CMS TN/96-??
??, 1996

Second level e/γ calorimeter trigger in CMS Simulation study with CMSIM data

R. Nóbrega[1)], J. Varela[2,a)]

ABSTRACT

We present in this note preliminary results on the second level e/γ trigger studies. Using the full granularity in the ECAL, CMSIM simulated data, and a dedicated algorithm, the trigger rate can be reduced by factors ~7 (with marginal reduction of the e/γ efficiency, 2%), with respect to the rates obtained after the first level trigger.

[1)] LIP, Av. Elias Garcia 14, P-1000 Lisbon, Portugal
[2)] CERN, CH-1211 Geneva 23, Switzerland
[a)] On leave from LIP, Lisbon, Portugal

1 Introduction

The Level-1 calorimeter trigger proposed for CMS [1, 2] does not exhaust the information the ECAL provides, and in the CMS Technical Proposal, a possible Level-2 calorimeter trigger was described. Being implemented on a CPU farm, and having access to the full resolution and granularity of the ECAL, it is expected that the Level-2 trigger reduces the trigger rate by one order of magnitude.

We investigate in this note the performance of an algorithm that concentrates on a small region around the shower, being thus appropriate to high, as well as low luminosity operation.

This note is organized as follows: Section 2 describes the simulation conditions; section 3 contains a description of the trigger algorithm, including a discussion on modular neural networks, on which the algorithm is based; the results on the trigger rates are presented in section 4.

2 Simulation conditions and data

The data for these simulations consist of QCD ($\sim$ 50k events) generated with PYTHIA 5.7 [3], and electrons and photons generated at 10, 20 and 40 GeV transverse energy (5k in each sample). The QCD events were divided into 10 bins of p_T, from 5 GeV to 225 GeV.

The high luminosity conditions were simulated by superimposing to each event 20 (poisson distribution average) minimum bias events (out of a file with $\sim$ 10k events, also simulated with PYTHIA 5.7).

The simulation of the CMS detector was performed using the CMSIM package, version 8, with the following options:

- ICVERS=10 — CMS detector version 10: TP design;
- ITVERS=2 — tracker version 2 (average description);
- IEVERS=–120 — $PbWO_4$ crystal ECAL version;
- IEHITS=–1 — semi-fast version of ECAL;
- IHVERS=–42 — semi-fast version of HCAL;
- preshower included in the endcaps only;
- endcap ECAL design as in Technical Proposal.

Before being considered for the Level-2 trigger studies, the patterns were processed by the e/γ Level-1 trigger algorithms [4]. A sample of $\sim$ 13k electrons/photons and $\sim$ 1500 QCD events was selected.

3 The Level-2 electron/photon trigger algorithm

The key features the algorithm uses are summarized in the following:

Sophisticated computations: being implemented on a CPU farm, the Level-2 algorithms can be somewhat more elaborated than the first level ones;

High resolution: the full resolution of the ECAL is used. Unlike in the Level-1 trigger, where the individual crystals were grouped into strips of $1(\eta)\times 6(\varphi)$ crystals, the Level-2 trigger has access to the contents of each crystal;

Flexibility on the region of application: the concept of fixed-geometry trigger cell (Level-1) is no longer used. A more precise determination of the shower (bary)centre can be performed;

In short, the idea behind the algorithm is that of a pattern recognition. After some pre-processing, the ECAL map in the region indicated by the Level-1 trigger is "compared" with a standard map, kept in memory. The result of this comparison, performed by a neural network (NN), yields the Level-2 trigger decision.

We begin by discussing the choice of an architecture for the neural trigger; then, the processing of the patterns is presented. Finally, we describe the training procedure used to determine the NN weights.

3.1 Choosing an architecture

The use of neural networks as classifiers, particularly in physics analysis [5], has been considered for quite some time. Usually, multi-layer perceptrons (MLP) are used, with good results.[1] This kind of NN is characterized by its hidden layer(s) of neurons, which in a way that is not intuitive, attempt to separate the two classes involved. The standard learning paradigm for these NN, back-propagation (or one of its countless variants), induces changes in these hidden neurons that minimize a pre-established cost function. The user has few or no control at all over the task each of these hidden neurons executes. This is sometimes seen as a drawback of MLP, particularly in critical applications, where a full understanding of the inner workings of the NN would be desirable.

Also important is the fact that standard NN are heavy in computational terms. To know the output of a MLP to a pattern, the output of each neuron in the hidden layer must be evaluated. For NN that are to be used at very high rates, like the ones being considered in this study, this can be a serious limitation.

We present in what follows a brief comparison between three different types of neural networks: **multi-layer perceptrons**, **binary neural trees** and **modular neural networks**. This is illustrated with a simple example of a two-class separation in two dimensions; we will focus on the differences in architecture only: no learning schemes are considered at this level.

Consider the task of separating the two classes A and B depicted in fig. 1, in a two-dimensional space. Because the two sets are not linearly separable, more than one neuron is required to perform the classification. It can be shown that three neurons, approximating the thick curve in the figure, are enough to solve the problem.

[1] See [6] for an interesting application in CMS

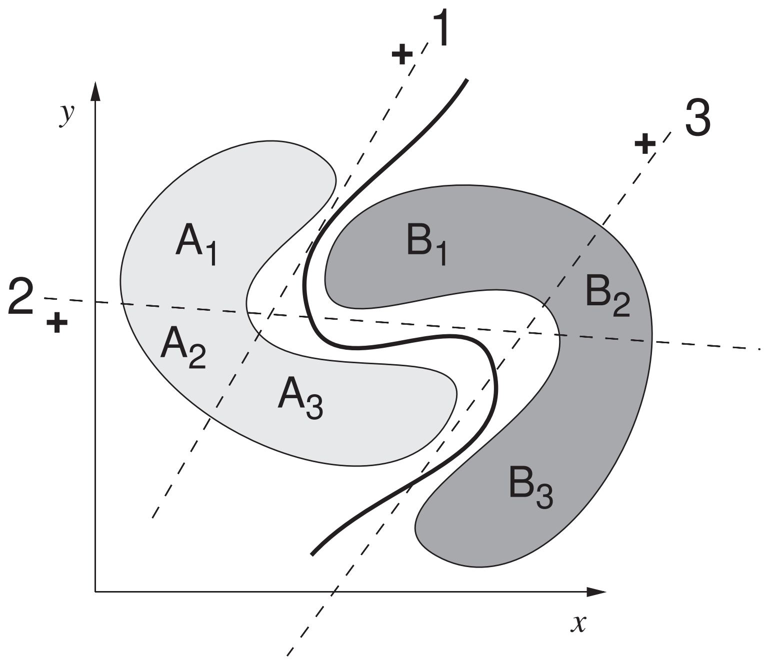

Figure 1: Class separation with a neural network

3.1.1 Multi-layer perceptron

To simplify, consider binary neurons, that output 1 if a pattern is on the "+" side of each dashed line in the figure, or 0 otherwise. If we compute the output of all three neurons to each subset A_i and B_j, the results will be the following:

Set	N_1	N_2	N_3	$N_1+N_2+N_3 > 1.5$
A_1	1	0	1	1
A_2	1	1	1	1
A_3	0	1	1	1
B_1	0	0	1	0
B_2	0	0	0	0
B_3	0	1	0	0

Notice how the three neurons project the sets A and B into a 3-dimensional space where the separation becomes possible with a single hyperplane (output neuron), defined by $N_1+N_2+N_3 > 1.5$.

When implemented, a single hidden neuron `w` involves 2 sums, 2 multiplications and one comparison:

```
if(w[0]+w[1]*x+w[2]*y>0)
  output=1
else
  output=0
```

Because there are three hidden neurons and one output neuron (which exists in a 3-dimensional space), the total number of operations involved in classifying a pattern is 9 sums, 9 multiplications and 4 comparisons. In general, an MLP consisting of H hidden neurons and 1 output neuron, operating on a space with D dimensions, will require

$$\begin{array}{ll} H \cdot (D \text{ sums } + D \text{ multiplications} + 1 \text{ comparison}) + & \textit{(hidden layer)} \\ +H \text{ sums } + H \text{ multiplications } + 1 \text{ comparison} & \textit{(output neuron)} \end{array}$$

Neglecting the comparisons (or, more generally, the sigmoid activation function for each neuron) and treating the sums and multiplications on an equal footing, this gives $\sim 2DH$ operations.

3.1.2 Binary neural tree

A faster version of the previous algorithm is possible, by a simple rearrangement of the computations. Consider the following "code":

```
if(output(neuron2)==1)
  evaluate_neuron3
else
  evaluate_neuron1
```

In spite of performing the same task as the MLP (as can be seen in fig 1 or in the table), in this case, no output neuron is required, and only two "hidden" neurons have to be computed. The number of operations is thus reduced by a factor > 2 in comparison with MLP. Only 4 sums, 4 multiplications and 2 comparisons are needed in this particular case.

When estimating the general computational cost of this architecture, we suppose optimistically that the neural tree is as compact as possible. In this case, its depth is $H' = \ln_2(H-1)$, and because there is no output neuron[2], the total number of operations increases as $\sim 2D \ln_2 H$.

3.1.3 Modular neural network

A modular neural network is in some sense an extended version of a neural tree, because the initial classification task is split between several different processing units (but not necessarily of the same kind), in a cascaded process. In our implementation, each pattern is classified in one of four classes, each class being separately processed by a simple perceptron. This way, the MNN process data at a constant rate (in contrast with binary neural trees, where a pattern follows a path of variable depth), and the time taken to have a decision is minimized, since in all cases, only one test and one neuron are evaluated: the number of computations is roughly $2D$, the smallest of all three cases considered.

If the full input space can be easily divided in a few different categories, then it may be more convenient to split it up *before* supplying it to the NN, and thus having specialized neurons (ore, more generally, specialized modules) for each sub-set, rather than having each and every neuron trying to separate the whole input set (as in the MLP). Putting it differently, the general problem of distinguishing an e/γ cluster from a jet will be divided in several

[2] All neurons being output neurons is an equivalent point of view.

(hopefully easier) problems. For instance, the barrel and endcap regions can be treated separately. More on this in sect. 3.2.4 below.

As a summary of the number of operations (or time consumption) respectively for modular neural networks, binary neural trees and multi-layer perceptrons, the following relation holds:

$$1 : \ln_2 H : H,$$

where H is in any case the number of processing neurons in the NN.

We now turn to the way the information in the ECAL crystals is supplied to the NN.

3.2 Pattern pre-processing

The Level-1 trigger included some processing of the fine-grain structure of the electromagnetic shower (along η only). However, this processing was not optimum due to the fixed geometry of the trigger cells; if a shower originates close to the boundary,[3] the energy will be significantly split among two neighbouring cells, degrading the performance of the Level-1 trigger.

The goal of the algorithm described here is to analyse the microscopic structure of the shower, in an attempt to resolve, e.g., π^0 vs. γ ambiguities. This is done by considering the detailed, 2-dimensional profile of the cluster, and using a neural network to classify it.

In spite of being implemented in a CPU farm, the second level algorithm should be as simple (thus, as fast), as possible. In terms of neural networks, this means on the one hand, using the smallest possible architecture, and on the other hand, the smallest possible dimension for the input space. As discussed above, a modular neural network (MNN) was considered, and so, the problem of using the smallest possible architecture is reduced to using the minimum number of neurons (see sect. 3.2.4). We shall present in the following the guidelines in defining what we consider to be the input space, or, what is a "pattern".

3.2.1 Symmetrization

The symmetries present in the calorimeter ($\eta \to -\eta$, $\varphi \to -\varphi$) suggest the implementation of some kind of "symmetrization" of the array of crystals. A possible implementation of this "symmetrization" is the following: consider a matrix of $(2n+1) \times (2n+1)$ crystals,[4] with the highest crystal in its centre; attach axis η and φ to the central crystal, thus labelling the crystals by $(\eta, \varphi) = (-n \ldots n, -n \ldots n)$; add to each crystal at $\eta \geq 0$ the contents of the crystal at $-\eta$; then, add to each crystal at $(\eta \geq 0, \varphi \geq 0)$ the contents of the crystal at $(\eta, -\varphi)$; finally, discard crystals with $\eta < 0$ or $\varphi < 0$ (a copy of them exists already at $(\eta \geq 0, \varphi \geq 0)$. This is interesting, because it allows a factor $\sim$4 reduction[5] in the number of crystals supplied to the algorithm, without (significant) loss in the information about the shower structure. In our implementation, an initial matrix of 13$\times$13 crystals is considered; the procedure just described folds this matrix into a 7$\times$7 component matrix. This is depicted in fig. 2 below.

3.2.2 Normalization

Two words about the pattern normalization. In order to make the algorithm independent of the energy, we normalize the patterns, dividing them by its highest component, which is, by

[3]If "close" means "less than half a crystal away", the cell boundary occupies 30% of the total cell area.
[4]We take $n = 6$
[5]Actually, in the case of 13$\times$13 to 7$\times$7 reduction, a factor $\sim$3.5

construction, $(\eta, \varphi) = (0, 0)$. Then, we take the square root of each pattern component. The reason for this has to do with the few orders of magnitude that separate the core from the tails in the shower, and the peculiar behaviour (infinite slope) of the SQRT near the origin. Since we are trying to distinguish clusters that mainly differ in the tails (remember the core is normalized to unity), the SQRT helps in amplifying the difference in this region (i.e., near the origin) (see figs. 2 and 3).

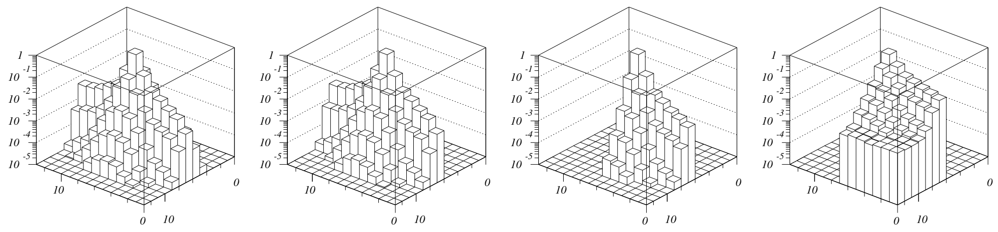

Figure 2: Pattern pre-processing. The histogram shows the average electron/photon shower, as it is folded and summed (three plots on the left). The fourth plot is the square root of the third. This is the 7×7 matrix that is supplied to the trigger algorithm

Fig. 3 below shows the contents, crystal by crystal, of the average pattern for e/γ and QCD, in the barrel, for low luminosity. This is just a one-dimensional rearrangement of the fourth histogram in fig. 2.

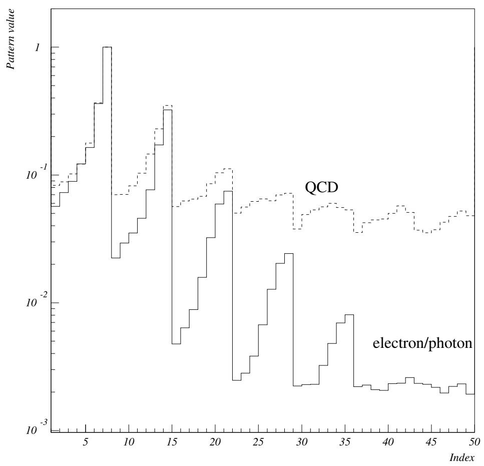

Figure 3: Average patterns for e/γ and QCD.

3.2.3 Endcap processing

Due to the geometry considered for the endcap, where the crystals coverage in η and φ increase with $|\eta|$, it was necessary to either adapt the algorithm to this region (and therefore have an algorithm that depends on the region of application), or remap the endcap region into cells of constant $\Delta\eta \times \Delta\varphi = 0.0145 \times 0.0145$ (see fig. 4). We chose the latter option, and performed this study with a uniform distribution of energy: if a crystal in the extreme endcap contains, say, 9 GeV, each of the 9 cells corresponding to it will have 1 GeV. It is possible that a more elaborated distribution of the energy among the cells could improve the results in this region.

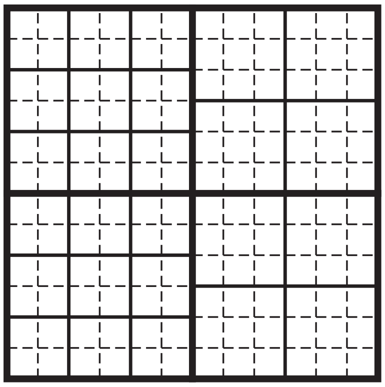

Figure 4: Endcap detail. Solid lines correspond to ECAL crystals. Dashed lines correspond to Level-2 "crystals". The energy deposited in each ECAL crystal is divided by the corresponding Level-2 "crystals"

3.2.4 Angular orientation

Consider now the clusters produced by a γ (or e) and a $\pi^0 = \gamma\gamma$. In the case where the γ does not pair-convert (or the e doesn't undergo bremsstrahlung), the π^0 should be distinguishable by the two peaks, *with isotropic angular distribution*, it produces in the calorimeter. Even if the γ converts (or the e radiates), the two clusters will tend to have larger φ than η dispersions, due to magnetic field bending. Therefore, information about the angular orientation of the cluster(s) will be incorporated in the decision. In fact, this will be the modularity condition for assigning each pattern to a single neuron. The classification task is divided in four sub-tasks, according to the region where the clusters' tail is more important:

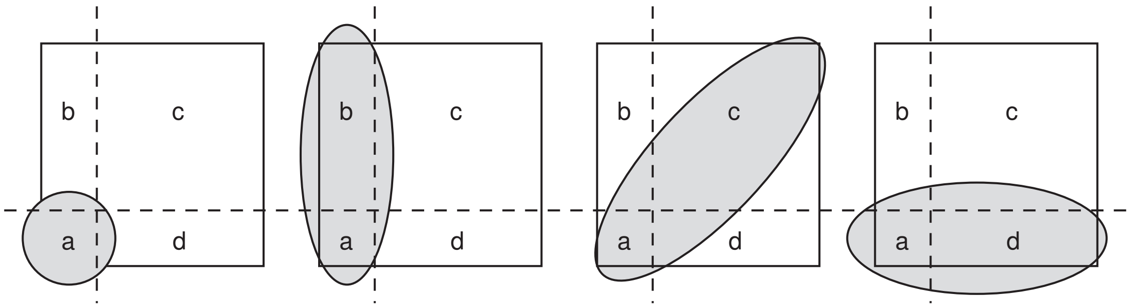

Figure 5: Separation of tasks in the modular neural network. There will be one specialized neuron for each of type of shower represented

There are several ways of implementing this feature. Without concerns of speed optimization, we compute

$$\hat{\eta^2} = \frac{\sum S(\eta,\varphi)\cdot\eta^2}{\sum S(\eta,\varphi)} \quad \text{and} \quad \hat{\varphi^2} = \frac{\sum S(\eta,\varphi)\cdot\varphi^2}{\sum S(\eta,\varphi)}, \quad S = \sqrt{E}$$

and use one neuron for each of the following 2×4 regions:

Barrel			**Endcap**		
Neuron	$\hat{\eta^2}$	$\hat{\varphi^2}$	Neuron	$\hat{\eta^2}$	$\hat{\varphi^2}$
1	< 0.2	< 0.3	5	< 1.4	< 1.9
2	< 0.2	> 0.3	6	< 1.4	> 1.9
3	> 0.2	< 0.3	7	> 1.4	< 1.9
4	> 0.2	> 0.3	8	> 1.4	> 1.9

The coefficients in the previous table are the ones that provide the best separation between e/γ and jets in the $\hat{\eta^2}$ and $\hat{\varphi^2}$ variables.

3.3 Network training

After fixing the structure of the network, the weights `w` have to be determined. Several learning schemes exist to accomplish this, the most straightforward of all being the perceptron learning rule [7]. As a behavioural paradigm, it requires the existence of two sets of data, one containing signal (e/γ), and one containing background (jets) samples. During the training loop, each pattern is supplied to the network (in a random sequence), and its output evaluated. Unless the network gives the correct output, a small correction is introduced to its weights, so that in the next cycle the rejection ratio (or the trigger rate) improves. However, this improvement is bounded by the condition that the efficiency remains above a given threshold. In this study, the efficiency in the signal set was required to exceed 98%.

4 Results

The algorithm described above was applied on the data sets, and the results obtained were summarized in the following table:

$\mathcal{L}$ (cm^{-2}s^{-1})	QCD removed (%)
10^{33}	78
10^{34}	71

The effect of taking the SQRT in building the pattern, and of considering several different architectures for the NN is quantified in the following table, for low luminosity data. The label "Linear" means that no SQRT was applied to the patterns. The case with one neuron corresponds to a single classifier for the whole calorimeter; when two neurons were considered, the endcap and the barrel were processed by independent neurons. Finally, when $N = 8$, the full modular neural network, as discussed above, is used.

Energy scale	Number of neurons	QCD removed (%)
Linear	1	58
Linear	2	65
Linear	8	70
SQRT	8	78

When evaluating the trigger rates after the Level-2 trigger, we should note that the increased cluster-E_T resolution will account for a supplementary reduction factor; this is shown in the following two figures. The bottom (labelled "L1") curve shows what the Level-1 trigger E_T cut should be, in order to have $94\ldots98\%$ efficiency for electrons and photons of 20 GeV (fig. 6) and 40 GeV (fig. 7) transverse energy. For a given Level-1 threshold, we may ask what is the maximum Level-2 E_T cut that guarantees a certain efficiency. This is depicted in the three curves labelled "L2" for 94, 96 and 98% efficiency. We see that the gain in threshold is approximately 1.5 GeV for 98% at 20 GeV, and $\sim$3 GeV for the same efficiencies at 40 GeV.

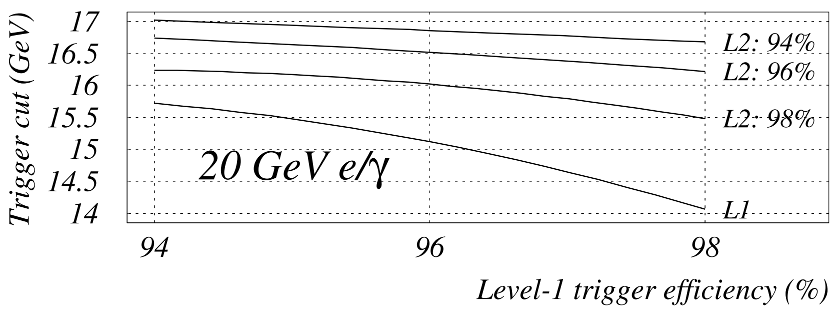

Figure 6: Values of the trigger E_T threshold, for various Level-1 and Level-2 target efficiencies. These data were obtained with e/γ samples of $E_T = 20$ GeV, at high luminosity

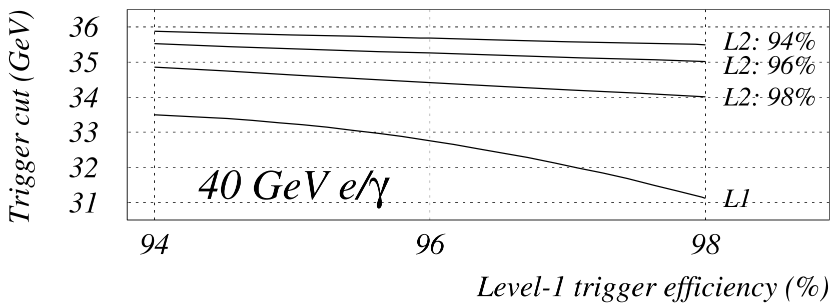

Figure 7: Same as above, for 40 GeV e/γ data

We pointed out in [4] that after specifying the trigger E_T threshold, one has to account for the dispersion of the cluster's energy, as seen by the trigger, when determining the appropriate trigger rate. So, for instance, in the case of 20 GeV E_T e/γ clusters, the E_T threshold has to be set at $\sim$ 14 GeV for the Level-1 trigger, in order to have 98% signal efficiency, but can be increased to $\sim$ 15.5 for the Level-2 trigger. Similarly, for 40 GeV E_T, the threshold can be increased from $\sim$ 31 GeV to $\sim$ 34 GeV when passing from the first to the second level triggers. By inspection of fig. 9, this factor alone is seen to be responsible for an approximate gain of 40% in rate (high luminosity).

Finally, figures 8 and 9 below contain the effect of the Level-2 trigger, as described in this note, has on the calorimeter trigger rates. We see that the rate is consistently decreased by factors around 5, in both cases of high and low luminosity. Considering the 40% gain discussed in the previous paragraph, the global Level-2 rejection factor is thus $\sim$ 7.

References

[1] CMS Technical Proposal

[2] CMS Calorimeter Trigger Group, *Preliminary specifications of the baseline trigger algorithms*, **CMS TN/96-010**

[3] T.Sjostrand, *PYTHIA5.7*, **CERN-TH. 7112/93**

[4] R.Nóbrega, J.Varela, *The CMS e/γ trigger. Simulation study with CMSIM data*, **CMS TN/96-021**

[5] `http://www1.cern.ch/NeuralNets/nnwInHepRefRev.html`

[6] S.Banerjee, A.Khan, *Application of Neural Networks in Detecting Higgs at CMS*, **CMS TN/96-023**

[7] V.Roychowdhury, *et.al.*, *Classification of Linearly Nonseparable Patterns by Linear Threshold Elements*, **IEEE Transactions on Neural Networks**, March 95

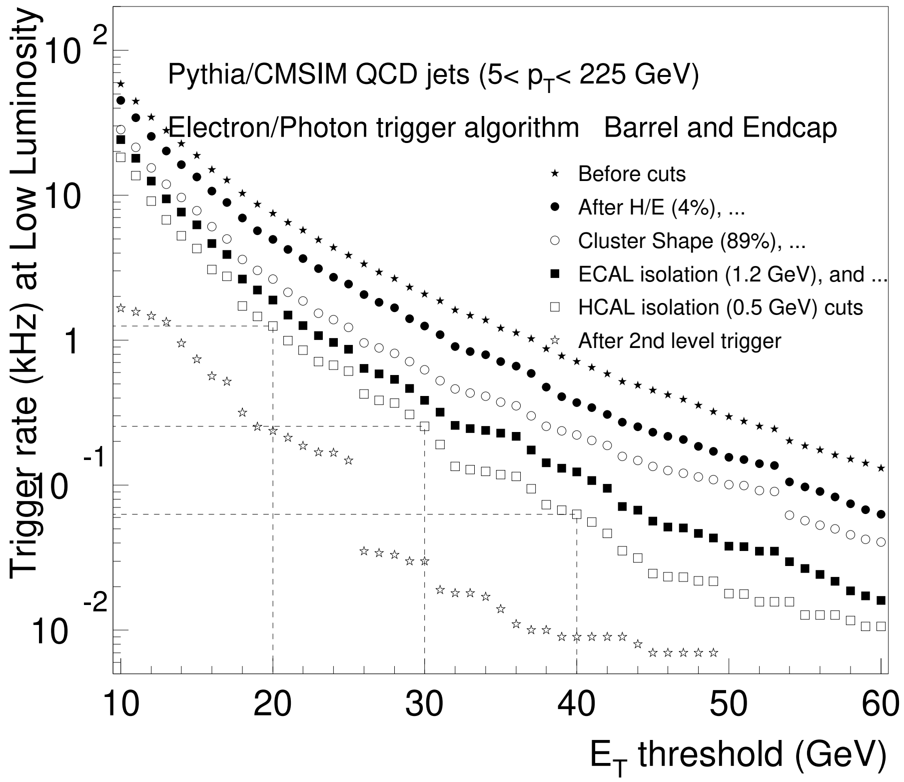

Figure 8: Trigger rate at low luminosity, as each trigger condition is included

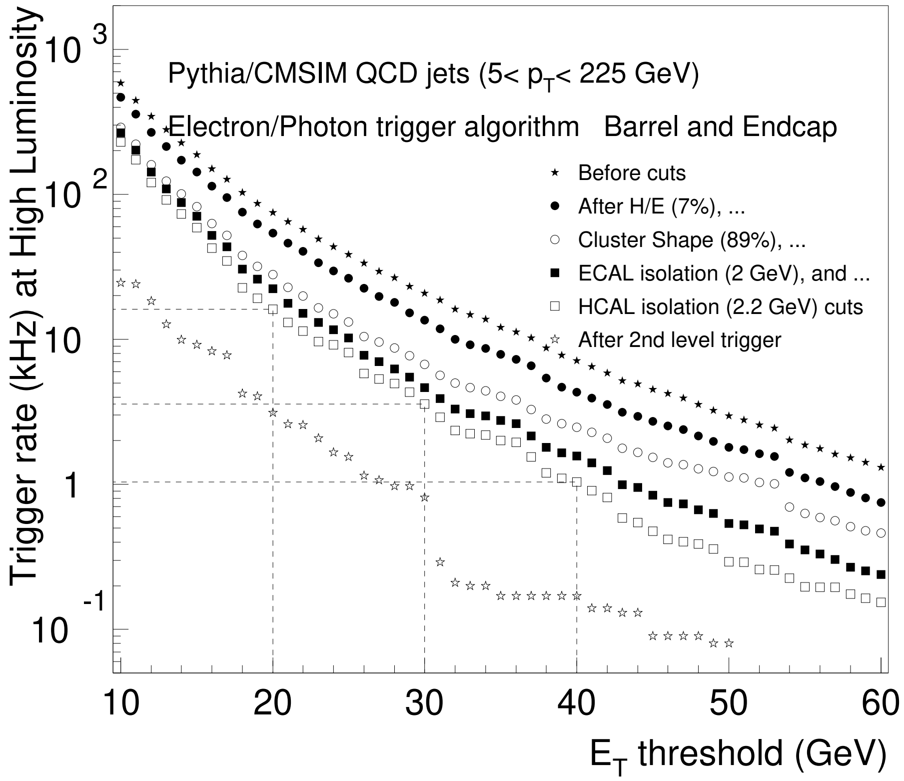

Figure 9: Same as above, for high luminosity