Online Track Vertex Reconstruction Method Based on an Artificial Neural Network for MPGD

Hao Liu and Yi Zhang *

School of Nuclear Science and Technology, Lanzhou University, 199 Donggang West Rd., Lanzhou 730000, China; hliu2018@lzu.edu.cn
* Correspondence: yizhang@lzu.edu.cn

Received: 24 July 2020; Accepted: 24 September 2020; Published: 27 September 2020

Abstract: In this paper, an online vertex reconstruction algorithm based on an artificial neural network (ANN) was proposed for the micro-pattern gaseous detector (MPGD). A simulation based on Geant4 was performed to generate the training and testing samples for the two cascade neural networks. Compared with a center-of-mass reconstruction, the proposed method shows better precision and much higher efficiency. Furthermore, a scheme for implementing the proposed algorithm on a Field-Programmable Gate Array (FPGA) chip is also presented to demonstrate that the algorithm could be integrated in a modern data acquisition (DAQ) system for online imaging techniques.

Keywords: MPGD; ANN; track vertex reconstruction; FPGA

1. Introduction

The micro-pattern gaseous detector (MPGD) was originally developed for high-energy physics experiments [1], and has several advantages such as high spatial resolution, fast response time, and a large sensitive area. Since then, the development of large scientific devices in high-energy physics has increased the performance requirements of the particle detectors they are paired with. The structure of the MPGD has been continuously optimized to obtain a higher spatial resolution and counting rate. Micromegas and gas electrons multiplier (GEM) detectors appeared and were equipped with larger and more complex readout devices [2–5]. This results in a large amount of data from one single event, which means that data acquisition and analysis of the detector is challenging.

In MPGD-based experiments, the pulsed signal is often sampled by an analog-to-digital converter (ADC) with a high frequency such as 40 MHz, and then uploaded to a computer for analysis. There are often bottlenecks in this process due to the huge amount of data transfer and storage, which is caused by the pulse shape sampling of the signal, especially in the large-area and high-precision case. The large number of readout electrodes lends itself well to parallel computation, whereas the traditional track vertex reconstruction algorithms run by personal computers (PCs) are mostly serial. The consumption of computational resources can be large. The signal generated by the incident particle does not always have a linear correlation with the position of the incident particle [6], especially when the incidental angle is large. Furthermore, this relationship is also affected by the detector structure and readout device, which makes it more difficult to use an analytical function to describe these relationships. In this work, our aim was to develop a new scheme of tracking vertex reconstruction to overcome the disadvantages mentioned above and to implement an online analysis with the precision to obtain a position resolution of 0.35 mm. An artificial neural network (ANN) can be used as a mature algorithm to solve difficult problems in the field of nuclear detection and other challenging projects [7]. Cenninia et al. used this algorithm to successfully extract physical information from the signal of the liquid argon time projection chamber [8]. Delaquis et al. used deep neural networks to reconstruct energy and position information for EXO-200 [9]. We used this algorithm for MPGD track vertex reconstruction method.
reconstruction and implemented the algorithm on a Field-Programmable Gate Array (FPGA) chip to obtain online processing data.

In this paper, we used a Geant4 software toolkit to implement a fast neutron simulation of MPGD [10], to train a reduced-dimensional network and a feedforward neural network (FNN) from the resulting dataset to reconstruct the track vertexes [11], to compare the reconstruction performance of this track vertex reconstruction algorithm with that of the centroid method [12], and then, to demonstrate the feasibility of the algorithm at the FPGA level.

2. Materials and Methods

2.1. Detector Simulation

The structure of the detector in the simulation is shown in Figure 1. The detector contains a cathode plate, a 0.3 mm-thick polyethylene conversion layer, a 5 mm-deep drift region filled with 75% argon and 25% carbon dioxide, a 100 µm-thick and 300 mm x 300 mm-large metallic mesh with 40% optical transparency, and a readout plate. The electric field in the drift region is 1 kV/cm. The size of the readout plate is 300 mm x 300 mm, it is located 100 µm below the metallic mesh, and it adopts the strip electrode readout method in both the X and Y direction. The anodic strips are arranged in the upper and lower layers of the printed circuit board (PCB) with a period of 0.75 mm, where the width of the anodic strips is 0.5 mm and the gap between the anodic strips in the arrangement is 0.25 mm. The upper and lower layers each contain 400 anodic strips, and the gap in the upper layer is hollowed out. The structure of the readout plate is shown in Figure 2. The range of fast neutron energies used in the simulation is 13.5 MeV to 14.5 MeV. Fast neutrons are injected vertically into polyethylene where (n, p) reactions take place to produce recoil protons, which enter the drift region at an angle and produce primary electrons that drift towards the anode under the action of the electric field in the drift region and diffuse laterally. The final location of the electrons that collect by the readout plate depends on the location of their initial ionization and the degree of lateral diffusion. The primary electrons near the cathode end of the plate drift longer and spread more laterally than the electrons near the metallic mesh end of the plate. The distribution is more decentralized. Figure 3 shows the position of the electron distribution at the anode after the incidence of a particle with a vertex of (32.76 mm, −4.04 mm), and it is clearly shown that the lateral spread of electrons is smaller for shorter drift times.

![Figure 1. Schematic diagram of the detector in the simulation.](image-url)
Figure 2. Diagram of the strip electrode readout anode.

Figure 3. Electron position distribution. The length of 300 mm in the x and y direction of the anode is defined as (−150 mm, 150 mm), and the color indicates the number of electrons.

The time that it takes for the primary electrons to be collected by the readout plate depends on the location of the z-directional generation and the degree of longitudinal diffusion. The drift velocity of an ionized electron is about 40 µm/ns and thus it takes more than 100 ns in a 5 mm thick drift region. The APV-25 chip that we used is a highly integrated chip that incorporates a large number of preamplifiers and analog-to-digital converters, and its operating frequency can be set. For instance, the APV-25 chip works at a frequency of 40 MHz, which leads to a 25-ns FEE period, so the entire track signal is sustained for several periods. Figure 4 shows the signal for one of the sampling periods of an event.
The relationship between the electrode channel number and its representative position is as follows:

\[ \text{xpos} = 0.75 \times (\text{Channel} - 0.5) - 150 \]  \hspace{1cm} (1)

In this formula, xpos represents the position where a certain readout channel is located. The channel is an integer value between 1 and 400. For example, in Figure 4, the channel with the highest signal value is 242, which represents a position of 31.125 mm.

2.2. Traditional Track Vertex Reconstruction Methods

The number of electrons in the MPGD is large after avalanche amplification, and because of lateral diffusion, the electron cloud forms a symmetric distribution with the position of the primary electrons as the center of gravity.

The centroid method weights the signal amplitude of each channel to find the particle incidence position, which can be calculated by the formula:

\[ G(Q, x) = \frac{\sum_{i=1}^{n} (Q_i - Q') x_i}{\sum_{i=1}^{n} (Q_i - Q')} \]  \hspace{1cm} (2)

\( G(Q, x) \) is the position determined by the gravity center method, \( Q \) is the charge value on the activated channel and \( x \) is the serial number of the activated channel. \( x_i \) is the position represented by the readout bar which is equal to xpos in formula (1), \( Q_i \) represents the \( i \)-th value of the value in \( Q \), \( Q' \) is the background value on the channel, which is equal to the mean of the noise on this channel, which in our simulation was set to 15. \( n \) is the number of channels with response in this direction.

As can be seen from the above formula, the centroid method requires the implementation of a division operation, which is more complex at the FPGA level.

2.3. ANN Algorithm

The ANN algorithm is an abstract mathematical model inspired by the organization and operating mechanisms of the human brain in biology [13]. It is often used as a black-box model. We can use it to find a functional relationship from one vector to another, especially when we are uncertain about the input to output relationship. Each neuron of the ANN algorithm accepts the output of the other
neurons, multiplies it by the assigned weights and sums them, and then uses it for output after passing through a nonlinear activation function. The neural network is trained by iteratively learning the data to obtain a network that more closely approximates the real output, using error backpropagation to adjust the weights of the internal neuronal connections.

2.4. ANN Reconstruction Algorithm

With the above simulation, we obtained a large number of events, each lasting 10 to 20 periods. Each period has 400 data points (corresponding to the channels) in the X and Y directions. Since the readout area of the detector is a square and the X and Y directions do not affect each other, we usually process the X and Y directions separately when using other data processing methods. In the approach used in this paper, if the X and Y directions were processed together, we would get a network with structural redundancy, long training time, and poor accuracy. Therefore, we chose to treat the X and Y directions separately. An event that lasts for 20 periods, produces up to 8000 valid data points. If these 8000 data points were used as input to the neural network it would create a very complex structured network, which would significantly increase the training time for the network. Since the data for an event is composed of different periods, we can downscale the data for all periods.

2.4.1. Dimensionality Reduction

The traditional method of data dimensionality reduction in machine learning is principal component analysis (PCA) [14], which uses linear transformation to expresses the data in a low-dimensional space. In 2006, Hinton proposed the use of autoencoder to achieve dimensionality reduction of high dimensional data [15]. He trained a neural network with the same output data as the input data, and the output of the hidden layer of the network enabled the dimensionality reduction of the original data. This method is more efficient and flexible than PCA in terms of dimensionality reduction. The sparse autoencoder adds a penalty factor to the autoencoder mean-variance loss function $J(W,b)$, which suppresses a portion of the nodes in the hidden layer and makes the coding sparser, thus increasing the dimensionality reduction capability of the autoencoder.

The sparse autoencoder is an unsupervised learning system, which is shown in Figure 5. Each neuron in the hidden layer has weight parameters for all neurons in the previous layer. The activation function is a sigmoid function with the following expression:

$$S(x) = \frac{1}{1 + e^{-x}}$$

(3)

The loss function is:

$$J_{\text{sparse}}(W,b) = J(W,b) + \sum_{j=1}^{z_h} KL(\rho || \hat{\rho}_j)$$

(4)

and

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} [a_j^{(2)}(x^{(i)})]$$

(5)

$$KL(\rho || \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}$$

(6)

$$J(W,b) = \frac{1}{m} \sum_{j=1}^{m} (y_j(W,b) - y'_j)^2$$

(7)

where $W$ represents the matrix of weight parameters of the neurons in the network and $b$ represents the bias matrix, and $m$ is the number of training samples. For example, $W^{(0)}_{ij}$ denotes the weight associated with the connection between unit $j$ in layer 1 and unit $i$ in layer $l+1$. Also, $b^{(l)}_i$ is the bias associated with unit $i$ in layer $l+1$. $a_j^{(2)}(x^{(i)})$ is the output of the response of the $j$-th neuron in the hidden
layer to the $i$-th sample. $J(W,b)$ is the loss function of the autoencoder, $\hat{\rho}_j$ denotes the average activity of the hidden layer neurons, and $\rho$ is the sparsity parameter. $KL(\rho || \hat{\rho}_j)$ is the Kullback–Leibler (KL) divergence between a Bernoulli random variable with mean $\rho$ and a Bernoulli random variable with mean $\hat{\rho}_j$. KL-divergence is a standard function for measuring how different two distributions are. $y_j$ denotes the true output value of the $j$-th sample. $y'_j$ denotes the output value of the $j$-th sample’s network computations. Formulas (4)–(7) are explained in more detail in [16].

Figure 5. Sparse autoencoder.

We use 30,000 periods of data to train the sparse autoencoder. Before proceeding to train the sparse autoencoder, the network input dataset needs to be normalized to $(0.1, 0.9)$. L-BLGS was used for training to optimize the loss function [17]. The learning rate of the training was 0.0001 and the maximum number of iterations was 1000. The structure of the network is 400-40-400 and it has a data compression ratio of 10:1. The parameters of the nodes were initialized randomly at the beginning of training. The value of the loss function was calculated at each iteration using all the data, changing the node parameters according to the optimization algorithm. After a number of iterations, the value of the loss function stops decreasing and converges, which indicates that the performance of our trained sparse autoencoder is excellent. After training, the output of the hidden layer is the dimensionality reduction data we obtained. The network was trained for 973 periods before stopping the iteration, and the value of the loss function changed during the training process as shown in Figure 6.

The loss function was eventually reduced to near 0.699 and converges, which indicates that the network can achieve good dimensionality reduction. There is no zero suppression in our algorithm. Traditional zero suppression reduces the amount of data by subtracting readout poles with lower response, which can be seen as a dimensionality reduction tool. Our sparse autoencoder is a more powerful dimensionality reduction tool because it is able to take useful information directly from 400 dimensions of data to take the next step.
2.4.2. Feedforward Neural Network (FNN)

Universal approximation theorem shows that feedforward neural networks (FNN) can fit functions of arbitrary complexity with arbitrary accuracy by having only a single hidden layer and a certain number of neurons [18]. Since there is a non-linear functional relationship between the signal generated by an incident particle on the readout plate and its incident position under a particular operating condition, we can use FNN to complete the reconstruction of the incident position. A multi-hidden layer deep neural network has higher training efficiency, better fitting effect, and stronger generalization ability compared to a single hidden layer network while consuming the same network nodes. We used data from 27,000 events to train the network. The sample set was divided into training, validation, and test sets, with a sample size of 18,900 for the training set and a sample size of 4050 for the validation set and test set and the data used lasts 10–20 periods. Instead of using the raw data directly during training, the raw data is passed through the sparse autoencoder shown above and then the output of the hidden layer of the sparse autoencoder is used as input to the FNN. When an event lasts for less than 20 periods, 0 is used instead of the result of the empty period dimensionality reduction. The output of the FNN is the incident position of the particle in the operating condition. We can use FNN to complete the reconstruction of the incident position.

The structure of the FNN is 800-30-20-3-1, and the number of layers and nodes of the hidden layer is constantly adjusted according to the training results. The activation functions used by FNN are tanh, tanh, and purlin, and its structure is shown in Figure 7.

The expression for the activation function is as follows:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$  \hspace{1cm} (8)

$$\text{purlin}(x) = x$$  \hspace{1cm} (9)

The network stops iterating after 869 iterations, which is a special case. The total number of iterations varies depending on the value of the initial parameter. The loss function converges. The convergence criterion is used for the loss function (as shown in Figure 8). The change in the mean squared error of the training set, validation set, and test set during the training process is shown in Figure 8. The mean squared error of the test set ensured that the training did not fall into overfitting. The training results of the network in the training, validation and test sets are shown in Table 1.
Figure 7. Structure of the feedforward neural network (FNN).

The expression for the activation function is as follows:

\[ \tanh(x) = \frac{1}{1 + e^{-2x}} \] (8)

\[ purlin(x) = x \] (9)

The network stops iterating after 869 iterations, which is a special case. The total number of iterations varies depending on the value of the initial parameter. The loss function converges. The convergence criterion is used for the loss function (as shown in Figure 8). The change in the mean squared error of the training set, validation set, and test set during the training process is shown in Figure 8. The mean squared error of the test set ensured that the training did not fall into overfitting. The training results of the network in the training, validation and test sets are shown in Table 1.

**Table 1.** FNN’s training results.

| Data Set   | Samples | MSE  | R   |
|------------|---------|------|-----|
| Training   | 18,900  | 0.093| 0.998|
| Validation | 4050    | 0.16 | 0.997|
| Testing    | 4050    | 0.12 | 0.998|

MSE (mean squared error): The average squared difference between outputs and targets. R (Regression): Measure the correlation between outputs and target. Note: the performance of the network can continue to be improved by adjusting the network structure and improving training methods.
3. Comparison of Results

We used the traditional centroid method and FNN to process the same track data. The results of the two reconstructions were compared with the original incidence locations, as shown in Figures 9a and 10a. The resolution of the two methods for reconstructing the track vertexes was analyzed using statistical methods, as shown in Figures 9b and 10b. The blue histograms are the statistical distributions of $P_{\text{cen}} - P_{\text{trk}}$ and $P_{\text{ANN}} - P_{\text{trk}}$, and the red lines were fitted using a Gaussian distribution of the result.

![Figure 9](image)

**Figure 9.** (a) Location distribution of the results of the centroid method reconstruction (the x direction is the position of the particle incidence in the simulation, the y direction is the position of the centroid method reconstruction); (b) The error distribution of the centroid method reconstruction results ($P_{\text{cen}}$ is the position of the centroid method reconstruction, $P_{\text{trk}}$ is the position of the proton incidence in the simulation).
Figure 10. (a) Artificial neural network (ANN) reconstruction position distribution (the x direction is the position of the particle incidence in the simulation, the y direction is the position of ANN reconstruction); (b) the ANN reconstruction result error distribution ($P_{\text{ANN}}$ is the position of the ANN reconstruction, $P_\text{trk}$ is the position of the proton incidence in the simulation).

Table 2 shows the results of the statistical analysis of the two reconstruction methods.

| Reconstruction Method | Sigma (mm) | Mean (mm) |
|-----------------------|------------|-----------|
| Centroid method       | 2.60       | 0.013     |
| ANN method            | 0.35       | 0.0015    |
The above results show that the incidence position of the particle can be reconstructed more accurately using the ANN algorithm. The sigma value in the error fit of the ANN reconstructed position is lower than the sigma of the centroid method, indicating that we can obtain better position resolution with the ANN algorithm. The centroid method gives biased results for some events because it takes the center of gravity of the track as the incident position of the particle. When a charged particle is not perpendicular to the detector drift region, the particle generates a longer track, which induces a charge on multiple readout strips. The center of gravity of the track will be off the incident position of the particle, resulting in a larger error in the result. Using an ANN to reconstruct the particle incident position improves accuracy, and the ANN can be implemented at the FPGA level.

4. Online Track Vertex Reconstruction

Typically, the use of modern GPUs instead of CPUs can increase the training efficiency of neural networks, which is not a complicated operation. Using FPGAs to accelerate training is a very complex project that we have not researched. This paper focuses primarily on the operation of a well-trained network. In this regard, FPGAs have great advantages over GPUs and CPUs, including fast operation, repeatable programming, and portability. Running a trained ANN algorithm model on a computer consumes a lot of time. We propose a method to speed up the data analysis using FPGA to implement ANN reconstruction algorithms. Since the APV-25 chip used in the MGPD data acquisition system is mounted on the FPGA host chip, the FPGA implements the data acquisition and uploading. We designed a solution to implement ANN on the FPGA chip and combine it with the data acquisition part to complete the whole process of data acquisition, data processing, and data uploading using FPGA to achieve rapid data acquisition and data analysis.

Figures 11 and 12 show the scheme that was used for implementing the ANN algorithm on FPGA. A lookup table was designed using the ROM in the FPGA to realize the nonlinear functions in the ANN. The method is used to select a domain of the function such as \((-6, 6)\), then 3000 points in equal steps are chosen within the range, and then the computer calculates the function values at all fetch points, and writes all function values to ROM. The X value of all points is a linear function of the address of the ROM and the function value is produced after the address is found according to the input X value. When the address is a non-integer, the integer is taken down and the function value is produced. A multiply-accumulator was designed to realize the vector dot product operation in ANN. A hidden layer node of the neural network will consume a multiplier. The Arria II GX EP2AGX125EF35 chip was used in the design, and Table 3 shows the hardware resources that were consumed by the project after synthesis.

| Table 3. Consumption of engineering resources. |
|-----------------------------------------------|
| Element             | Value      |
|---------------------|------------|
| Combinational ALUTs | 36,686     |
| Dedicated logic registers | 23,565   |
| Block memory bits   | 1,674,540  |
| Multiplier 18 bits  | 96         |
Running ANN algorithms on the computer side uses floating-point operations, whereas realizing floating-point operations on the FPGA consumes a huge amount of resources. We converted floating-point operations to fixed-point operations, which converts all input values and parameters in the ANN to the specified fixed-point format. Since the ADC can output a 12-bit complement, the desired number of fixed points can be obtained by complement and shift operations. The fixdt (1.18.14) mode was selected after constant adjustment of the fixed-point number pattern, and taking into account hardware cost and calculation accuracy. This represents numbers that will be represented using 18 bits, 1 symbolic bit, 3 integer bits, and 14 decimal bits. We performed RTL simulations on a computer for an FPGA project built using the fixed-point model. The results show that the error range between the fixed-point model and the floating-point model is (0.1 mm, 0.2 mm), which would increase the position resolution by approximately 0.2 mm, thus indicating that the ANN algorithm model works well on the FPGA. This effect can be ameliorated by increasing the number of fixed-point bits by using a more powerful FPGA.
Compared to CPU, using FPGA to implement ANN results in a significant increase in speed, which is reflected in the parallelism within the algorithm. For matrix calculations in ANN, the CPU usually completes one unit calculation before proceeding to the next unit and waits for the program to finish before proceeding to the next run, which consumes a lot of time even with a high running frequency. Computation using FPGA can be highly parallelized, not only in matrix computing, but also in different layers of the network. In our project, the FPGA ran at 500 MHz and consumed approximately 3000 ns to complete a single calculation, thus processing data at speeds up to $3.3 \times 10^5$ tracks/s in theory, which is difficult to achieve with a CPU. In fact, this approach speeds up the analysis of data not only in terms of the speed of the calculations, but also in regard to the volume of data transferred and the drastic reduction in manpower required. As shown in Equation (2), the centroid method requires a lot of logical judgments and division operations, which are difficult to implement on FPGA and slow to run, which makes it unsuitable as an algorithm for online track vertex reconstruction.

5. Conclusions

In this paper, the MPGD track vertex reconstruction method was investigated and a scheme for online track vertex reconstruction was proposed. The centroid method, a conventional reconstruction algorithm, was first introduced and its reconstruction results were presented. The results show that the centroid method has a larger reconstruction error when the particles are not perpendicularly incident. In this situation, we propose the use of ANN for the track vertex reconstruction. Two networks were trained using a set of simulated generated data to complete the dimensionality reduction and reconstruction of MPGD track data. The trained neural networks have good convergence and strong generalization ability. The reconstruction results of the centroid method and ANN algorithm were compared, and the results show that the neural network model reconstructs the particle incidence position more accurately than the center of gravity method, which results in a better position resolution of 0.35 mm for the detector.

Most traditional methods of track vertex reconstruction are serial algorithms on a computer, which result in a reduction in the speed of data analysis. In this paper, we proposed that the ANN reconstruction algorithm be implemented on FPGA. An FPGA design based on the principle of the ANN algorithm was proposed to improve the data analysis efficiency by using the high operating speed of FPGA. The computer receives only valid data from the analysis of the raw trail data, thus reducing the amount of data storage.

Author Contributions: Conceptualization, H.L., Y.Z.; methodology, H.L.; writing—original draft preparation, H.L.; writing—review and editing, Y.Z.; visualization, H.L.; supervision, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sauli, F.; Sharma, A. Micropattern gaseous detectors. Annu. Rev. Nucl. Part. Sci. 1999, 49, 341–388. [CrossRef]
2. Giomataris, Y.; Rebourgeard, P.; Robert, J.P.; Charpak, G. Micromegas: A high granularity position-sensitive gaseous detector for high particle-flux environments. Nucl. Instrum. Methods A 1996, 376, 29–35. [CrossRef]
3. Sauli, F. Micro-pattern gas detectors. Nucl. Instrum. Methods A 2002, 477, 1–7. [CrossRef]
4. Van Der Graaf, H. GridPix: An integrated readout system for gaseous detectors with a pixel chip as anode. Nucl. Instrum. Methods A 2007, 580, 1023–1026. [CrossRef]
5. Wu, H.; Zhao, S.; Wang, X.; Zhang, X.M.; Qi, H.R.; Zhang, W.; Wu, K.Y.; Hu, B.T.; Zhang, Y. Discriminating cosmic muons and X-rays based on rise time using a GEM detector. Chin. Phys. C 2016, 40, 8. [CrossRef]
6. Flierl, B.; Hertenberger, R.; Bielke, O.; Zeitelhack, K. TPC-like readout for thermal neutron detection using a GEM-detector. Nucl. Instrum. Methods A 2016, 824, 528–531. [CrossRef]
7. Cappuzzello, F.; Agodi, C.; Cavallaro, M.; Carbone, D.; Tudisco, S.; Presti, D.L.; Oliviera, J.R.B.; Finocchiaro, P.; Colonna, M.; Calvo, D.; et al. The NUMEN project: Nuclear Matrix Elements for Neutrinoless double beta decay. *Eur. Phys. J.* 2018, 54, 72. [CrossRef]

8. Cennini, P.; Cittolin, S.; Revol, J.P.; Rubbia, C.; Tian, W.H.; Motto, S.; Picchi, P.; Cavanna, F.; Wang, G.H.; Vingoli, C.; et al. A neural network approach for the TPC signal processing. *Nucl. Instrum. Methods A* 1995, 356, 507–513. [CrossRef]

9. Delaquis, S.; Jewell, M.J.; Ostrovskiy, I.; Weber, M.; Ziegler, T.; Dalmasson, J.; Kaufman, L.J.; Richards, T.; Albert, J.B.; Anton, G.; et al. Deep neural networks for energy and position reconstruction in EXO-200. *J. Instrum.* 2018, 521, 436–444. [CrossRef]

10. Zhang, Y.; Wu, H.; Zhou, J.; Zhao, S.; Hu, B. Track identification and reconstruction in fast neutron detection by MPGD. *Nucl. Instrum. Methods A* 2018, 906, 68–76. [CrossRef]

11. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* 2015, 521, 436–444. [CrossRef] [PubMed]

12. Mankel, R. Patter recognition and event reconstruction in particle physics experiments. *Rep. Prog. Phys.* 2004, 67, 553–622. [CrossRef]

13. Islami rad, S.Z.; Gholipour Peyvandi, R. A novel and fast technique for evaluation of plastic rod scintillators as position sensitive gamma-ray detectors using artificial neural networks. *Radiat. Phys. Chem.* 2019, 157, 1–5. [CrossRef]

14. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley Interdiscip. Rev. Comput. Stat.* 2010, 2, 433–459. [CrossRef]

15. Hinton, G.E.; Salakhutdinov, R. Reducing the Dimensionality of Data with Neural Networks. *Science* 2006, 313, 504–507. [CrossRef] [PubMed]

16. Zhang, C.; Cheng, X.; Liu, J.; He, J. Deep Sparse Autoencoder for Feature Extraction and Diagnosis of Locomotive Adhesion Status. *J. Control. Sci. Eng.* 2018, 2018, 1–9. [CrossRef]

17. Zhu, C.; Byrd, R.H.; Lu, P.; Nocedal, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* 1997, 23, 550–560. [CrossRef]

18. Hornik, K.; Stinchcombe, M.B.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* 1989, 2, 359–366. [CrossRef]

19. Burden, F.R.; Winklei, D.A. Bayesian Regularization of Neural Networks. *J. Mol. Biol.* 2008, 458, 23–42.