Regadío Carretero, A., Sánchez Prieto & Esteban, L., S. 2019, "Filtering of pulses from particle detectors using neural networks by dimensionality reduction", Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 942, art. no. 162372.

Available at https://doi.org/10.1016/j.nima.2019.162372

(Article begins on next page)

Filtering of pulses from particle detectors using neural networks by dimensionality reduction

Alberto Regadío[a,*], Sebastián Sánchez-Prieto[b], Luis Esteban[c]

[a]*Electronic Technology Area, Instituto Nacional de Técnica Aeroespacial, 28850 Torrejón de Ardoz, Spain*
[b]*Department of Computer Engineering, Space Research Group, Universidad de Alcalá, 28805 Alcalá de Henares, Spain*
[c]*Universidad Antonio de Nebrija, C/Pirineos, 55 28040 Madrid, Spain*

Abstract

This article presents a comparison between different filtering methods based on dimensionality reduction for pulses generated on particle detectors. This reduction has been carried out using Neural Networks (NNs). In particular, three topologies have been used: Autoencoders (AEs), Denoising Autoencoders (DAEs) and Restricted Boltzmann Machines (RBMs). A detailed explanation of these methods, a noise reduction analysis, filtering with simulated data and processing of pulses from a neutron detector have been carried out to verify the feasibility of using these NNs as pulse filters.

Keywords: Digital pulse processing, Pulse filtering, Noise, Dimensionality reduction, Denoising, Autoencoder, Restricted Boltzmann Machine, Neural Network

1. Introduction

When particles interact with particle detectors, pulses of current or charge are generated. In general, they are converted to voltage pulses at the output of the preamplifying stage and analyzed in successive stages. The features of these pulses carry several useful information about the incident particle such as its type, energy, or angle of impact [1].

Output pulses of particle detectors are always mixed with noise that limits the accuracy of the information carried by them. In these systems, noise is mainly generated in the detector and in the readout circuit attached to the detector. Some of the noise is a result of fundamental physical processes such as the discrete nature of electric charge and therefore cannot be avoided. However, its effect can be reduced by implementing proper noise filtering strategies with analog and digital electronics.

Compared to analog filters, digital filters reduce the number of electronic devices needed. Among them, Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) are the most common in particle detectors [1]. In addition, they allow to implement more complex shapers and filters to improve the incoming

*Corresponding Author

Email addresses: regadioca@inta.es (Alberto Regadío), sebastian.sanchez@uah.es (Sebastián Sánchez-Prieto), lesteban@nebrija.es (Luis Esteban)

pulse quality. Examples of these methods include Lagrange interpolations [2], adaptive filtering [3], simulated annealing [4], wavelets [5, 6], genetic algorithms [7] and filtering using Singular Value Decomposition (SVD) [8, 9]. This last method is based on dimensionality reduction, concept explained in Section 2.

In this article, the dimensionality reduction will be performed using neural networks (NNs), namely Autoencoders (AEs), Denoising Autoencoder (DAEs) and a Restricted Boltzmann Machines (RBMs), which will be explained in detail in Sections 3 and 4, respectively. This new approach involves advantages that will be explained throughout the article.

The use of NNs on particle detectors is not new. They have been used for gamma/neutron [10] and alpha/gamma discrimination [11], particle identification coming from cosmic rays [12, 13] and nuclide identification [14–16]. The cited articles have in common that NNs, namely Adaline or Multilayer perceptrons, allow the identification of particles. However, in this paper, we propose their use to maximize the Signal-to-Noise Ratio (SNR), leaving the identification for later stages of the detection chain of the spectroscopy system.

The proposed filter isolates the complete pulse before being processed. Therefore, the proposed filters does not add of ballistic deficit with respect to the previous stages and is feasible for any type of detector. In other words, with FIR or IIR filters we cannot transform CR–RC pulses into CR pulses without ballistic deficit unless filters are used to isolate the pulse.

The rest of the paper is structured as follows: Section 5 shall be devoted to compare the filtering methods used in this paper. The results of applying these methods with simulated and real pulses are exposed in Section 6. Finally, the conclusions of this work close the manuscript.

2. Dimensionality reduction and manifold learning

A concept underlying dimensionality reduction is that of a manifold. Mathematically, it is a connected region embedded into a space that locally can be approximated by another space of lower dimension. A typical example of manifold is the Earth. We experience the surface of our planet as a two-dimensional plane, but it is actually a sphere embedded into a three-dimensional space. The definition of locality of each point of the manifold implies the existence of transformations that can be applied to move on the manifold from one position to a neighboring one. In the example of the Earth surface as a manifold, we can walk along the four cardinal points [17].

The formal definition of manifold add several constraints but it is part of the differential geometry and is out of the scope of this paper. In NNs, it tends to be used more loosely to designate a connected set of points that can be approximated well by considering only a small number of dimensions embedded in a higher-dimensional space. Each dimension corresponds to a local direction of variation.

AEs, DAEs and RBMs are subsets of NNs that carry out manifold learning by dimensionality reduction

[18]. They work as follows: given a vector of length (dimension) L, these algorithms assume that most input combinations $\boldsymbol{x} \in \mathbb{R}^L$ are invalid, and that valid inputs occur only along a manifold of dimension $H < L$ containing a small subset of points, in this case the pulse without noise. There will be variations in the output of the learned function occurring only along directions that lie on the manifold. For instance, in Figure 1, we can see two-dimensional data points that can be grouped onto a one-dimensional manifold. When the learning process is unsupervised (Figure 1(a)) the manifold is estimated and when the learning is supervised (Figure 1(b)) the learning data lie on the manifold. Once the NN is trained, the test data points (x_1, x_2 on the figure) are projected onto the one-dimensional manifold (h_1) and then they are projected again to (x_1, x_2) (Figure 1(c)).

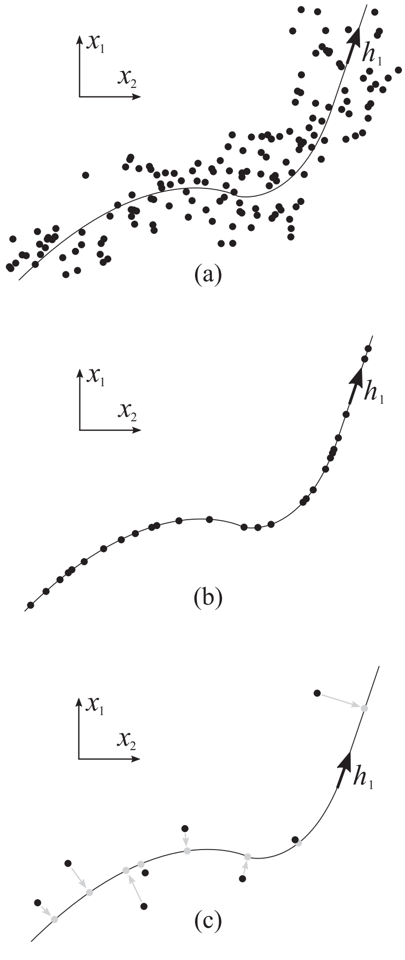

Figure 1: Dimensionality reduction from 2 to 1 (x_1, x_2 to h_1); (a) Unsupervised one-dimension manifold learning; (b) Supervised one-dimension learning; (c) Adjustment of x_1, x_2 using dimensionality reduction (figure adapted from [17])

In the context of pulse filtering and along this article, the input dimension L is the length of the pulse measured in cycles whereas the manifold dimension is H in which we assume that the data lies will be demonstrated along this manuscript. In fact, one key aspect is to find out how many dimensions have actually the input pulses. As we will see along this article, when pulses with similar shape are reduced to just one dimension, it is the pulse height.

The linearity of the algorithms is related with the fact that the data lie on a linear surface (i.e. a H-dimensional straight line). However, hereafter all the NNs in this paper will be linear because the vast majority of the filters used for these applications are also linear [1] and because the optimal weight values for these NNs can be derived directly by purely linear techniques [19]. Both input and output stages of NNs will have just one layer because the effect of adding more layers is usually negligible using linear activation functions.

3. Autoencoders (AEs) and Denoising Autoencoders (DAEs)

An AE is a type of unsupervised NN that codifies and decodifies an incomplete or noisy input signal $\boldsymbol{x} \in \mathbb{R}^L$ with the aim of reconstruct it at the output $\boldsymbol{y} \in \mathbb{R}^L$. AEs are also defined as NNs that are trained to learn the identity function [18]. They consist of two parts: an encoder function $\boldsymbol{h} = f(\boldsymbol{x})$ that carries out a dimensionality reduction form L to H and a decoder that produces a reconstruction $\boldsymbol{y} = g(\boldsymbol{h})$ restoring the dimensions from H to L. They have one internal layer $\boldsymbol{h} \in \mathbb{R}^H$ with H hidden units to store an autogenerated coding of the input.

A diagram of this architecture is presented in Figure 2. Note that constant inputs $x_0 = 1$ and $h_0 = 1$ have been included as a bias at the input and at the hidden layer respectively. Actually, both input and output layers can be composed by more than one layer but for simplicity these topologies are out of the scope of this paper.

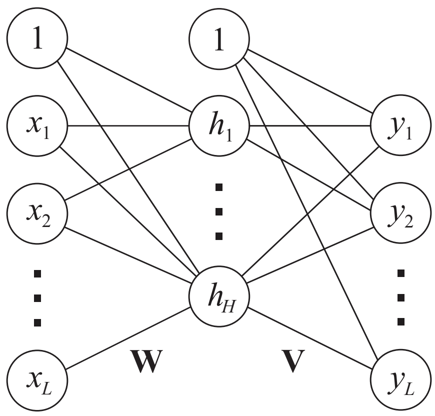

Figure 2: Diagram of a generic autoencoder.

Given an input pulse $\boldsymbol{x}$, the value of the hidden layer is

$$\widetilde{\boldsymbol{h}} = \mathbf{W}\boldsymbol{x} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{(L+1)\times H}$ is the first weight matrix. The value of H must be lower than L to perform the dimensionality reduction.

To cover most of the possible space solutions and to try to simulate the behavior of real neurons, an activation function should be applied to $\widetilde{\boldsymbol{h}}$:

$$\boldsymbol{h} = f(\widetilde{\boldsymbol{h}}) \tag{2}$$

Common activation functions in NNs include sigmoid, hyperbolic tangent, no activation function ($f(x) = x$) or the Rectified Linear Unit (ReLU(x)) whose result is x if $x > 0$ and 0 otherwise [20].

Once calculated $\boldsymbol{h}$, the estimated output value $\boldsymbol{y}$ is calculated reverting the dimensionality reduction in the following way

$$\widetilde{\boldsymbol{h}} = \mathbf{V}\boldsymbol{h} \tag{3}$$

where $\mathbf{V} \in \mathbb{R}^{(H+1)\times L}$ holds for the second weight matrix. Then, the same activation function used in (2) must be applied again to obtain the output $\boldsymbol{y}$

$$\boldsymbol{y} = f(\widetilde{\boldsymbol{h}}) \tag{4}$$

There exists a subset of AEs named Denoising Autoencoders (DAEs) that receives noisy data as input $\boldsymbol{x}$ and is trained to estimate the noiseless data $\boldsymbol{x}^*$ as its output $\boldsymbol{y}$. The main difference with respect to AEs is that the learning process of DAEs is supervised.

In order to break the linearity, several tests were performed using different activation functions such as hiperbolic tangent or ReLU. However, one desired feature on filters for particle detectors is linearity and, in fact, the best results were achieved without using any transfer function, i.e. with linear NNs. For these reasons, henceforth in this article, we replace (2, 4) by $\boldsymbol{h} = \widetilde{\boldsymbol{h}}$ and $\boldsymbol{y} = \widetilde{\boldsymbol{y}}$.

3.1. Learning rule

Both AE and DAE learn using backpropagation and gradient descent of the cost function j, following the chain rule for NNs. In learning algorithms, j is defined as as the deviation of the output $\boldsymbol{y}$ with respect to the training signal $\boldsymbol{x}^*$, that is

$$\boldsymbol{j} = \boldsymbol{x}^* - \boldsymbol{y} \tag{5}$$

According to backpropagation algorithm an the chain rule, when no activation function is used, all entries of the weight matrix $\mathbf{V}$ are updated according to the following formula:

$$\mathbf{V} \Leftarrow \mathbf{V} + \alpha(\boldsymbol{j} \otimes \boldsymbol{h}) \tag{6}$$

where $\otimes$ denotes the external product and α is the learning rate. After applying this equation, the input-to-inner layer errors $\boldsymbol{\delta}$ are calculated applying

$$\boldsymbol{\delta} = \mathbf{V}^{\top} \boldsymbol{j} \tag{7}$$

where $\mathbf{V}^{\top}$ is the transpose matrix of $\mathbf{V}$. This value is used to adjust the weights of the first layer $\mathbf{W}$

$$\mathbf{W} \Leftarrow \mathbf{W} + \alpha(\boldsymbol{\delta} \otimes \boldsymbol{x}) \tag{8}$$

These formulae are repeated for each input during a number of times established by the user to adjust the weights of the NN.

4. Restricted Boltzmann Machines (RBM)

RBMs are specific type of Boltzmann Machines and Markov Networks that can learn a probability distribution over its set of inputs in an unsupervised way.

Unlike the Bolzmann machines, RBMs are composed by two groups of units (commonly referred to as the "visible" and "hidden" groups respectively) set as a bipartite graph (Figure 3). They may have a symmetric connection between them and there are no connections between nodes within a group. By contrast, "unrestricted" Boltzmann machines may have connections between hidden units. This restriction allows for more efficient training algorithms that are available for the general class of Boltzmann machines, in particular the gradient-based contrastive divergence algorithm.

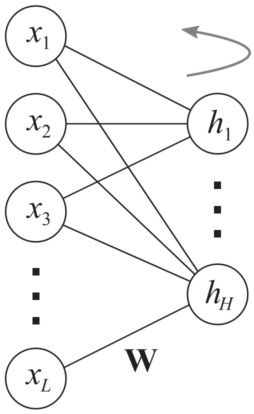

Figure 3: Diagram of a generic Restricted Boltzmann Machine. Bias $\boldsymbol{a_1}$ and $\boldsymbol{b_1}$ were omitted for clarity.

Given an input pulse $\boldsymbol{x}$, the value of the hidden layer is

$$\widetilde{\boldsymbol{h}} = \mathbf{W}\, \boldsymbol{x} + \boldsymbol{a_1} \tag{9}$$

where $\mathbf{W} \in \mathbb{R}^{L \times H}$ is the weight matrix and the $\boldsymbol{a_1}$ is the bias.

In the same way that AEs, to cover most of the possible space solutions and to try to simulate the behavior of real neurons, an activation function should be applied to $\widetilde{\boldsymbol{h}}$:

$$\boldsymbol{h} = f(\widetilde{\boldsymbol{h}}) \tag{10}$$

The obtained value $\boldsymbol{y}$ is

$$\widetilde{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{h} + \boldsymbol{b_1} \tag{11}$$

where $\mathbf{W}^\top$ is the transpose matrix of $\mathbf{W}$ and $\boldsymbol{b_1}$ is another bias, generally different from $\boldsymbol{a_1}$.

Finally, the same activation function must be applied again to obtain the output $\boldsymbol{y}$

$$\boldsymbol{y} = f(\widetilde{\boldsymbol{y}}) \tag{12}$$

In order to keep the linearity of the network and in the same way that with AEs and DAEs, we replace (10, 12) by $\boldsymbol{h} = \widetilde{\boldsymbol{h}}$ and $\boldsymbol{y} = \widetilde{\boldsymbol{y}}$.

4.1. Learning rule

In the same way that AEs, RBMs learns using backpropagation and gradient descent in an unsupervised way. However, in this case the output comes back from the hidden units, as shown in Figure 3, that yields

$$\boldsymbol{z} = \mathbf{W}\boldsymbol{y} + \boldsymbol{b_1} \tag{13}$$

Then, if we define $\mathbf{P}$ and $\mathbf{N}$ as

$$\begin{aligned} \mathbf{P} &= \boldsymbol{h} \otimes \boldsymbol{x} \\ \mathbf{N} &= \boldsymbol{z} \otimes \boldsymbol{y} \end{aligned} \tag{14}$$

the weights and bias become

$$\begin{aligned} \mathbf{W} &\Leftarrow \mathbf{W} + \alpha(\mathbf{P} - \mathbf{N}) \\ \boldsymbol{a_1} &\Leftarrow \boldsymbol{a_1} + \alpha(\boldsymbol{x} - \boldsymbol{y}) \\ \boldsymbol{b_1} &\Leftarrow \boldsymbol{b_1} + \alpha(\boldsymbol{h} - \boldsymbol{z}) \end{aligned} \tag{15}$$

Similarly to AEs and DAEs, these formulae are repeated for each input during a number of times established by the user to adjust the weights of the NN.

5. Comparison among methods

In this section, AEs, DAEs and RBMs are compared to SVD filtering [9]. The reason for this comparison is that these methods share common features: (a) the pulse must be isolated in an input $\boldsymbol{x}$; (b) both NNs and SVD filtering are linear; (c) all of them can be used for dimensionality reduction as explained in Section 2.

SVD is a linear procedure that transforms a set of N observations $\boldsymbol{x}$ of length L grouped in $\mathbf{X} \in \mathbb{R}^{N \times L}$ to a new coordinate system in which the value of the first coordinate have the largest possible variance, and the values of each succeeding coordinates have the largest possible variance under the constraint that they are uncorrelated with the preceding coordinates [18]. Thus, $\mathbf{X}$ is decomposed according to the following formula:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{T}^{\top} \tag{16}$$

where $\mathbf{S} \in \mathbb{R}^{N \times N}$ is a diagonal matrix and $\mathbf{U} \in \mathbb{R}^{N \times N}$ and $\mathbf{T} \in \mathbb{R}^{L \times N}$ are orthonormal matrices.

The alternative basis of dimension H is obtained by selecting the higher H eigenvalues of $\mathbf{S}$ and by replacing the other by zeros. Afterwards, the inverse change of basis is applied to obtain the filtered pulse in the original basis multiplying the modified $\mathbf{S}$ by $\mathbf{T}^{\top}$. All of these operations can be boiled down to a single matrix: the SVD filtering matrix. A feature of this matrix is that when calculated with noiseless pulses, its eigenvectors are the noiseless pulses with unitary height and its eigenvalues their height. A detailed description about this method and its performance can be found in [9].

As explained in Section 2, the components of this basis are transformed according to a new basis with a lower dimension and afterwards transformed back. These two transformations are similar to the one carried out in AEs and DAEs using $\mathbf{W}$ and $\mathbf{V}$. However, unless NNs, in SVD filtering the reduced basis represented by $\mathbf{U}$ and $\mathbf{T}$ are orthonormal and therefore yields different results.

Actually, the main advantage of implementing NNs instead SVD is the less use of memory and the lower computation time during the pulse processing, namely $N \cdot L$ in SVD compared to $2 \cdot L \cdot H$ in NNs (note that in general $H \ll N$). An additional advantage is that a NN can be re-trained with new pulses when necessary.

About the NN types, both DAEs and AEs use separate matrices to perform the two changes of basis whereas RBMs use only one matrix. This fact allows RBMs to have a faster convergence in weights but also it can affect their performance as we will see in the next section.

AEs and RBMs are unsupervised whereas DAEs are supervised. In fact, DAEs transform the input pulse to another. This is particularly useful in DAEs to avoid the modification of the shape of the pulses as a consequence of the degeneration of the detector due to radiation (e.g. in space environment). Hence by using DAEs, the pulse filters can be recalibrated. Similar methods to avoid this issue were discussed in [21–24] for specific shaping types such as trapezoidal and cusp-like.

On the contrary, the use of NNs compared to the use of SVD filtering has the following drawbacks: (a) high computation time during the training; (b) need for the NN to converge. In every test executed by setting the learning rate α to an adequate value the NNs converged properly.

6. Results

In order to validate the performance of the filters based, a set of simulations and tests have been carried out. These filters can be placed after the Analog-To-Digital Conversion (ADC) or after the digital shaping stage. In these tests, the former option was chosen.

6.1. Results with simulated pulses and noise

In this test, a set of 2500 CR–RC pulses with arbitrary heights and white noise were generated as a training set for the NNs. Afterwards, another subset of pulses of the same type were generated for backtesting the trained NNs. Examples of filtering using each method are shown in Figure 4. Clearly, the results are improved with respect to an example that uses a FIR filter with $y[z] = \frac{1}{10}\sum_{k=1}^{10} z^{-k}$ as response function (Figure 5). As explained in the previous section, SVD filtering has been also applied to the same pulses for comparison purposes.

In figure 4 we can observe in all cases that using a low value of H we obtain the best results. It is also observed that the DAE outperforms the AE and RBM which is to be expected since the DAE is trained with $\boldsymbol{x}^*$ instead $\boldsymbol{x}$.

To see in more detail the deviation of the output signal with respect to the ideal one we considered Pulse Height Analysis (PHA) and Pulse Shape Analysis (PSA) because they are two of the most common ways of extraction of information from pulses. Then, given an ideal pulse $\boldsymbol{x}^*$, the error Δ_H is defined for PHA as

$$\Delta_H = |\max(\boldsymbol{x}^*) - \max(\boldsymbol{x})| \tag{17}$$

Whereas for PSA, the error is defined as

$$\Delta_S = \frac{1}{L}\sum_{k=0}^{L} |\boldsymbol{x}^* - \boldsymbol{x}| \tag{18}$$

The error magnitude vs. H for PHA and PSA is depicted in Figure 6. In this figure, we verified that the lower the value of H, the better the noise is filtered. This fact fits into the premise exposed in section 2 that the first dimension of the pulses is their height. Only when the shape of the training pulses becomes very different from each other the unfiltered noise stops being proportional to H as will be shown in the next section.

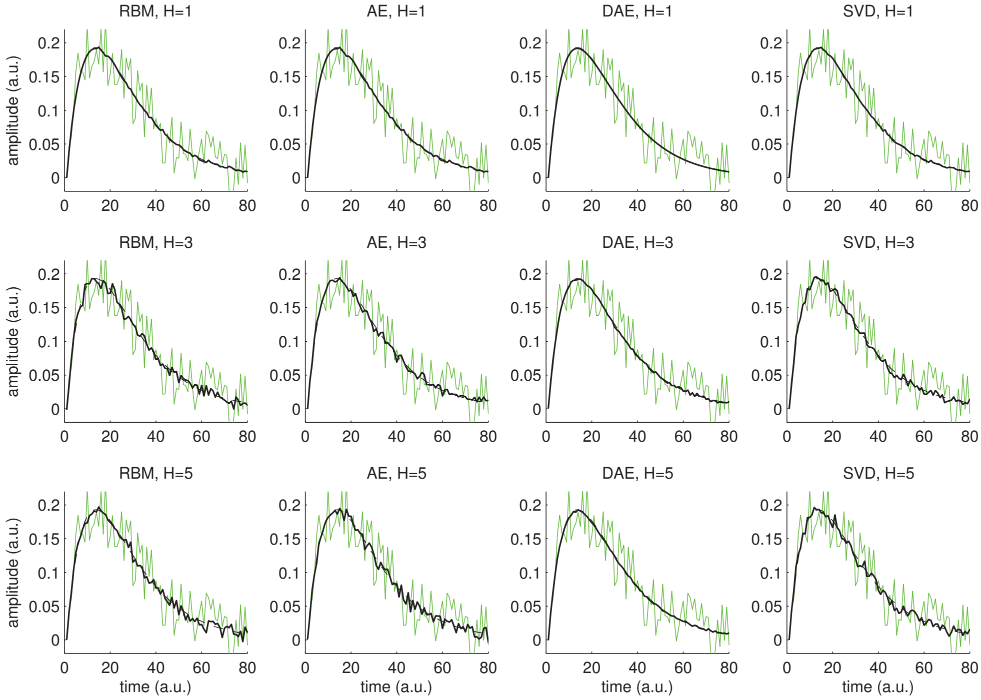

Figure 4: Filtering pulse comparison. The green line is the input pulse $\boldsymbol{x}$, the black dashed line represents the objective output $\boldsymbol{x}^*$ and the black solid line is the obtained output $\boldsymbol{x}$.

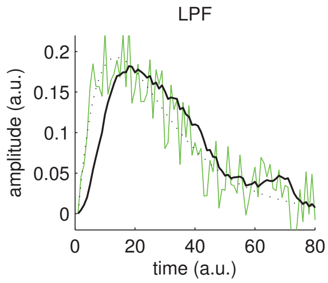

Figure 5: Low-pass filtering.

The lowering of the cost function $\boldsymbol{j}$ as the epochs go on for the three NNs is shown in Figure 7. Along this work the learning rate $\alpha = 0.1$. We can observe that RBM is fastest in its convergence followed by AE and DAE. The results also indicate that H does not affect the speed of convergence too much. Obviously, the more training pulses are used the lower $\boldsymbol{j}$ are achieved.

About the shape of the incoming pulses (e.g. replace in training and tests CR–RC by a triangular shape), no changes in the efficiencies of the pulses are observed although Δ_H and Δ_S oscillate slightly depending on the shape of the pulse. In other words, DAE continues to be the best, followed by AE and RBM, while SVD

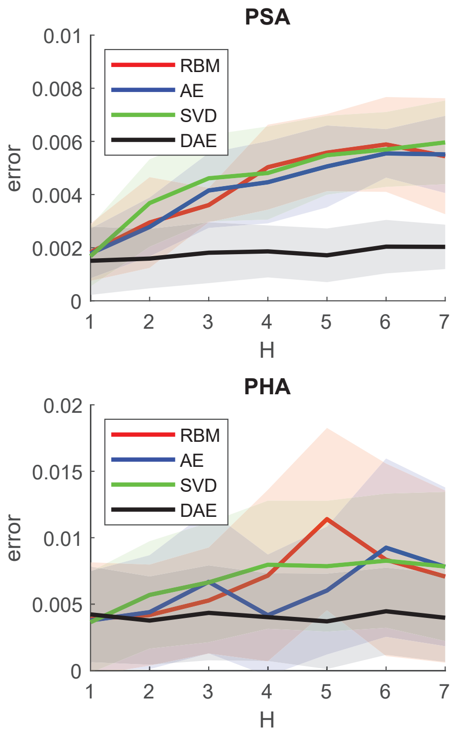

Figure 6: Δ_S and Δ_H with their corresponding standard deviation vs. H for PHA and PSA.

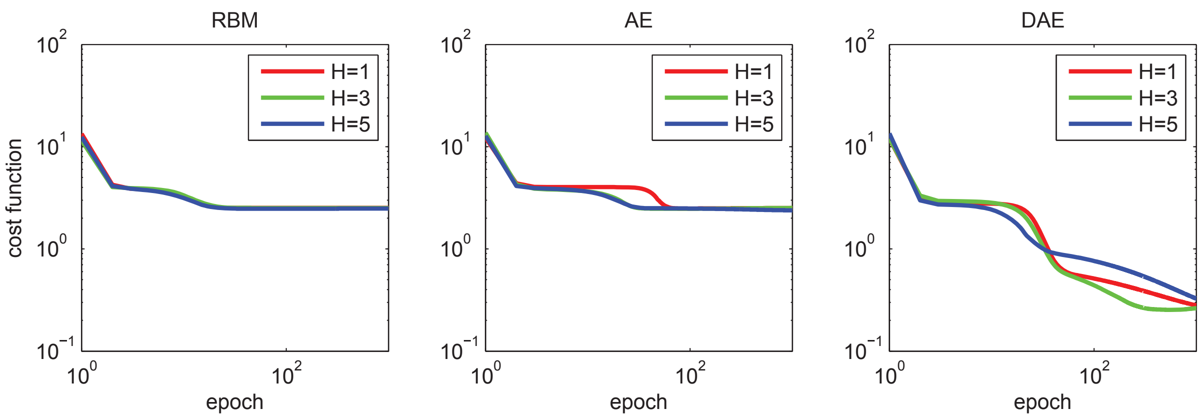

Figure 7: Learning rate for differents values of H. The cost function stands for the quadrature sum of each component of $\boldsymbol{j}$.

filtering offers slightly worse results. What is still observed is that according to the general rules [25], the effect of white noise is inversely proportional to shaping time τ_s and Brownian noise is directly proportional to it.

6.2. Noise filtering analysis

In this section, we will analize the noise impact on the NNs used in this article. Such impact is proportional to the Equivalent Noise Charge (ENC) Q, which is the quadrature sum of the spectral density of each noise type multiplied by its corresponding system's response to that type of noise [26].

When calculating the impact of the two fundamental types of noise that exist in all low-noise pulse amplifiers, voltage (white) noise and current (brownian) noise [25], and normalizing the detector capacitance (i.e. $C = 1$ F), we get

$$Q^2 = i_n^2 F_i \tau_s + v_n^2 F_v \tau_s^{-1} \tag{19}$$

where i_n^2 is the current noise spectral density measured in $\mathrm{A}/\sqrt{\mathrm{Hz}}$ and v_n^2 is the voltage spectral density in $\mathrm{V}/\sqrt{\mathrm{Hz}}$, $\tau_s = T_s L$ (T_s the sampling period) holds for the duration of the pulse, F_i and F_v are the squared system's response (also called noise index) $h(t)$ for a unit of current and voltage noise, respectively. For Linear Time-Invariant (LTI) systems, a unit of voltage noise is a Dirac delta function $\delta(t)$ and a unit of current noise is a step function $u(t)$ [25]. Thereby, taking into account that $h(t) * \delta(n) = h(t)$, the system's response to noise are equal to

$$F_i = \frac{1}{S^2} \int_0^{\infty} (h(t) * u(t))^2 dt \tag{20}$$

$$F_v = \frac{1}{S^2} \int_0^{\infty} (h(t))^2 dt \tag{21}$$

where S is the signal amplitude. Along this paper, $S = 1$ because it is assumed that pulses will be filtered, not amplified.

Noise spectral densities (i_n and v_n) are specific of each particle detector type and can be estimated using thermal and shot noise models among others [27]. However, any accurate calculation of noise spectral densities demands a detailed knowledge of the physical processes involved in the circuit elements. In contrast, F_i and F_v are completely determined by the shaper or filter response [25] and for this reason we focus on them along this Section.

In addition, system's response for generalized noise types can also calculated using fractional calculus [28, 29]. This technique allows to obtain additional noise responses such as $1/f$-noise [30]. One way to obtain the fractional derivative or integral of the Dirac delta function $\delta(t)$ centered at t_0 is obtained via the Riemann-Liouville definition [28, p. 106]. Thus, we define ν_β as

$$\nu_\beta \equiv D^\beta \delta(t - t_0) = \begin{cases} \frac{(t-t_0)^{-\beta-1}}{\Gamma(-\beta)} & \text{if } t \geq t_0 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where $\Gamma(\cdot)$ is the Gamma function and $D^\beta \equiv \frac{d^\beta}{dt^\beta}$ is the differential operator. In this way, for instance when $\beta = 1$ we are working out the first derivative and when $\beta = -1$ the first integral.

Thus, Eq. (20, 21) can be generalized to

$$F_\beta = \frac{1}{S^2} \int_0^\infty (h(t) * \nu_\beta(t))^2 dt \tag{23}$$

Thus, for example, for $1/f$-noise, the filter response will be equal to $F(0.5)$. Eq. (20, 21, 22) are applicable to both analog and digital filters. Figure 8 shows some values of the discretized ν_β depending on the β applied. Note that noise types with $\beta < -1$ generate infinite $F(\beta)$ unless bipolar pulse shapers are included [30].

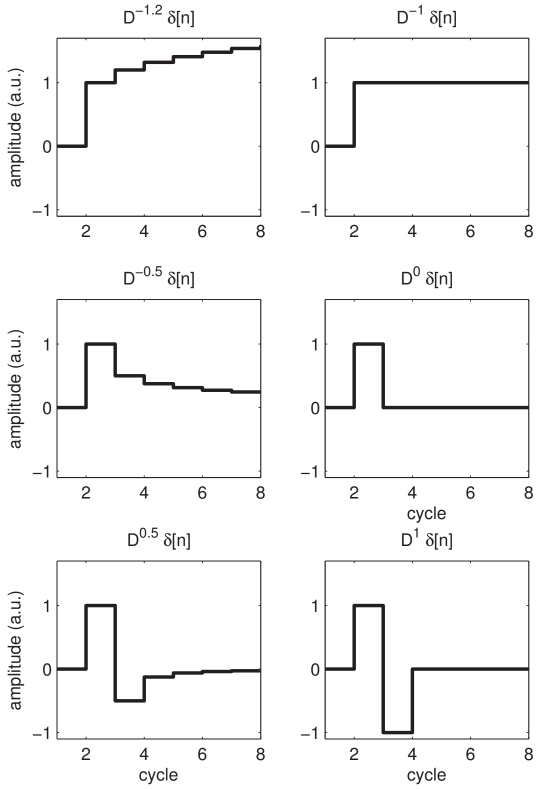

Figure 8: Some differintegrals of the discrete Dirac function. $\nu_0 = D^0\delta[n]$ is the unit of voltage noise (discrete delta function), $\nu_{-0.5} = D^{-0.5}\delta[n]$ is the unit of $1/f$ noise, and $\nu_{-1} = D^{-1}\delta[n]$, that is its integral, is the unit of current noise.

However, noise indexes of shapers and filters based on isolation of pulses (namely NNs) are not comparable among them. This is because we distinguish between shapers, which change the shape of the pulse with the aim of reducing noise, and filters, which reduce noise leaving the signal unaltered. For the latter, we define the ratio of a given noise type as the quotient between the effect of the filtered noise and the unfiltered noise.

$$N_\beta = \sum_{n_0=0}^{L} \frac{\sum_L \left(y(\nu_\beta[n-n_0])\right)^2}{\sum_L \left(\nu_\beta[n-n_0]\right)^2} \tag{24}$$

where $y(\nu_\beta)$ is the output of any NN explained in Sections 3 and 4 as a function of the input ν_β and $\sum_L \left(y(\nu_\beta[n-n_0])\right)^2$ the sum of their squared components.

In this way, given a noise type (for instance $\beta = 0$, that is white noise), if is unaltered after the filtering stage (because, for instance, the output of the NN is always equal to its input), $N_\beta = 1$. When the filter increases the noise level, $N_\beta > 1$ and when the filters reduces it, $N_\beta < 1$. Obviously, this is the only case in which NNs are useful.

Thus, if we set $\beta = 0$ (white noise), Eq. (24) yields

$$N_0 = \sum_{n_0=0}^{L} \frac{\sum_L \left(y(\delta[n-n_0])\right)^2}{L} \tag{25}$$

Making tests using Eq. (24) we get that these noise ratios are proportional to $L^{-\beta}$ when L tends to infinity.

In Figure 9, noise ratio vs. noise type for RBMs, DAEs, AEs are depicted for different values of H. For comparison purposes the results obtained with SVD filtering are also shown. Note that for noise types with $\beta < 0$, the noise ratio is increased with L, which in practice these noise types are the ones that affect the filters the most.

In all four cases, the noise that most affects them, like most filters, is noise with $\beta = -1$ (that is red noise). Besides, in all four cases, the lower the value of H, the greater the noise immunity the filter has. The lines of the NNs are more irregular than those of the SVD because they depend on the level of convergence during the training. Finally, we can see that the best results (almost two orders of magnitude with respect to SVD filtering) are those obtained with RBM. A better result is also observed in the upper limits of DAE than in those of AE because the former train is carried out with noise-free pulses. Finally, the most common noise in particle detectors has a continuous range of $-1 < \beta < -0.25$ [31, 32], therefore, these NNs are suitable to filter this type of noise.

6.3. Effect of the type of noise during the training

Figure 10 shows the error Δ_S for PSA when the NNs and the SVD filter are trained and tested with different types of noise (brownian, pink and white). We can observe that, as H increases, the error increases, in accordance with what was observed in Figure 6. In addition, as the noise becomes brownian (i.e. β decreases) in both learning and tests, a higher error is obtained but it can be mitigated lowering tau_s according to (20). Furthermore, as explained at the beginning of this section, a filter for brownian noise can be placed before the NN. Finally, DAE is the one that offers better results against any noise type because

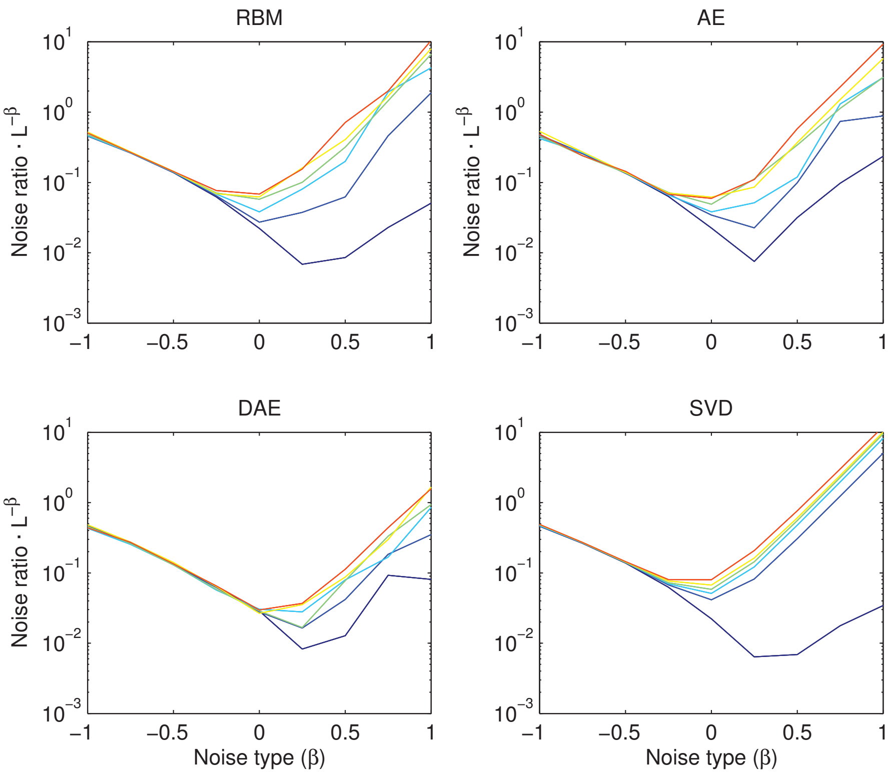

Figure 9: Noise ratio vs. β for RBM, DAE, AE and SVD.

it has the advantage that it is trained with pulses without noise, which also makes it have a more constant error. These conclusions are confirmed when we perform a PHA (Figure 11).

6.4. Results with pulses from a neutron monitor

Finally, a test in a real environment to check the proposed filtering has been performed. The main objective of this test has been to obtain similar results to those obtained in the experiments done with SVD filtering [9].

This test was performed in the Castilla-La Mancha Neutron Monitor (CaLMa) located in Guadalajara, Spain. This instrument consists of 15 proportional gas counter tubes. More information about features, setup and results of this instrument can be found in [33]. In both the cited experiment and the present test, an LND206 tube connected to a Canberra ACHNA98 preamplifier was used.

The raw data fed out from the preamplifier was digitized using a Data Acquisition system (DAQ) working at $L/\tau_s = 50$ Msamples/s and storing it in a PC. Pulses were stored in a text file that can be used multiple times without recapturing new data. In addition, it ensures that possible changes in the obtained results during the test are exclusively due to digital pulse processing. The total raw data length was of 45000 pulses

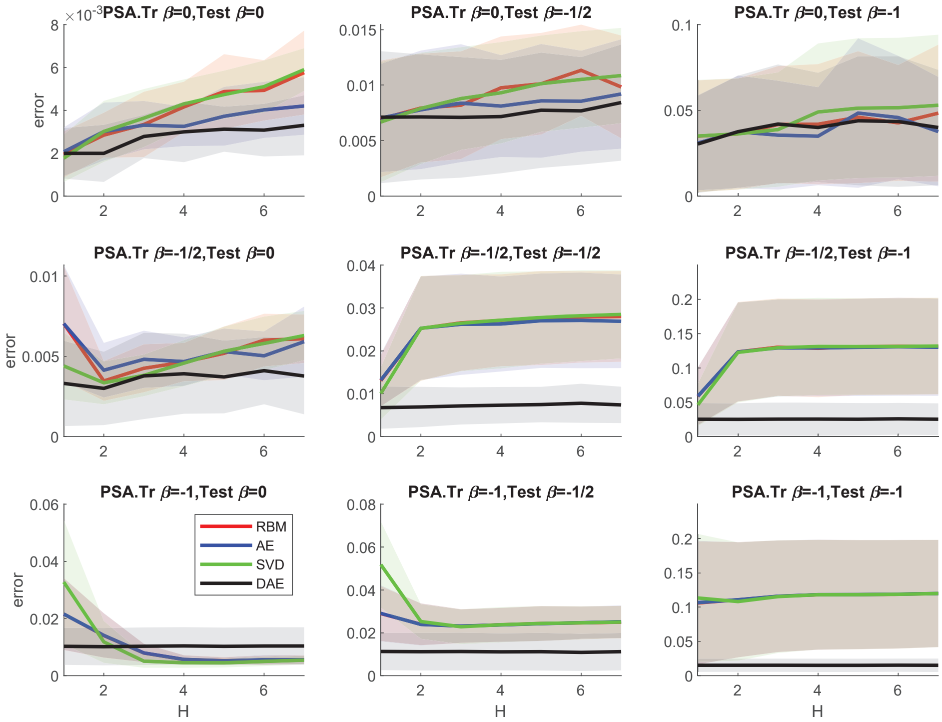

Figure 10: Mean and typical deviation of Δ_S (PSA) vs. training noise type. A set of 1000 samples were used to calculate them.

$\times$ 1002 samples per pulse (i.e. $L = 1002$) captured during over 5 hours. To separate the input pulses, a trigger threshold of 1 V without any previous digital filtering was used.

To carry out a supervised learning with DAE, a pulse without noise obtained from simulations has been used during the training stage. Both in supervised and unsupervised learning, the number of epochs to train the NNs were 120000 with a learning rate $\alpha = 0.1$. These figures were selected taking into account that the more noise the training signal has, the more pulses and epochs are required to reduce the noise. It also was taken into account that preamplifier yields pulses of different shapes (Figure 12) making the training process harder. However, in all the tests the NNs converged.

Figure 13 shows a single pulse filtered with AE, SVD and DAE. For comparison purposes with different H values, SVD filtering was also depicted. We can observe that the shape of the DAE is adjusted to the pulse shape used during the training stage whereas the shape of $\boldsymbol{y}$ is adjusted in proportion to H. Anyway, the pulse height is conserved independently of the pulse shape.

In Figure 14, the learning rate for the different NNs used and different H values ARE shown. We can observe that RBM and AE learning rates are similar and converge quicker than RBM.

Figure 15 depicts a series of histograms of the height of each pulse obtained during the experiment and filtered with NN. The histograms corresponding to SVD filtering were also added for comparison purposes.

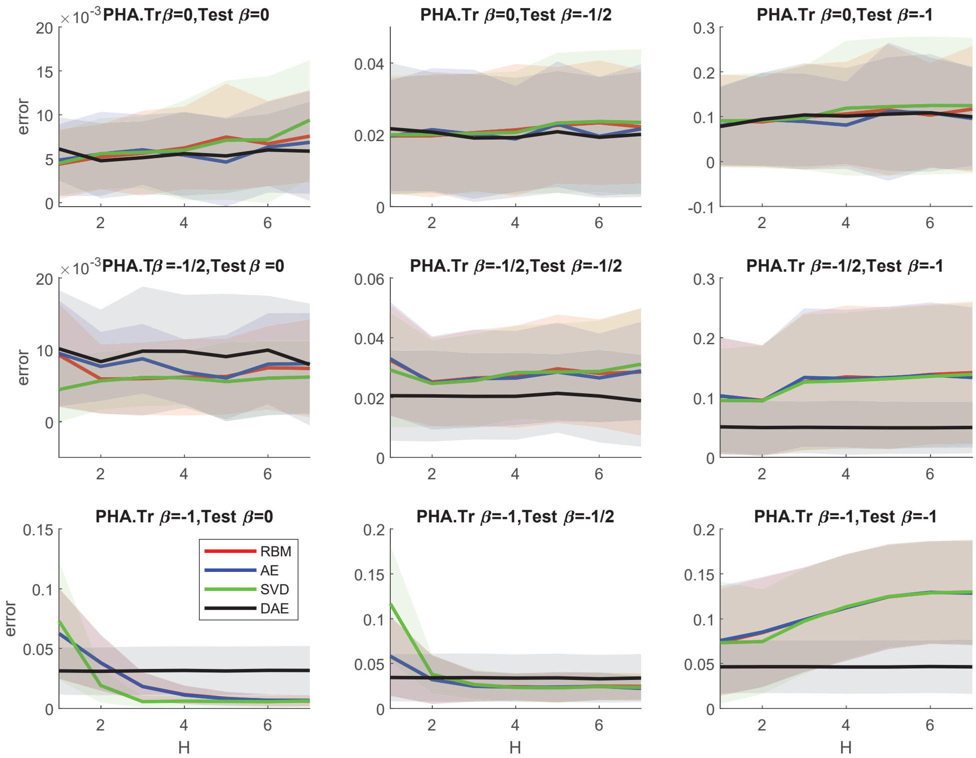

Figure 11: Mean and typical deviation of Δ_H (PHA) vs. training noise type. A set of 1000 samples were used to calculate them.

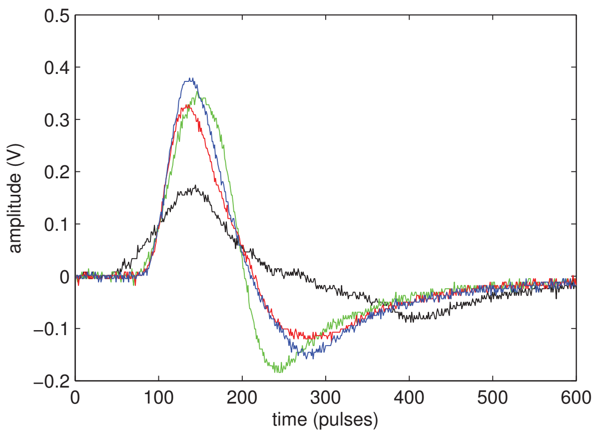

Figure 12: Pulses from the neutron monitor without filtering.

The number of channels used were 100. Since Eq. (18, 17) cannot be used to evaluate the obtained results, because the value of $\boldsymbol{x}^*$ is not known, the filter quality was measured using the Full Width at Half Maximum (FWHM) which is defined as the width of the distribution at a level that is just half the maximum value of the peak divided by the location of the peak maximum [1].

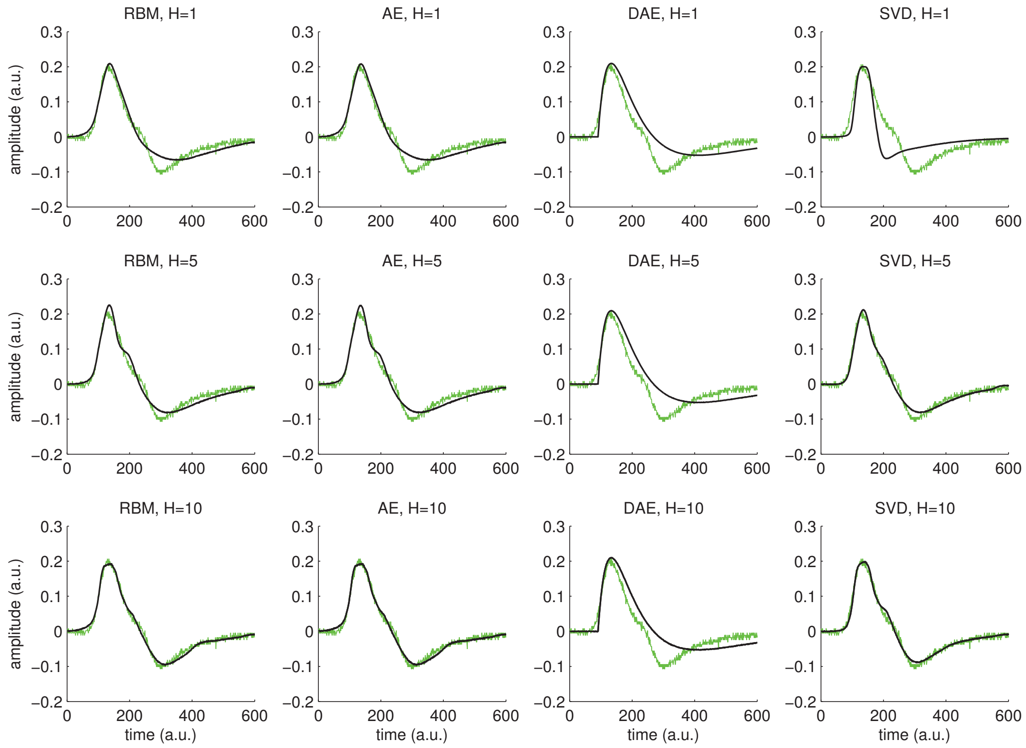

Figure 13: Filtering pulse comparison. The green line is the input pulse $\boldsymbol{x}$ and the black solid line is the obtained output $\boldsymbol{y}$.

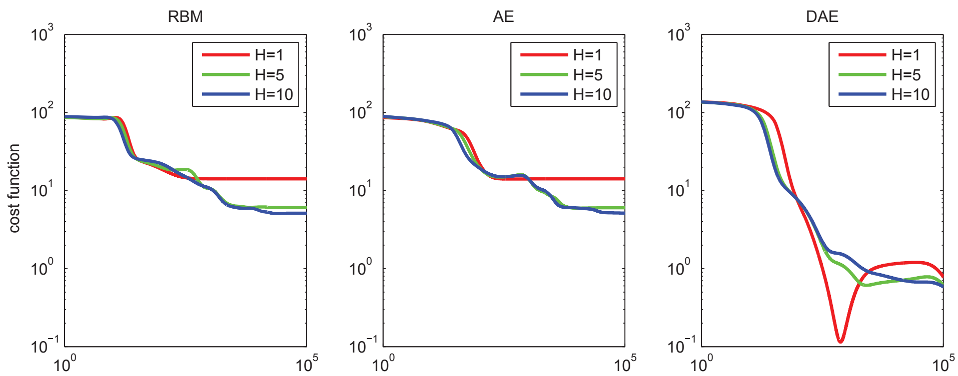

Figure 14: Cost function vs. epoch for the RBM, AE and DAE used in the neutron monitor with different H.

In all of histograms presented, the maximum was in channel 66 or 67 depending of the NN and H. The best results obtained were with the DAE followed by AE and RBM with high H. On the other hand, due to pulse shape irregularities RBM need a high H to learn all the pulse possible shapes and get a low FHWM.

To finalize, we would like to point out from observations that these filtering methods do not handle

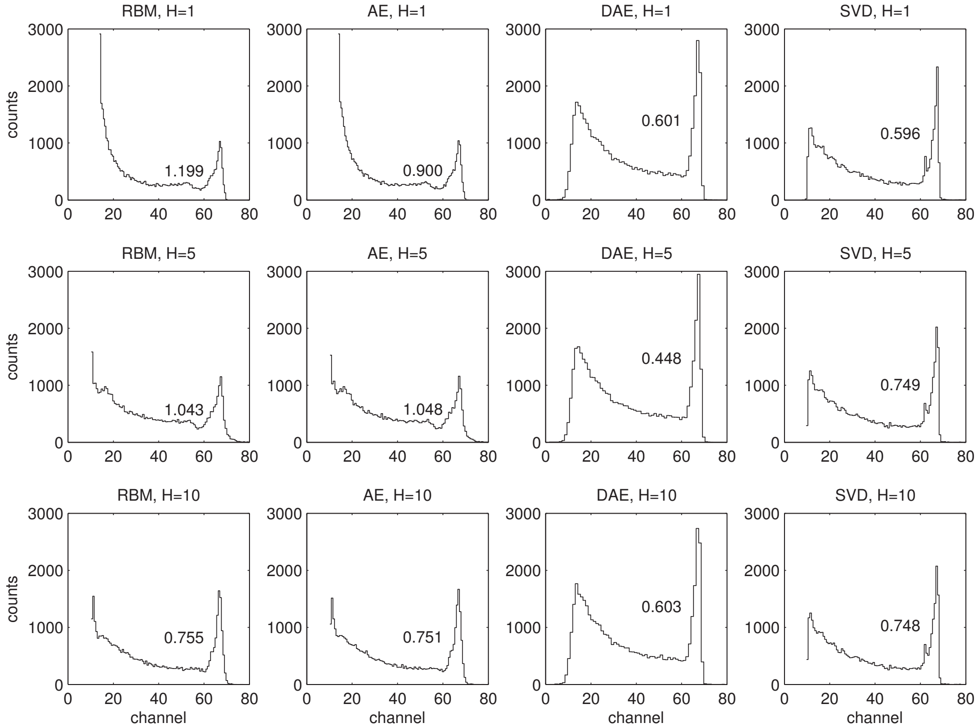

Figure 15: Histograms obtained using different filtering methods and different H with their corresponding FWHM.

properly with pile-up pulses. This fact also happened in previous works such as [8] and [9].

7. Conclusions

A novel filtering technique of isolated pulses based on NNs (namely RBMs, AEs and DAEs) has been presented in this article. These NNs have been compared with SVD in simulation and test with pulses coming from a neutron detector to evaluate its performance. In the same way that SVD, NNs have similar responses to noise and are less susceptible to white noise, but more to brownian noise than common digital filters such as FIR and IIR. In any case, the noise impact continues being dependent on the length of the pulse in the same way that common filters. However, despite the computation time for the training, AEs and DAEs improve significantly the results of SVD filtering. During the training stage, in all the tests the NNs converged without problems. Among these NNs, the best results were obtained using the DAE, despite pulses without noise are mandatory to train it. These methods based on linear NNs overcome the results obtained with common digital filters and are relatively easy to implement in software and therefore suitable for analysis of pulses coming from particle detectors.