} = \mathbf{c}_q^{old} - \frac{\mathbf{p}_k - \mathbf{c}_q^{old}}{|S_q^{old}|}$, $c_i^{old} = \mathbf{c}_i^{old} + \frac{\mathbf{p}_k - \mathbf{c}_i^{old}}{|S_i^{old}|}$
          $S_q^{old} = S_q^{old} \setminus \{\mathbf{p}_k\}$, $S_i^{old} = S_i^{old} \cup \{\mathbf{p}_k\}$,
     END

4. IF $(\exists q)(S_q^{new} \neq S_q^{old})$
   THEN for all such $q$ let $\left\{\mathbf{c}_q^{new} = \mathbf{c}_q^{old}, S_q^{new} = S_q^{old}\right\}$ and GOTO 2

5. STOP
```

After clustering each cluster is jointed with a neuron in the layer and consequently parameters of this neuron are adjusted with regard patterns in the corresponding cluster only. Parameters are adjusted in such a way that the mean square error (cost function) is minimized over the set of neuron parameters (vector of connections weights and vector of threshold values). This is done by UFO library (see [Luk94]).

6 Software simulation

Above neural net classifier was implemented under MS DOS operating system in C program language (except UFO interface subrutine, which is generated by UFO preprocessor in the FORTRAN 77). This modules are compiled and linked together by MS C++ ver. 7 compiler and linker. The following parts document the possibilities of this neural net implementation.

6.1 Command line options

There are three possibilities to control program C7NNN.EXE :

1. directly from command line,
2. interactively from C7NNN.EXE prompt,
3. the commands can be read from command file.

Note:

- commands can not contain blank character

6.1.1 General commands:

?	short list of options available
v	view current settings
@*file.ext*	options will be read from specified file *file.ext*
! *command*	DOS command ('_' will be placed by ' '), use ! for shell
p*string*	defines net prompt (all '_' will be placed by ' ')
;*string*	comment: this option will be skipped
h	show helpfile <u>NNN.HLP</u>
q	quit program

6.1.2 Net commands:

+nl	load net from file defined by **-nn** option into memory
+ns	save net from memory into file defined by **-nn**
-nn*netfile*	put net file name
+nd	delete net from memory
+nr	run learning process (net and data must be loaded before this command)

6.1.3 Testing and learning data commands:

Note: t/l means l or t exclusively.

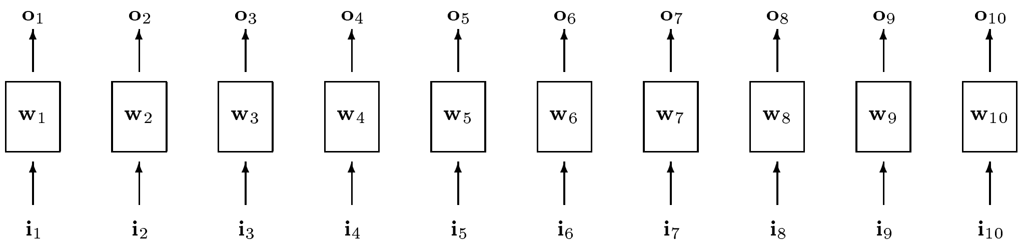
-t/ln*datafile*	put testing/learning data file name
+t/ll*int*	load testing/learning data, (*int* is the number of data to load, if not, all data will be loaded
+t/lt	test testing/learning data
-t/le*errfile*	put name of error file for testing/learning data
-t/ld*string*	this *string* defines which data from data file will be used for learning and testing (is not case sensitive). The meaning of characters in this string is: **L** only for learning; **T** only for testing; **B** both learning and testing; **N** none of above (skipped for learning and testing) for example, **-tdllbTn** takes patterns number 1,2,5,6,9,10... for learning net and patterns number 2,3,6,7,10,11... for testing results.

6.1.4 Learning parameters commands:

-rf *int*	defines starting layer for learning, if is greater than number of layers already learned, learning process starts from the first unlearned layer
-rt *int*	number of the last layer used in the learning process
-rs *float*	desired value of the error function. If it is reached learning process stops.
-rc *[+/-]*	defines border for switching units. For '+' border is between clusters, for '–' border is in the center of the cluster.
-rb *float*	the number *float* defines the border for cluster separation during learning process.

6.1.5 Hardware simulation commands:

Behaviour of a hardware realization of opto-electronic neural net is described using the following schema and equations:


$\mathbf{o}_1$ $\mathbf{o}_2$ $\mathbf{o}_3$ $\mathbf{o}_4$ $\mathbf{o}_5$ $\mathbf{o}_6$ $\mathbf{o}_7$ $\mathbf{o}_8$ $\mathbf{o}_9$ $\mathbf{o}_{10}$
$\mathbf{w}_1$ $\mathbf{w}_2$ $\mathbf{w}_3$ $\mathbf{w}_4$ $\mathbf{w}_5$ $\mathbf{w}_6$ $\mathbf{w}_7$ $\mathbf{w}_8$ $\mathbf{w}_9$ $\mathbf{w}_{10}$
$\mathbf{i}_1$ $\mathbf{i}_2$ $\mathbf{i}_3$ $\mathbf{i}_4$ $\mathbf{i}_5$ $\mathbf{i}_6$ $\mathbf{i}_7$ $\mathbf{i}_8$ $\mathbf{i}_9$ $\mathbf{i}_{10}$

where

$$\mathbf{o}_j = \Big(\mathbf{w}_j + \beta_W(\mathbf{w}_{j-1} - 2\mathbf{w}_j + \mathbf{w}_{j+1}) + \Phi_W\Big) \times \Big(\mathbf{i}_j + \beta_I(\mathbf{i}_{j-1} - 2\mathbf{i}_j + \mathbf{i}_{j+1}) + \Phi_I\Big) + \Phi_{ABS},$$

and

$\Phi_W = \alpha_W e^{-\frac{x^2}{\Gamma_W}}$	weights error,
$\Phi_I = \alpha_I e^{-\frac{x^2}{\Gamma_I}}$	input error,
$\Phi_{ABS} = \alpha_{ABS} e^{-\frac{x^2}{\Gamma_{ABS}}}$	absolute error,
β_W	lateral weight influence factor,
β_I	lateral input influence factor.

This values are entered using the following commands:

-hr *int*	run *int* $\times$ the random generator rand()
-h *[+/-]*	'+' use hardware simulation, '–' exact computing
-hw *[+/-]*	'+' use Φ_W correction, '–' do not use Φ_W correction
-hwaα_W	and
-hwgΓ_W	for $\Phi_W = \alpha_W e^{-\frac{x^2}{\Gamma_W}}$
-hi *[+/-]*	'+' use Φ_I correction, '–' do not use Φ_I correction
-hiaα_I	and
-higΓ_I	for $\Phi_I = \alpha_I e^{-\frac{x^2}{\Gamma_I}}$
-ha *[+/-]*	'+' use Φ_{ABS} correction, '–' do not use Φ_{ABS} correction
-haaα_{ABS}	and
-hagΓ_{ABS}	for $\Phi_{ABS} = \alpha_{ABS} e^{-\frac{x^2}{\Gamma_{ABS}}}$
-hbw *[+/-]*	'+' use W-lateral correction, '–' do not use W-lateral correction
-hbwbβ_W	for W-lateral influence factors
-hbi *[+/-]*	'+' use I-lateral correction, '–' do not use I-lateral correction
-hbibβ_I	for I-lateral influence factors

Following group of hardware options control the number of bits which can hardware implementation transmit. Usually this precision is 8 or 9 bits.

-hpw *[+/-]* '–' do not use restricted precision, '+' use precision of **-hpwp***int* levels for weights, which means that all weights will be truncated to the range $[\Omega_{min}, \Omega_{max}]$ and then rounded to desired precision. E.g. to $2^k + 1$ equidistant values starting from Ω_{min} and ending Ω_{max}.

-hpwp *int* defines the number of precision levels (e.g. *int* $= 2^k$ for k-bit precision of the hardware transition of weights)

-hpwn *int* defines the number of smallest and greatest weights which are neglected during learning or testing of data sets, for **-hpwn0** all possible values will not be changed, for **-hpwn**α α minimal and maximal values arising during net performance will be truncated to the α-th minimal $(= \Omega^w_{min})$ or maximal $(= \Omega^w_{max})$ value

-hpi *[+/-]* '–' do not use restricted precision, '+' use precision of **-hpwp***int* levels for inputs, which means that all inputs will be truncated to the range $[\Omega_{min}, \Omega_{max}]$ and then rounded to desired precision. E.g. to $2^k + 1$ equidistant values starting from Ω_{min} and ending Ω_{max}.

-hpip *int* defines the number of precision levels (e.g. *int* $= 2^k$ for k-bit precision of the hardware transition of inputs)

-hpin *int* defines the number of smallest and greatest inputs into layers which are neglected during learning or testing of data sets, for **-hpin0** all possible values will not be changed, for **-hpin**α α minimal and maximal values arising during net performance will be truncated to the α-th minimal $(= \Omega^i_{min})$ or maximal $(= \Omega^i_{max})$ value

The last group of hardware options controls simulation of errors arising during analog multiplication.

-he *[+/-]* '–' exact hardware multiplication, '+' use hardware simulation of multiplied errors, which means, that the values of $\mathbf{w}_j$, $\mathbf{o}_j$ and $\mathbf{i}_j$ will be truncated to the range $[\Omega_{min}, \Omega_{max}]$. Let us denote this values as $\mathbf{w}_j^{tr}$, $\mathbf{o}_j^{tr}$ and $\mathbf{i}_j^{tr}$. Then $\mathbf{o}_j$ will be transformed to

$$\mathbf{o}_j' = \mathbf{o}_j^{tr} \cdot \left[1 + f\left(i_1 + \frac{(\Omega^i_{max} - \mathbf{i}_j^{tr})(i_n - i_1)}{\Omega^i_{max} - \Omega^i_{min}}, w_1 + \frac{(\Omega^w_{max} - \mathbf{w}_j^{tr})(w_n - w_1)}{\Omega^w_{max} - \Omega^w_{min}}\right)\right].$$

The value $\mathbf{o}_j'$ will be again truncated.

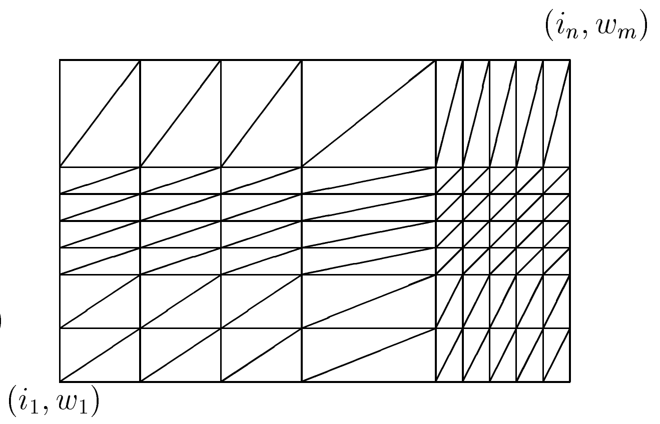
-hef *errfile* define the file which function of errors obtained during hardware multiplication. This file contains values of error function in specified points. Rectangular rule will be used for interpolation between adjacent points.

6.1.6 Hardware nonlinearity file:

This file which is specified by **-hef***file.ext* contains set of numbers which defines the errors obtained during hardware multiplication. This errors are described by twodimensional error function.

It is supposed that two first numbers are numbers of subintervals (in the i- and w-direction, respectively) and the following numbers are values of the arguments. Finally, the function values follows. Exactly, if the contents of this file is

n	m		
i_1	i_2	$\cdots$	i_n
w_1	w_2	$\cdots$	w_m
$f(i_1, w_1)$	$f(i_2, w_1)$	$\cdots$	$f(i_n, w_1)$
$f(i_1, w_2)$	$f(i_2, w_2)$	$\cdots$	$f(i_n, w_2)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$f(i_1, w_m)$	$f(i_2, w_m)$	$\cdots$	$f(i_n, w_m)$



the error function f is defined on splines defined by values of i_i and w_j as corresponding linear interpolation of values in spline vertices. Outside of the rectangle $[i_1, i_n] \times [w_1, w_m]$ is defined as zero.

6.1.7 Example:

6.2 Data structure:

Neural net parameters, data and results are stored in ASCII files, with the following structures.

6.2.1 Neural net parameters file:

FILE CONTENTS	DESCRIPTION
text ended **$**	comment only
$n_i, n_o, n_p, n_{lps}, n_{ll}, n_{tf}$	n_i number of inputs n_o number of output (must be 1) n_p number of parameters of the threshold function (for $n_{tf} = 1$ must be 2) n_{lps} number of layers with predefined size n_{ll} number of learned layer n_{tf} type of threshold function (must be 5)
$ls_1, \cdots, ls_{n_{lps}}$	size of layers, layers above has implicitly two neurons
$s_{1,0,1} \cdots s_{1,0,n_i}$	selector in the layer 0
$w_{1,1,1}, \cdots, w_{1,1,n_i}$	weights of the 1 neuron in the 0 layer
$s_{1,1,1} \cdots s_{1,1,n_i}$ $\cdots$	selector n. 1 in the layer 1
$s_{ls_1-1,1,1} \cdots s_{ls_1-1,1,n_i}$	selector n. $ls_1 - 1$ in the layer 1
$w_{1,1,1}, \cdots, w_{1,1,n_i}$ $\cdots$	weights of the 1 neuron in the 1 layer
$w_{ls_1,1,1}, \cdots, w_{ls_1,1,n_i}$	weights of the ls_1-th neuron in the 1 layer
$\cdots$	other layers
$v_1, \cdots v_{n_i}$	variet

6.2.2 Data files:

FILE CONTENTS	**DESCRIPTION**
text ended $	comment only
n_p	number of patterns
n_i n_o	number of inputs and outputs (=1)
$i_{1,1} \cdots, i_{1,n_i}$	first pattern
$o_{1,1} \cdots, o_{1,n_o}$	
$\cdots$	
$i_{n_p,1} \cdots, i_{n_p,n_i}$	last pattern
$o_{n_p,1} \cdots, o_{n_p,n_o}$	

6.2.3 Error output file:

This file contains results obtained by net for inputs:

FILE CONTENTS	**DESCRIPTION**
R	
J/E$_{float}$ $\cdots$ J/E$_{float}$	results after the first layer
G	
$g_{1,1}, \cdots, g_{1,ls_1-1}$	thresholds of the first selector
$\cdots$	
$\cdots$	
R	
J/E$_{float}$ $\cdots$ J/E$_{float}$	results after last learned layer
G	
$g_{n_{lll},1}, \cdots, g_{n_{lll},ls_{n_{lll}}-1}$	thresholds of the first selector
S	
some summary information	

6.2.4 Learning info file:

The learning info file <u>LEARNING.INF</u> is created during learning of the net. It contains references to files with net parameter and with learning and testing data.

Also results of the learning process and comments to cluster analysis are stored in this file.

6.3 Some notes:

- This program creates some temporally files on disk when it runs. Therefore it is recommended to run this program from RAM disk because speed of the learning part remarkable increased in this case.
- memory requirements: this program was tested under 560 Kb of DOS memory for data sets about 2000 patterns (from R^{25} space).
- on PC486 66MHz machine the learning procedure of one layer for data sets sets of 2000 patterns (from R^{25} space) takes about a half hour in average.

7 Utilities

7.1 Purpose of the utility program:

Utility program NNN_UTIL.EXE is a program which allow evaluate the performance of the network. It visualize results of learning process and answers to testing data. It also provide the opportunity to check influence of accidental influence of signals in the hardware.

This utility program is controlled by the same manner as the program C7NNN.EXE for simulation of the neural net.

7.2 Command line options:

7.2.1 General commands:

?	short list of options available
v	view current settings
@*file.ext*	options will be read from specified file
! *command*	DOS *command* ('_' will be placed by ' '), use **!** for shell
p *string*	defines net prompt (all '_' will be placed by ' ')
; *string*	comment: this option will be skipped
h	call help
q	quit program

7.2.2 Run commands:

+s show results histogram after layers. Density of histogram, such as smoothing factor of histogram can be defined. This plots allow appreciate quality of clustering (see page 33).

+r evaluate rate ρ and compress factor γ, defined as (see page 32)

$$\rho = \frac{e_E}{e_J} \quad \text{and} \quad \gamma = \frac{JasE + JasJ}{JasE},$$

where

$$e_J = \frac{JasE}{JasE + JasJ} \quad , \quad e_E = \frac{EasE}{EasE + EasJ}$$

and

EasE=number of well classified electrons
EasJ =number of misclassified electrons
JasJ =number of well classified jets
JasE=number of misclassified jets

store results for specified file into files defined by **-o***file.ext* . This operation is performed over all files specified in **-u***file.ext* .

+c convert result file generated by program C7NNN.EXE to file readable by utility program. It must be performed before commands **+r** , **+l** , **+s** , **+d** .

+l show "lattice", thats means two dimensional plot showing "path" of patterns in the net(see page 34).

+d make four plots with good classified and misclassified electrons and jets. Perhaps, it could be helpful to see if undistinguishable patterns are contained in learning and testing sets.

7.2.3 File specification:

Wildchars ? and * are allowed ion the names of files.

-n	define set of result files generating by C7NNN.EXE which will be transformed into files with the same name and extension ended by $, which are readable by utility program.
-u	define files (which have been obtained by **+c** previously) for **+s** , **+l** , **+d** , **+r** commands
-e	define file with learning and testing data (is necessary for **+d** only)
-o	define output file for **+r**

7.2.4 Graph commands:

-g *int*	if *int* nonzero then print graph on the screen
-b *int*	if *int* nonzero black and white graphs will be printed (useful if sending screen contents to printer)
-x *int*	if *int* is nonzero fixed X-range of graphs defined by **-m** *int* and **-i** *int* is used
-y *int*	if *int* nonzero same Y-scale will be used in graphs

7.2.5 Other:

-a *float*	defines maximal value for X axis
-m *float*	defines minimal value for X axis
-t *float*	limiting value for electrons discrimination for **+d** . Electrons above this value will be supposed misdefined.
-j *float*	limiting value for jets discrimination for **+d** . Jets under this value will be supposed misdefined.
-l *int*	if *int* is nonzero graph of the last layer will be printed only
-h *int*	defines histogram size (used in **+s**)
-k *int*	defines is smooth factor for histogram (used in **+s**)

8 Conclusion

Specially designed neural nets can solve recognition tasks without knowing exact rules for recognition. The ability of neural nets to learn and generalize enables to define the task needed by suitable set of examples. The generalization ability of neural net does not mean only the ability correctly process data “never seen before”. The essential aspect of generalization is the ability to learn properly even if the learning set contains small amount of wrong examples. After such learning the neural net recognizes the wrong examples as the other class than the proper examples of the learning set. From this folows high but not 100% recognition of examples from the learning set.

In this study we have shown application and implementaion of neural net for L2 triggering. It was supposed direct generation of L2Accept/Reject signal rather than generation of physical features.

References

[atl94] Atlas technical proposal for a general-purpose pp experiment at the large hadron collider at cern, Dec 1994. CERN/LHCC/9445 LHCC/P2.

[BvK95a] Pavel Bitzan, Jana Šmejkalová, and Miroslav Kučera. Neural networks with switching units. *Neural Network World*, 4:515–526, 1995.

[BvK$^+$95b] Pavel Bitzan, Jana Šmejkalová, Miroslav Kučera, Miroslav Pařízek, and Miloš Matyáš. Theory and technical implementation of a neural network with switching units. Technical report, CERN Geneva, Feb 1995. EAST Note 94-05.

[Hak93] F. Hakl. Basic theory of neural networks derived from the B-S-B model. *Neural Network World*, 3:319–351, 1993.

[Kra93] M. A. Kraaijveld. Small sample behavior of multi-layer feedforward network classifiers theoretical and practical aspects. Delft University Press, 1993.

[LG19] M. Los and N. De Groot. B-tagging in delphi with a feed-forward neural network. In O. Behar et al. (eds.):, editor, *Neural Networks: From Biology to High Energy Physics. EDS*, pages 459–466. Editrice Pisa Italy, 19??

[Luk94] Ladislav Lukšan et al. Interactive system for universal functional optimization (ufo). Technical Report 599, Institute of Computer Science, Academy of Sciences of the Czech Republic, Jan 1994.

A Jancey's flow diagram

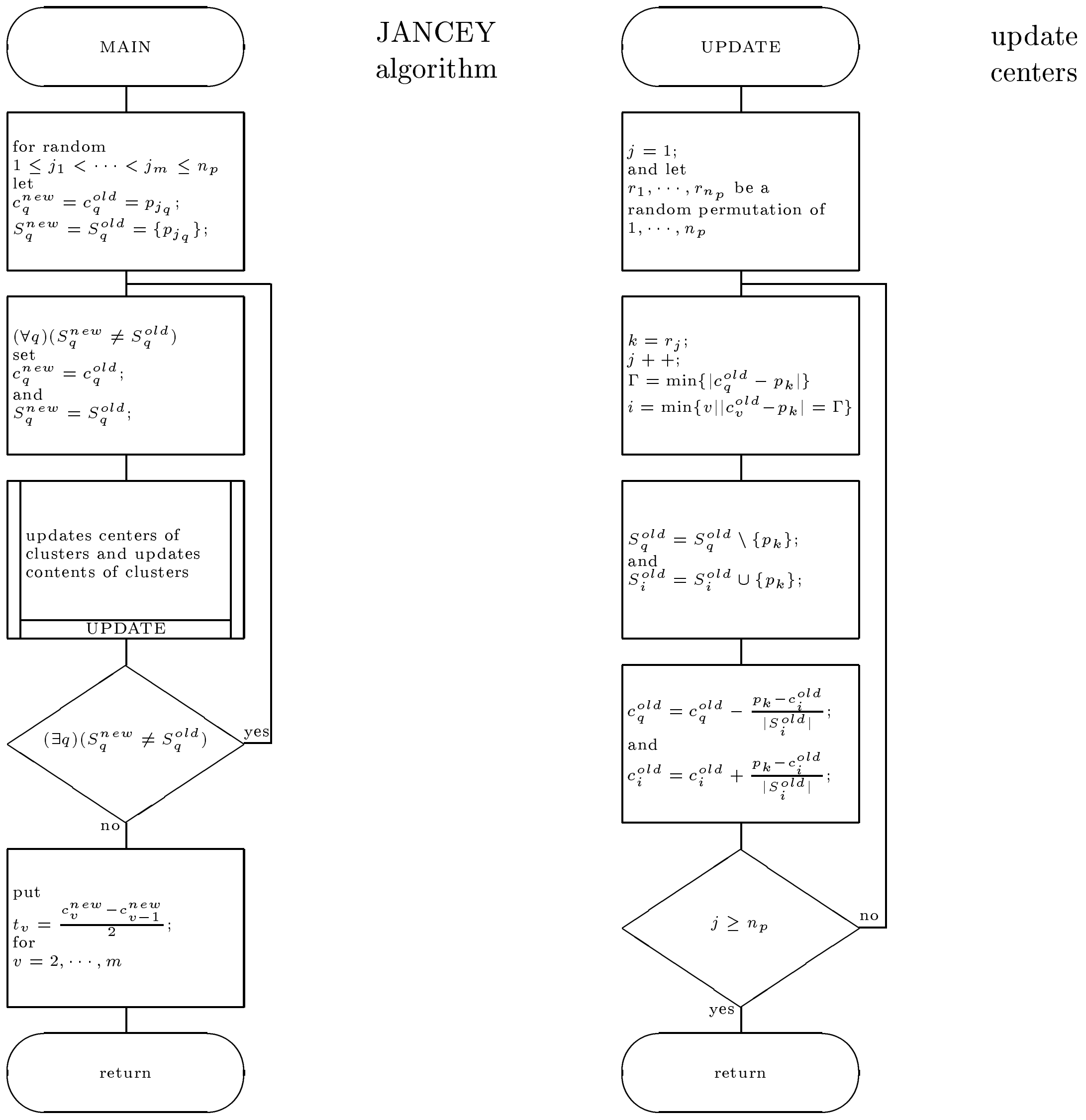

MAIN
JANCEY algorithm
for random $1 \le j_1 < \cdots < j_m \le n_p$ let $c_q^{new} = c_q^{old} = p_{j_q}$; $S_q^{new} = S_q^{old} = \{p_{j_q}\}$;
$(\forall q)(S_q^{new} \ne S_q^{old})$ set $c_q^{new} = c_q^{old}$; and $S_q^{new} = S_q^{old}$;
updates centers of clusters and updates contents of clusters
UPDATE
$(\exists q)(S_q^{new} \ne S_q^{old})$
yes
no
put $t_v = \frac{c_v^{new} - c_{v-1}^{new}}{2}$; for $v = 2, \cdots, m$
return
UPDATE
update centers
$j = 1$; and let $r_1, \cdots, r_{n_p}$ be a random permutation of $1, \cdots, n_p$
$k = r_j$; $j++$; $\Gamma = \min\{|c_q^{old} - p_k|\}$ $i = \min\{v \mid |c_v^{old} - p_k| = \Gamma\}$
$S_q^{old} = S_q^{old} \setminus \{p_k\}$; and $S_i^{old} = S_i^{old} \cup \{p_k\}$;
$c_q^{old} = c_q^{old} - \frac{p_k - c_i^{old}}{|S_i^{old}|}$; and $c_i^{old} = c_i^{old} + \frac{p_k - c_i^{old}}{|S_i^{old}|}$;
$j \ge n_p$
no
yes
return

Figure 4: Flow diagram of the Jancey clustering method

B Examples of output of the utility program:

B.1 Rates results

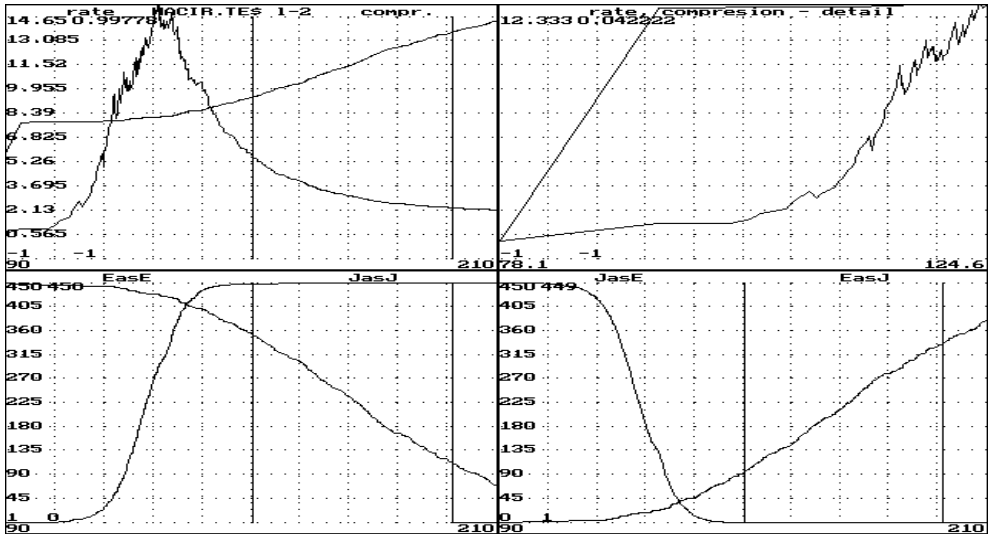


Figure 5: Rates and concentration of electrons after 1-st layer. (↖ - rates of electrons after and before 1-st layer and compression of the data flow, ↗ - detail of the preceding (steps are caused by small number of events in border parts of diagram), ↙ - well classified electrons (left) and jets (right), ↘ - misclassified electrons (left) and jets (right)

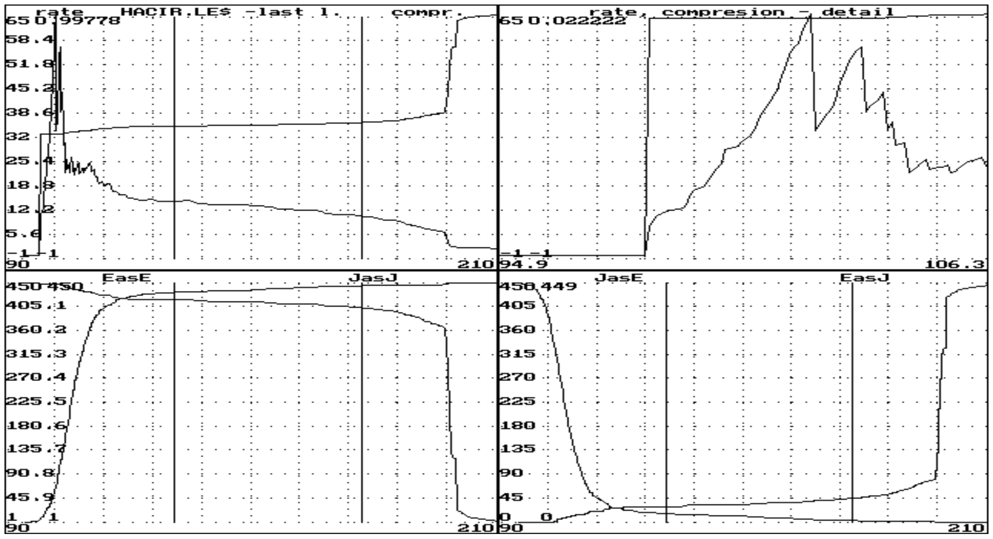


Figure 6: Rates and compression of electrons after triggering.

B.2 Some clusters

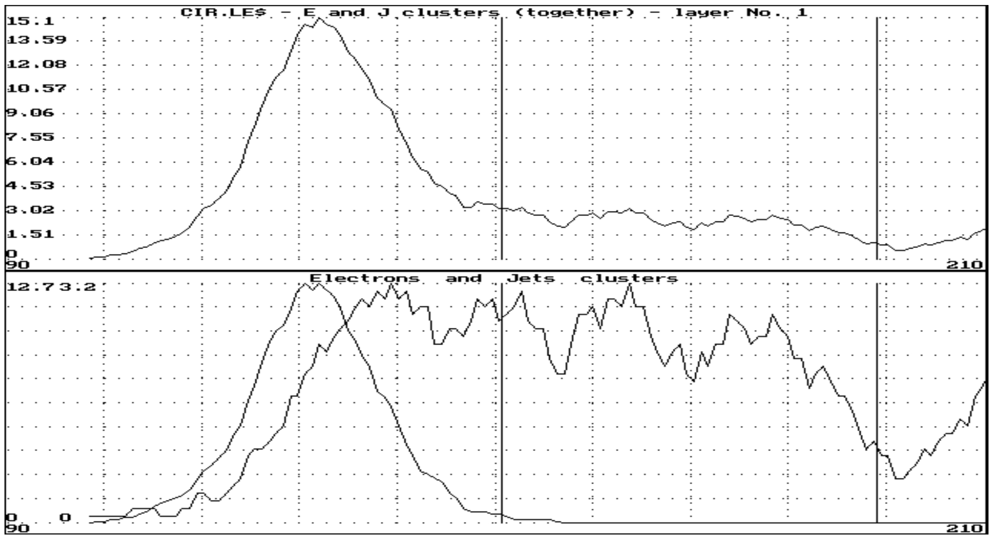


Figure 7: Clusters of resulting values after 1-st layer. Electrons are in the left part of the graph. Vertical bold lines denote the border of the clusters

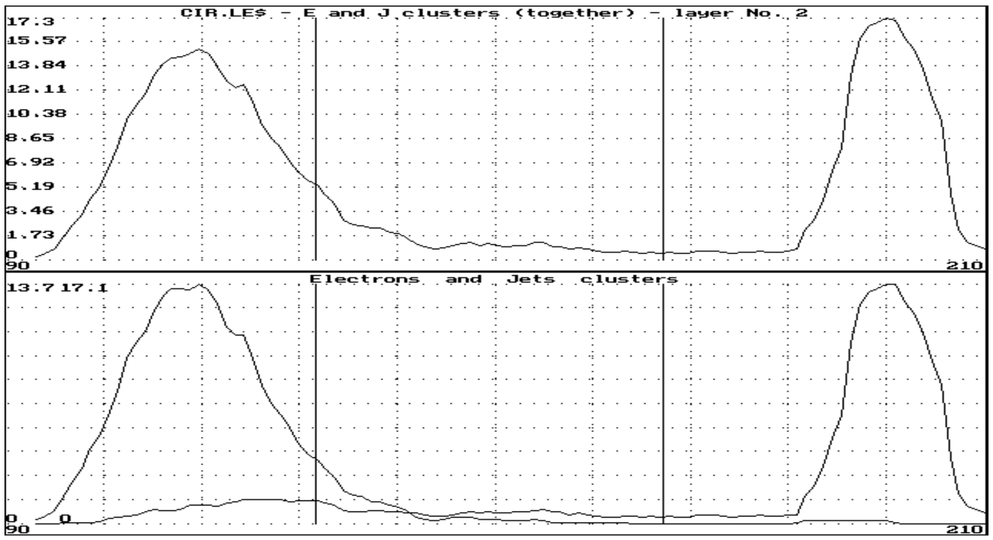


Figure 8: Clusters of resulting values after last layer.

B.3 Layers results

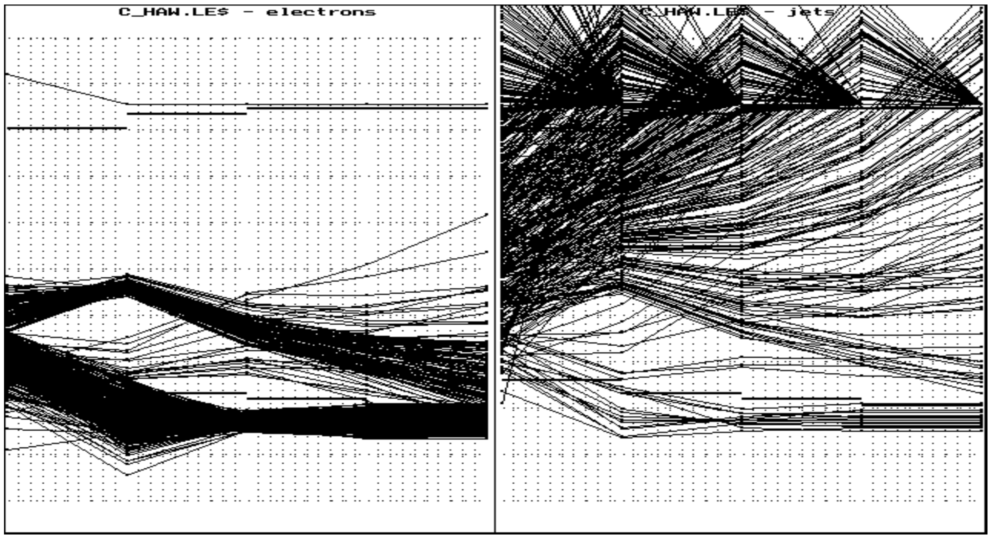


Figure 9: The process of electron/jets separation. Horizontal bold lines denote borders of clusters in corresponding layer.

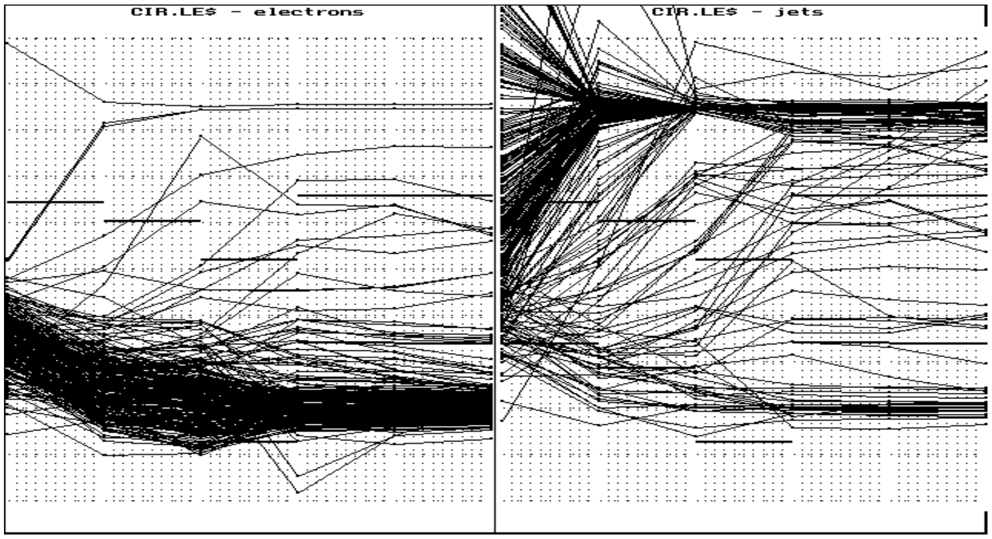


Figure 10: The process of electron/jets separation.