Stability study of the neural network at particle physics detectors

Tamás Majoros
Intelligent Embedded Systems Research Laboratory
Faculty of Informatics
University of Debrecen
Debrecen, Hungary
majoros.tamas91@gmail.com

Balázs Ujvári
Department of Experimental Physics
Faculty of Science and Technology
University of Debrecen
Debrecen, Hungary
ujvarib@gmail.com

Abstract— **Neural networks are used as triggers at high-energy physics detectors. These triggers can separate the event that must be saved for later analysis from the other events or noises. Using the raw data of the detector, the signal and the background can be separated offline. After separation, sets of signals and backgrounds can be used to train the neural network. A gas-filled detector (multiwire proportional chamber) was used to study the trigger at different noise levels to find the most stable neural network that tolerates the random hits. The ratio of the recognized and the unrecognized signal and background events is used for the measurement. Its stability is part of the systematical uncertainty.**

Keywords—neural network, pattern recognition, multiwire proportional chamber, experimental physics detector, FPGA

I. INTRODUCTION

Neural networks were used in experimental high-energy physics even some decades ago for analyzing the data of large and complex detectors [1]. The reconstruction time and accuracy were good enough, but in physics the estimation of the uncertainties is very important and with the neural network it is hard to calculate. It was used to reduce the background to find rare events [2], neural networks helped to select good quality data without additional subdetectors. These experiments used massive parallel processing, and the decision time was in the order of seconds.

The next big step of the neural network in the high-energy physics was the triggering. It was possible to use neural networks to identify the interesting events in some microseconds [3], after some low-level hardware logic. The recent big detectors are using neural network for track finding, particle identification [4], and there are running projects to use neural network at the hardware level of the data acquisition [5]. That means huge amounts of data have to flow through the network to select the important events in the order of hundreds of nanoseconds. Integrating the neural network at this hardware level needs fine-tuned solutions. Since the hardware level of these big detectors is not accessible, a dedicated setup should be used for learning the way to implement the hardware level neural network. A simple detector with special front-end electronics is used to show it in this paper.

The gas-filled detectors, like the multiwire proportional chamber (MWPC), and the time projection chambers (TPC) are widely used in large volumes, and since the gases are low density materials, the particle can move without interaction with the detector itself along its path, thus no additional corrections are needed for the reconstruction. Muon is a temporary particle, which will decay into stable electron and neutrinos, but lives enough to fly several kilometers before the decay. Muons can be used to measure the short-living particles (Higgs-boson, Z-boson) that decay almost immediately and cannot reach the detector, instead, they decay into muons (Higgs-boson can decay into four muons).

From the parameters of the muon these particles can be measured. These short-living particles can be created by colliding particle beams at dedicated particle accelerators. These accelerators work on very high collision rates in the order of 10 millions collisions per second (10 MHz), and the detectors at the collision points have to be prepared for these rates. At 10 MHz, the time difference between two collisions and therefore two muons from two different decays is 100 ns. Time precision measurements are needed to separate the events, in order to avoid the false measurements.

There are fast detectors, such as semiconductors, scintillators, Cherenkov-detectors, and their signal can be used in this precision regime. Unfortunately, the gas-filled detectors are slower, so additional calibration and data analysis need to operate in time precision, high collision rates measurements. The particle physics colliders are complex and expensive devices. In the early phase of detector developments, cosmic muons are widely used to test the hardware and software. Cosmic particles hit the atoms in the upper atmosphere and create temporary particles that decay to muons. These cosmic muons reach the surface of the Earth, and can be detected in every laboratory.

II. MWPC

At the University of Debrecen readout electronics were developed for MWPCs to study the time precision measurements with gas-filled detectors. The setup consists of four layers MWPC (fig. 1).

Fig. 1. MWPC at University of Debrecen

The muon goes through the setup and ionizes the gas (argon 82%, carbon-dioxide 18%), creating few (order of ten) electrons. Every layer (fig. 2) has 16 sensitive anodes (+1500 V, 50 µm, the thinner in the photo) and 16 grounded field wires (100 µm).

Fig. 2. One layer of MWPC

The electric field accelerates these primary electrons towards the anodes. These electrons will have enough kinetic energy to kick off new electrons and even these secondaries can ionize the gas, then an electron avalanche (order of millions) reaches the anode wires. The field wires are read out, the potential changes by electron avalanches on the anodes capacitively coupled to the field wires. A hit means that this analog signal is above a given threshold, which is the 1, otherwise the wire is 0 for the path reconstruction. A typical signal looks like the one shown on fig. 3.
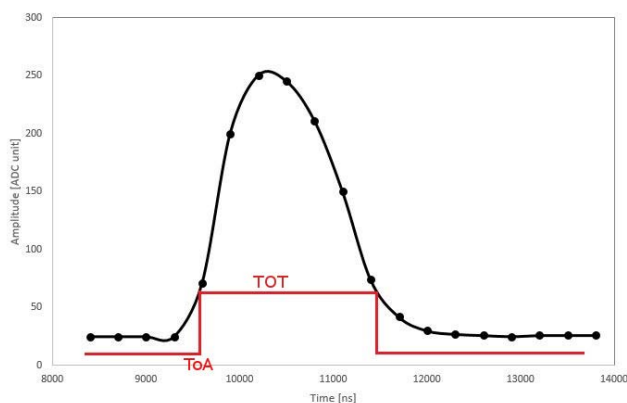

Fig. 3. A typical signal on field wires of MWPC

III. RECONSTRUCTION OF SIGNALS

For path and arriving time reconstructions, the important parameters are the Time of Arrival (ToA), Integral, Time Over Threshold (TOT). On fig. 3 black dots are the amplitudes measured by a 3 MS/s ADC. From the ADC measurement, the integral can be easy calculated, but the arriving time can be measured only by reconstructing the shape of the signal from the discrete ADC values.

The uncertainty by the signal shape reconstruction with the 3 MS/s ADC can be order of 10 ns. With a given comparator level, ToA and TOT can be measured with a comparator and a precise timer. The integral of the signal (that is proportional to the number of electrons arrived at anode) and the TOT have a non-linear relationship that can be determined with a calibration process for each wire.

With amplifiers, comparators and an FPGA 1-2 ns precision can be reached to measure ToA and TOT, and the integral can be calculated from the TOT used for the calibrations. Since there is no need for fast ADCs, such front-end electronics is the most cost-effective setup for measuring the arriving time of the muons at high collision rate accelerators. For path reconstruction the wire position and the integral of the signal on the given wire are used, but the integral is used only for weighted average, its precision is not crucial. In this case the hit is the signal above the comparator threshold, and the hit's parameters are the ToA and TOT.

IV. APPLICATION OF NEURAL NETWORK

The comparators are used to suppress the noise, but in a gas-filled detector even one free electron can create an electron avalanche, therefore a high rate of false hits is expected. These random hits are uncorrelated. An effective trigger can be operated with a neural network that trained to recognize the structure of the hits of the passing muon. The trigger separates signal and background noise using not the full measurements, but an aggregated or lower resolution sample to decide as soon as possible. The full resolution measurements are buffered and read out only after a positive trigger answer to lower the necessary bandwidth of the readout. At modern experiments not only the signal and the noise are separated, but with hierarchical trigger systems different particles can be separated by the different event structures, and only the events are read out which are among the physical goals of the detector.

In this detector the full measurements are the ID of the wire, the ToA, the TOT for every hit, for every wire where the signal is above the comparator threshold. The front-end FPGA buffers these values, but send only one bit per wire whenever the signal at the given value was above the threshold (1) or not (0). In every 160 ns front-end FPGA sends 64 bits to the FPGA of the neural network to evaluate. If the 64 bits representation is a muon candidate, the full measurement with the precise timing (1-2 ns) is read out.

These gas-filled detectors are sensitive to high voltage setups and to gas flows. There can be random scratches on the wires, and the properties of the noise are not stable. In this study the neural network was trained with long measurements, thus the trigger efficiency is shown with these measurements. Then random zeros and ones are added to the measurement to estimate the stability of the neural network.

V. ROOT AND TMVA

ROOT is an object-oriented program and library developed by CERN, which provides platform independent access to a computer's graphics subsystem and operating system using abstract layers. The packages provided by

ROOT include histogramming, graphing, curve fitting, statistics tools, matrix algebra, standard mathematical functions, 3D visualizations, multivariate data analysis (e.g. using neural networks).

The Toolkit for Multivariate Analysis (TMVA) provides a ROOT-integrated environment for the processing, parallel evaluation and application of multivariate classification and regression techniques. All multivariate techniques in TMVA belong to the family of supervised learning algorithms, which make use of training events, for which the desired output is known, in order to determine the mapping function that either describes a decision boundary (classification) or an approximation of the underlying functional behavior defining the target value (regression). The software package provides training, testing, performance evaluation algorithms and visualization scripts [6].

TMultiLayerPerceptron is a class in the TMVA package. It describes a multilayer perceptron (MLP), which is a class of the feedforward artificial neural network. It contains facilities to train the network and to use the output. The input layer is made of inactive neurons (returning the normalized input), while hidden layers are made of neurons with an activation function, and output neurons are linear. The basic input is a training and a test event list. For classification jobs, a branch must contain the expected output. Different learning methods are available. A neural network can be instantiated by calling a *TMultiLayerPerceptron()* constructor. The network is described by a simple string: the input/output layers are defined by giving the branch names separated by commas. Hidden layers are described by the number of neurons and the activation function. Training and test are two cuts defining events to be used during neural network training and testing.

The most common algorithm for adjusting the weights that optimize the classification performance of a neural network is the so-called back propagation, which belongs to the family of supervised learning methods. During the learning process the network is supplied with training events. For each training event the neural network output is computed and compared to the desired output (1 for signal, 0 for background events). An error function measures the agreement of the network response to the desired one. The set of weights that minimizes the error function can be found using the method of gradient descent, provided that the neuron response function is differentiable with respect to the input weights. Starting from a random set of weights, the weights are updated by moving into the direction where error function decreases most rapidly.

VI. COMPARISON OF ACTIVATION FUNCTIONS

Neural networks with same structures but different activation functions were trained to compare the effect of the type of the activation functions. The input layer contains 64 neurons as 64 wires are read out from MWPC. These normalize the incoming zeros and ones. Two hidden layers with five neurons and a linear output neuron in the output layer were used. Applicated activation functions were linear, gaussian, hyperbolic tangent (tanh) and sigmoid. The following figures (4-7) show the difference in classification efficiency between them on a logarithmic scale.
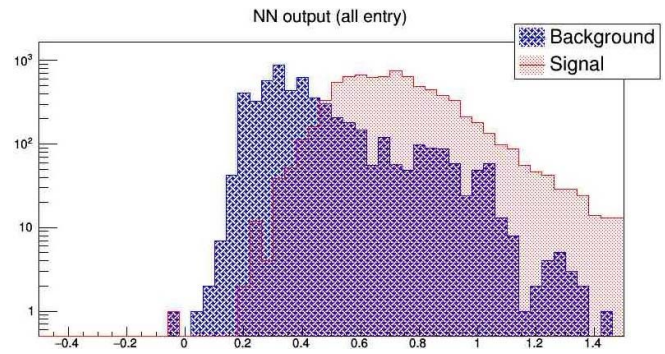

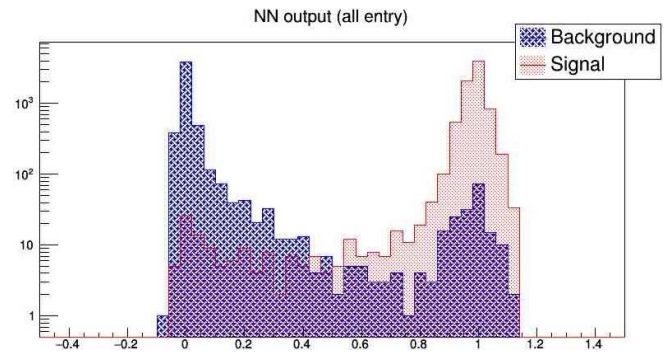Fig. 4. Neural network output with linear activation function

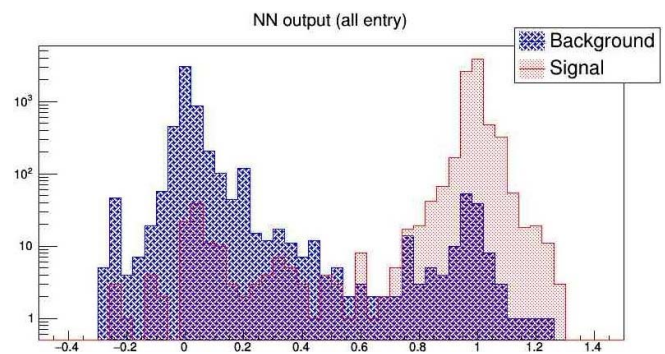Fig. 5. Neural network output with Gaussian activation function

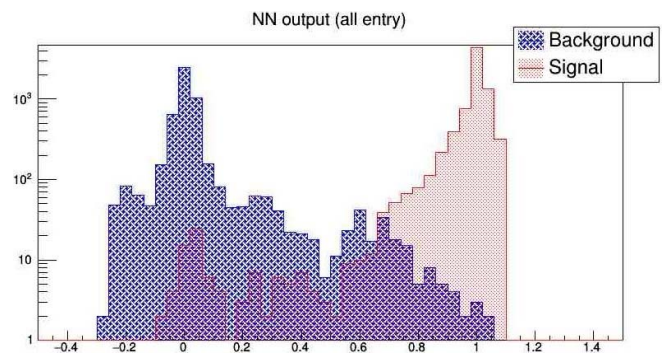Fig. 6. Neural network output with tanh activation function

Fig. 7. Neural network output with sigmoid activation function

The highest classification efficiency (lowest number of misrecognized inputs at optimal output threshold value) was reached with sigmoid function, therefore it is used in later sections.

VII. COMPARISON OF NETWORK STRUCTURES

Different network structures were compared to examine how the number of neurons and hidden layers affect the noise sensitivity of the trained neural network. Output

threshold value is selected at each case as the lowest value, where 90% of signal inputs are recognized as signal. Threshold values are shown on fig. 8-9. Ratio of recognized and misrecognized signal and background inputs on different neural network structures are shown on fig. 10-14 and table 1. Positive value of noise means the probability in percentage, that the wire is set to 1 (even it was 1 or 0). The negative noise value means the probability in percentage, that the wire is set to 0 (even it was 1 or 0). Network structure is described with the number of neurons in hidden layers.
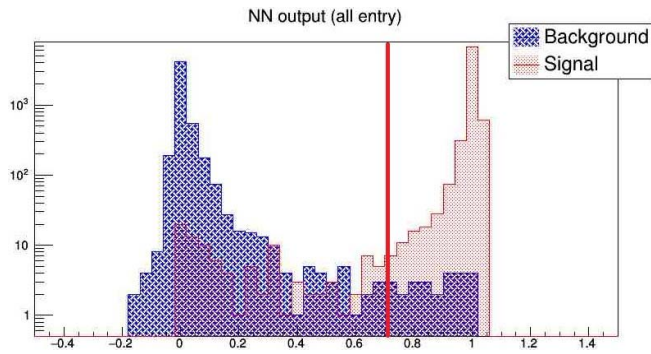
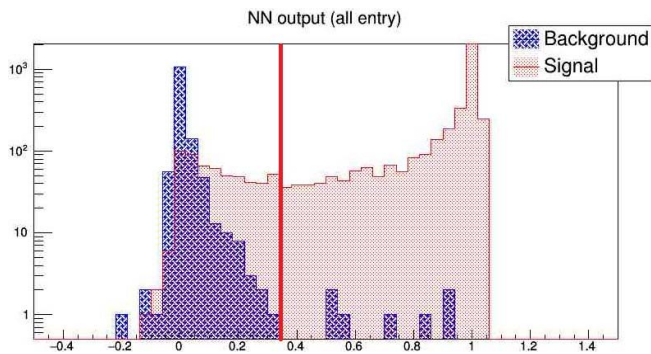Fig. 8. Neural network output to optimal input (red line: threshold value)

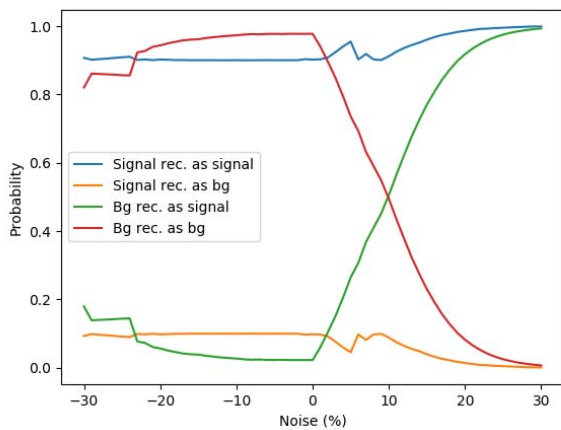Fig. 9. Neural network output to very noisy (-30%) input (red line: threshold value)

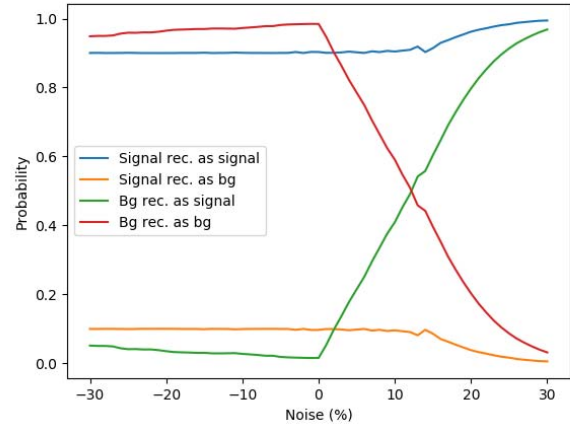Fig. 10. Input recognizing efficiency of a 3x3x3 network

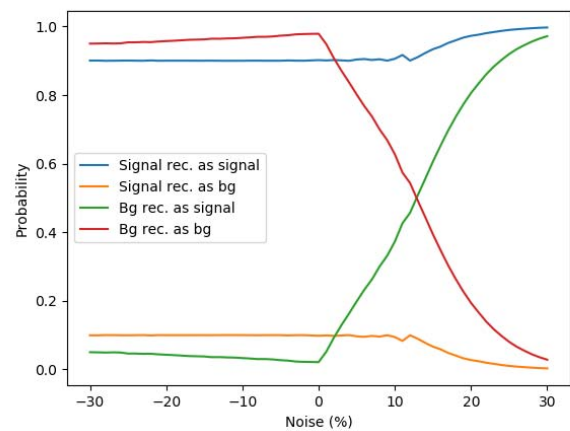Fig. 11. Input recognizing efficiency of a 5x5 network

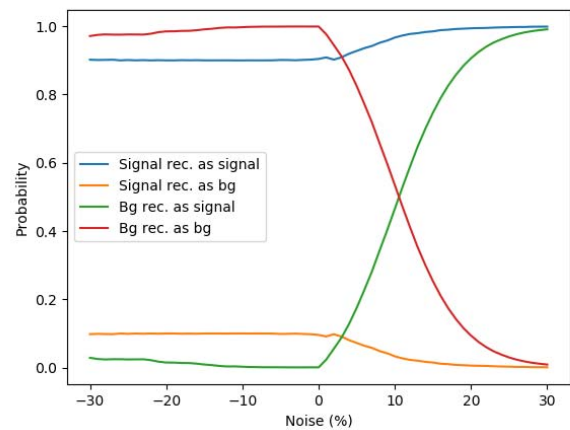Fig. 12. Input recognizing efficiency of a 5x5x5 network

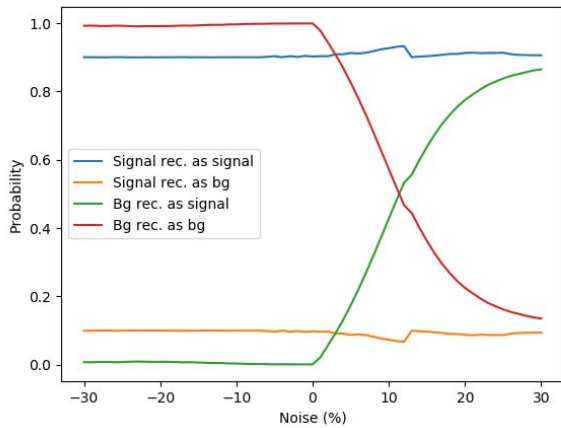Fig. 13. Input recognizing efficiency of a 10x10x10 network

Fig. 14. Input recognizing efficiency of a 20x20x20 network

	3x3x3	5x5	5x5x5	10x10x10	20x20x20
-5%	2.27%	1.84%	2.66%	0.09%	0.09%
-4%	2.32%	1.71%	2.54%	0.07%	0.11%
-3%	2.25%	1.64%	2.27%	0.06%	0.08%
-2%	2.27%	1.58%	2.19%	0.06%	0.08%
-1%	2.26%	1.53%	2.14%	0.08%	0.08%
0%	2.25%	1.55%	2.10%	0.06%	0.08%
1%	5.96%	5.31%	5.12%	2.34%	2.13%
2%	10.42%	9.79%	9.23%	5.53%	5.59%
3%	15.15%	13.69%	12.90%	8.77%	9.19%
4%	20.58%	17.84%	16.30%	12.80%	13.05%
5%	26.44%	21.58%	19.82%	17.51%	17.41%

Table 1. Background recognized as signal at different noise levels

As shown in the above figures and table, a more complex network, containing more neurons gives a better recognition result. If the background rate (number of background events per second) is much higher than signal rate, the background recognized as signal (false signal) uses the major part of the readout bandwidth, the more complex neural network trigger can reduce this false signal rate. For this study, the number of signal and background was in the same order of magnitude, but in real experiments the ratio of signal and background events have to be separated by the trigger, which can be 1:1000000. A fine-tuned neural network is necessary to manage the data flow.

VIII. FPGA IMPLEMENTATION

Synapses values and neuron offset values are exported from the training program. These values (and the network structure) are used for hardware implementation of the neural network in an FPGA. For this purpose, a Python program was made. This program reads the exported file and generates a synthesizable source code in Verilog hardware description language. Input layer neurons are implemented as multiplexers to save resources and processing time. Activation functions are implemented as look-up tables in

read only memory (ROM), where memory address is the input value of the function, and the content on that address is the function value. A threshold value can be set to get a single bit output (signal or background) from the network. A test bench file can be generated for validation purpose. It uses random input values and compares the output values of software-implemented and hardware-implemented networks.

A simplified version of the neural network with 32 inputs, 2 hidden layers (with 5 and 3 neurons, respectively) was implemented in FPGA fabric part of a Xilinx Zynq 7020 SoC device. This configuration uses 184 of the available 240 digital signal processing (DSP) units in the FPGA. Proper functionality is proven by using a generated test bench file. Three of the 2000 testcases failed, all of them were very close to the output threshold value, which is due to the smaller number representation precision in FPGA (for lower resource demand). 12.5 MHz clock frequency was used, and the latency of the network is four clock cycle. Since 80 ns is the trigger decision time and the front-end electronics can buffer order of 10 events, this setup can be used at high collision rate experiments to select the particle by the structure of the hits and reject background events or noises. With further optimization resource usage can be decreased, and more complex networks can be implemented at the cost of higher latency.

REFERENCES

[1] P. Cennini, S. Cittolin, J. P. Revol, C. Rubbia, W. H. Tian, D. Dzialo Giudice, …, S. Suzuki (1995): A neural network approach for the TPC signal processing. Nuclear Instruments and Methods in Physics Research A 356 507-513

[2] L. Litov (2003): Particle identification in the NA48 experiment using neural networks. Nuclear Instruments and Methods in Physics Research A 502 495-499

[3] B. Denby, P. Garda, B. Granado, C. Kiesling, J. Prévotet, A. Wassatsch (2003): Fast Triggering in High-Energy Physics Experiments Using Hardware Neural Networks. IEEE Transactions on Neural Networks, Vol. 14, No. 5

[4] B. Alessandro, F. Antinori, J. A. Belikov, C. Blume, A. Dainese, P. Foka, …, E. Vercellin (2006): ALICE: Physics Performance Report, Volume II. J. Phys. G: Nucl. Part. Phys. 32 1295-2040. doi:10.1088/0954-3899/32/10/001

[5] S. Neuhaus, S. Skambraks, F. Abudinen, Y. Chen, M. Feindt, R. Frühwirth, …, J. Schieck: A neural network z-vertex trigger for Belle II. arXiv:1410.1395v2

[6] A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss: TMVA 4 Users Guide