BELLE2-NOTE-XX-YYYY-ZZZ
DRAFT Version 1.0
March 30, 2020

Hopfield Network for Cluster PID at the PXD

Irina Heinz[*] and Soeren Lange[†]

Justus Liebig University, Giessen, Germany

Abstract

Due to data measurement, recognizing a real signal out of a large number of background data arises as a problem at the PXD of Belle II experiment. Thus, the primary objective is finding an analysis method to separate highly ionizing particles, such as anti-deuterons and magnetic monopoles, from e.g. beam background.
Since neural networks showed up as a useful tool for image and data recognition, several types of neural networks for signal recognition of PXD data were developed. A new approach was considered by taking raw pixel data as input for a Hopfield network, which is a special network structure usually used for image recognition. Different optimization variants, such as a detailed analysis of data, improved network performance.

[*]Electronic address: irina.heinz@desy.de
[†]Electronic address: Soeren.Lange@exp2.physik.uni-giessen.de

Contents

1. INTRODUCTION

The Belle II experiment, located at high-energy research facility KEK in Tsukuba (Ibaraki) Japan, started 2018. The asymmetrical electron positron collider SuperKEKB was designed with a luminosity increase of 40 times compared to the predecessor experiment Belle and focuses on precise measurements of rare processes. Around the interaction region several detectors are installed, including the Pixel Detector (PXD), which is used for tracking of charged particles close to the interaction region. Due to data measurement, recognizing a real signal out of a large number of background data arises as a problem. Thus, the primary objective is finding an analysis method to separate highly ionizing particles, such as anti-deuterons and magnetic monopoles, from e.g. beam background.
Since neural networks showed up as a useful tool for image and data recognition, several types of networks were developed. Already K. Dort [1] and S. Ks [2] considered multilayer perceptrons and self-organizing maps (SOM) for signal recognition of PXD data by transforming data properties in abstract input vectors.
Now, a completely different approach should be attempted. Instead of abstract input vectors raw pixel data can be taken as input data. Because measured data consists of a pixel field containing energy information, the idea is to recognize pattern by using a Hopfield network. This special network structure, in which every node is connected with each other, is mostly used for image recognition and showed good performances already [3]. Additionally, to achieve better performance, different optimization variants are considered, such as a detailed analysis of simulation data. For this research no real signal data was available, but simulated data's properties could be considered instead.

2. PHYSICS BACKGROUND AND BELLE II EXPERIMENT

2.1. Physics Motivation

The standard model of particle physics (SM) contains all so-far observed elementary particles and three of the currently known interactions: weak, strong and electromagnetic (EM) interaction. Gravitation is not included. Due to this and other phenomena like dark matter and dark energy, an extension of the SM is claimed from theoretical as well as experimental physics. At experiments like Belle II big collaborations of physicists, engineers and computational specialists are currently working on understanding and studying new effects in physics beyond the standard model.

2.2. Belle II

SuperKEKB accelerator, where Belle II experiment is located, was build to explore new physics and try to explain it. The accelerator reaches an instantaneous luminosity of $8 \cdot 10^{35} cm^{-2} s^{-1}$, an improvement compared to its predecessor KEKB [4]. This luminosity increase requires an update of the detector positioned at interaction point of electron-positron collisions.
In Fig 1 one can see the SuperKEKB structure with electron and positron accelerator rings on the left and a cross section of the collision region surrounded by detectors on the right. It includes the Pixel Vertex Detector (PXD), the Silicon Vertex Detector (SVD), the Central Drift Chamber (CDC), a Time of Propagation Counter (TOP), the Aerogel Rich Detector (ARICH), Electromagnetic Calorimeter (ECL), Superconducting Solenoid and the K_L and muon detector (KLM) [5]. The Vertex Detector consisting of the PXD and SVD provides accurate tracking close to the interaction region. The Central Drift Chamber (CDC) is installed for tracking information as well and measures the energy loss of particles. Identification of particles is done by the TOP and the ARICH. The EM Calorimeter (ECL) is designed to measure the total energy and the angular position of particles. The outermost KLM is used for muon and K_L identification [1].

2.2.1. Pixel Vertex Detector

The PXD consists of 40 modules in total, which are arranged in two layers of a windmill-like structure. The inner layer build out of 16 modules has a diameter of 14 mm and the outer one 22 mm. In Fig 2 one can see 8 ladders building the inner and 12 ladders the outer layer. Each ladder is made out of two modules glued together. One module contains 250×256 and the other 250×512 pixels. Signal measurement is realized by Depleted Field Effect Transistor (DEPFET) pixel sensors based on p-channel Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFET).
If one particle hits the detector it causes a signal of pixels around the point of impingement. These pixels can be combined to a cluster. Clusters have specific shapes, charges and positions, depending on the sort of incoming particles. Most clusters fit into a 5×5 pixel field. Thus, global data can be transferred in local data with relative coordinates u and v (Fig. 2). In the following relative coordinates of a local 5×5 pixel field will be simply

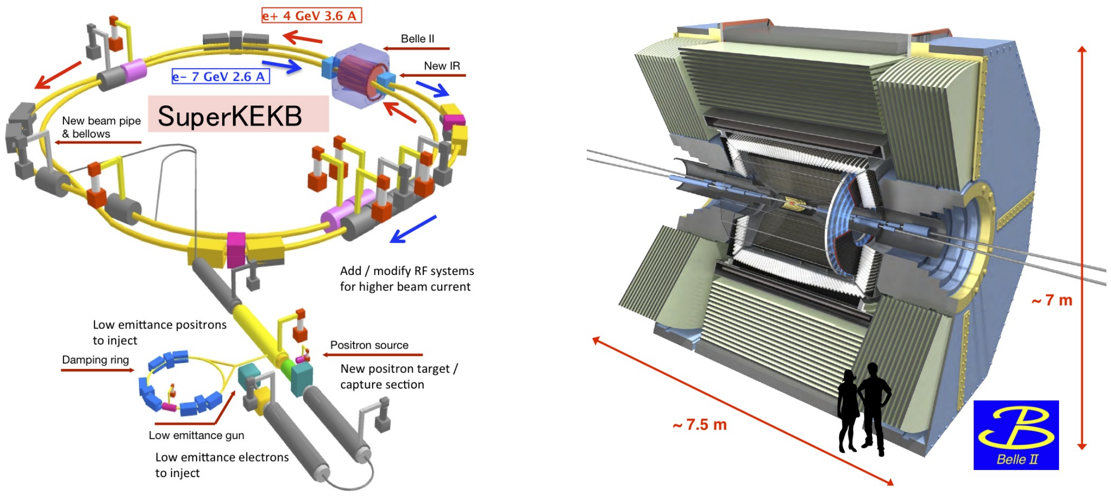

FIG. 1: SuperKEKB accelerator and detectors around interaction region[1]

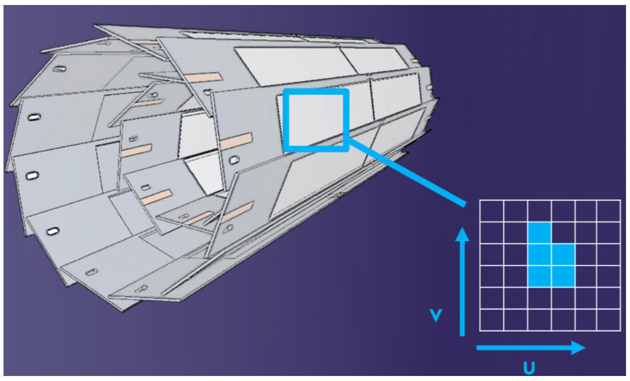

FIG. 2: Inner and outer layer of the PXD: One ladder made of two modules, which measure particles by bunching signal pixels to clusters[2]

named x and y, since global coordinates are not explicitly mentioned in this research.

[1] https://www.belle2.org/project/super_kekb_and_belle_ii
[2] S. Ks, K. Dort, J. S. Lange: Poster on Multiparameter Analysis of the Belle II Pixeldetector's Data

3. HOPFIELD NETWORK

3.1. Neural Networks

In the last few years neural networks got more and more importance in economical, medical and other sientific sectors, such as data analysis. Especially in image recognition it yields good results and proved to be very useful. Beside most common network structures like deep layer systems, a large number of various network structures for different applications has been created, differing in size, input data, connection between neurons etc.

3.2. Structure

A special architecture of neural networks can be achieved by connecting every neuron to every other neuron. This idea is used in Hopfield networks. Hopfield networks, named after John Joseph Hopfield, are especially used for pattern recognition. Therefore, every pattern entry is a single node, connected to every other node without self-interaction. The connections are determined by symmetric weights in a weight matrix. This structure implies, that every node is in the same layer, which is input and output at the same time (Fig 3). All states are either binary $\{0, 1\}$ or bipolar $\{-1, 1\}$.

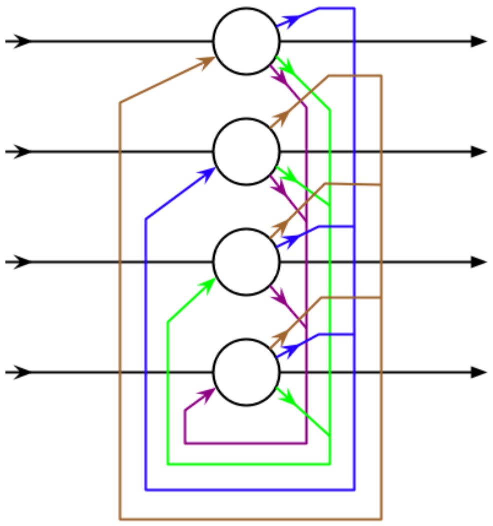

FIG. 3: Structure of a Hopfield network with four nodes: every node is connected to each other and functions as input and output layer node[1]

3.3. Hebbian Rule

Saving patterns, that the network should remember is done by training. This can be realized by several learning rules. Each of them will provide a associative memory by calculating the connection weights between two neurons.

[1] https://de.wikipedia.org/wiki/Hopfield-Netz#/media/Datei:Hopfield-net-vector.sv

The simplest learning rule for training Hopfield networks is the Hebbian learning rule [6]. The connection between two nodes is determined by the sum of their entries in each pattern, that should be stored. If the pattern entries $x_i, x_j \in \{-1, 1\}$, it holds that

$$w_{ij} = \sum_{k=1}^{p} x_i^k x_j^k \tag{1}$$

in case of binary entries every 0 has to be transformed into -1, this can be done by

$$w_{ij} = \sum_{k=1}^{p} (2x_i^k - 1)(2x_j^k - 1) \tag{2}$$

with $\forall\, i = j \Rightarrow w_{ij} = 0$ to avoid self-interaction. In the following only bipolar structure will be used, binary is analogue to this example. For N nodes that yields a N $\times$ N weight matrix, which is symmetric and zero on the diagonal, such that only $\frac{N^2}{2} - N$ weights have to be calculated[16].

3.4. Storkey's Rule

Another way of training a Hopfield network is the Storkey learning rule, introduced by Amos Storkey in 1997. For bipolar input it is defined recursive [7] as

$$w_{ij}^k = w_{ij}^{k-1} + x_i^k x_j^k - x_i^k h_{ji}^k - x_j^k h_{ij}^k \tag{3}$$

and additionally considers the local field

$$h_{ij}^k = \sum_{m=1:m\neq j}^{N} w_{im}^{k-1} x_m^k \tag{4}$$

Such as in Hebbian learning rule to aviod self-interaction it requires that $w_{ij} = 0 \; \forall\, i = j$. Storkey learning rule has shown a significant improvement of capacity for Hopfield networks compared to the conventional Hebbian learning [8].

3.5. Updating Nodes

Once a network is trained, it can be used for pattern recognition. Therefore, a test pattern will be given as input. Updating nodes can then be done synchronously or asynchronously. In synchronous updating all nodes are considered in the same step, i.e. updating one node doesn't influence the calculation of all other node states in the same step. However, in asynchronous updating modification of one node influences the following updates, such as a cycle of updating every pattern bit contains many sup steps. The order of nodes is randomly chosen.
In both cases the weighted sum V_{ij} has to be calculated [6]. Therefore, the value of the considered node and the value of each other node multiplied by the weight matrix is summed up.

$$V_i = \sum_{j=1}^{N} w_{ij} x_i x_j \tag{5}$$

If V_{ij} is greater than a given threshold θ_i[15], the value will be updated to 1, otherwise it will be updated to -1. This procedure is repeated until a stable end state is achieved. Since the network has a limited number of possible states, it can be proven, that it always terminates. Three different types of end states exist: saved pattern state, reversed saved pattern state and a spurious state. The last one arises from storing several patterns in Hopfield networks and should be mostly avoided.

3.6. Ising Model

To understand the idea of the energy function, a simple solid state model is considered. The Ising model is an approximative description for spin-spin interaction in crystal structures, which assumes that spins only contain the z-component with values up (1) or down (-1). The energy of a lattice system (Fig 4) is given by

$$E = -\sum_{\langle i,j \rangle}^{N} J_{ij} s_i s_j - \mu \sum_{i=1}^{N} h_i s_i \tag{6}$$

where J describes the spin-spin interaction, s_i is the z-component of spin, μ the atomic magnetic moment and h_i an external magnetic field interacting with the ith spin. The expression of $\langle i,j \rangle$ determines the range of considered neighbour interactions. Often it contains only next neighbour interaction, but for calculating the exact model one have to look at all interactions. To get the equilibrium state, the system have to be at energy minimum. Therefore, a random configuration of spins is generated. Then, one picks any

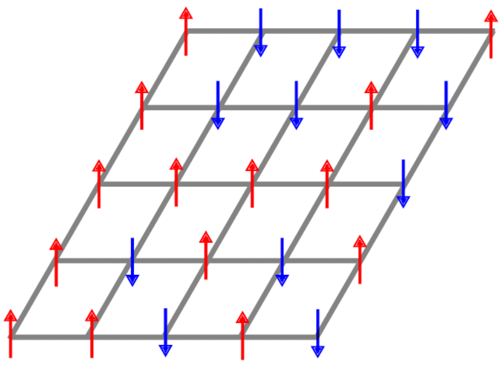
FIG. 4: 2D spin lattice structure in Ising model[1]

spin and calculates the system energy of the flipped spin. If this decreases the total energy, this spin will be flipped. That way, every spin will be calculated until energy cannot be decreased any more and a local minimum is reached.

[1] https://www.researchgate.net/figure/Schematic-representation-of-a-configuration-of-the-2D-Ising-model-on-a-square-lattice_fig2_321920877

3.7. Energy

Comparing the Ising model to a Hopfield network yields an analogue definition [9] of an energy function of the system:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j + \sum_{i=1}^{N}\theta_i x_i \tag{7}$$

The interaction is determined by the connection weight between two nodes and their values and the external field is replaced by the threshold. Updating nodes implies decreasing the total energy in every single step, until a local minimum is reached. Thus, once being in a local minimum the system will stay there. This enables recognition of spurious pattern, which will be discussed in the following section.

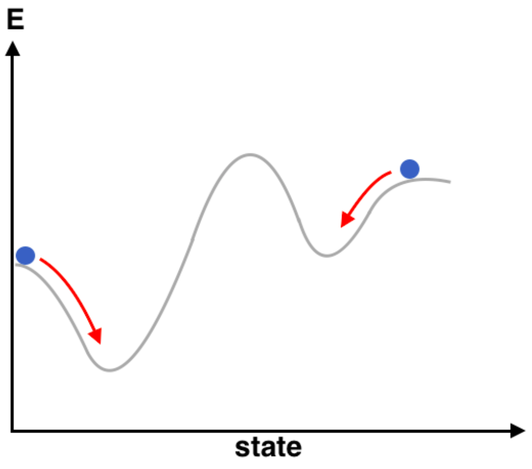

FIG. 5: Schematic figure of an energy function: Starting from a given state (blue dots) the network will proceed to a local minimum. States are all possible configurations of a pattern.

3.8. Capacity

The capacity of a network describes how many pattern can be stored, such that it is still able to recognize them. For Hopfield networks this it not well defined, since different sets of saved patterns define different weight matrices and thus different energy contours in the space of states (Fig 5). If saved pattern are "far" enough, i.e. fulfil special characteristics concerning orthogonality, the number of stored pattern p learned by Hebbian learning rule can be approached by $p/N < 0.14$, where N is again the number of nodes [10]. Nevertheless, this should be seen as a rule of thumb, because storage capacity in Hopfield networks is very pattern specific.

3.9. "Post Training"

One idea of increasing capacity of a Hopfield network is "post training" [11] [12], i.e. modifying the trained network's weight matrix W. Therefore, a complex trained network is considered as shown in Fig 6. A simple idea to emphasize the saved target pattern is to lift

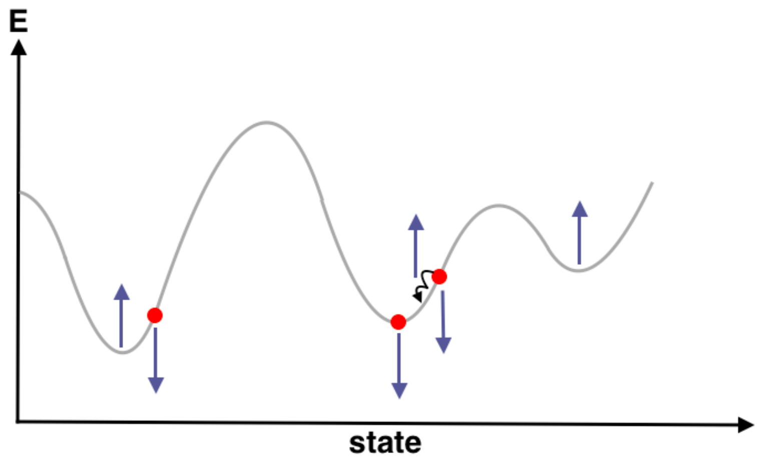

FIG. 6: Schematic figure of a complex network's energy function: Target pattern (red dots) should be shifted down, all other local minima should be shifted up. Unstable target pattern (right red dot) can be avoided by calculating a few steps and lift it up.

the energy of all other sates up and decrease for the target states X_p. To achieve a better capacity, spurious states X_s should be removed. Thus, a much easier and effective approach is to lift only these energies (blue up arrows) and additionally decrease them for the saved states (red dots with down arrows) by a parameter μ.

$$W = W + \mu \left(\sum_{x \in X_p} x_i x_j - \sum_{x \in X_s} x_i x_j \right) \tag{8}$$

To avoid unstable target patterns as shown in Fig 6 in the middle local minimum, one can calculate a few iterations, starting from the unstable pattern, and lift up the intermediate state's energy.

3.10. Activation Function

In neural networks, the activation function defines the output of a neuron by the given input. So far, a simple step function was used with threshold θ. The most common activation functions are logistic sigmoid function and hyperbolic tangent function [9]. In this way, additional information as probabilities for bit flipping can be fed into the network.

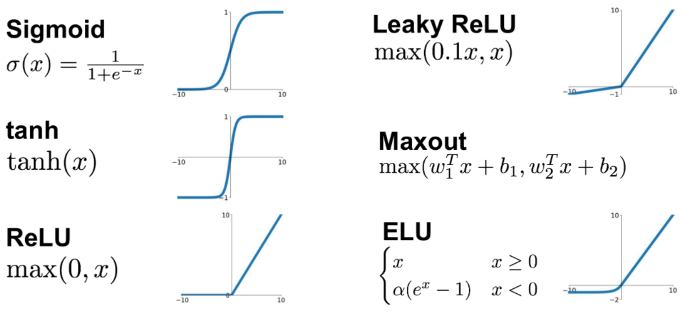

FIG. 7: Different types of activation functions to transfer input into output of a neuron[1]

[1] https://www.researchgate.net/figure/Schematic-representation-of-a-configuration-of-the-2D-Ising-model-on-a-square-lattice_fig2_321920877

4. PXD DATA

4.1. Cluster Properties

First, one have to define how to feed data into the network. As described in section 2.2.1, incoming particles cause signals on several pixels of the detector, which form a cluster. Each pixel measures an energy in form of charge. Clusters usually are not greater than a 5×5 pattern. This data has to be modified, such that the Hopfield network can deal with it as input. Also the global signal position in the PXD is given as additional information. The

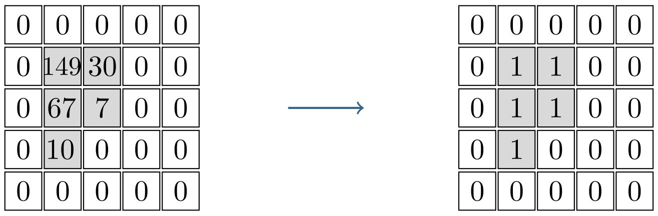

FIG. 8: Transformation of local PXD data to valid input for a Hopfiel network

trivial idea is to transform the pixel data as shown in Fig 8, such that every signal pixel gets the value 1 and all others are 0. This binary data contains only the shape of the measured cluster. Nevertheless, a cluster has many properties, which can be observed, such as:

- total number of signal pixels
- x-length, y-length
- maximal charge
- total charge
- position in detector and angle

As it turned out, the simple transformation to a binary pattern is not enough information to distinguish signal data from background. Therefore other properties should also be concerned.

4.2. 3D Pattern

One idea to enlarge input information is to include the pixels' charge values by extending the 2D pixel field to a 3D pattern, where the third dimension represents charge. Since the measured charge range is from 0 up to 250, it is split in 5 parts all of the same size of 50. This value seemed the most suitable not slow down calculation time to much. As shown in Fig 9 pixel data is easily converted into a binary 3D picture. This enlarges the capacity to approximately $p \approx 17$, instead of 3 for 2D, but since all signals will have a similar shape in z-direction, not a large increase of capacity is expected.

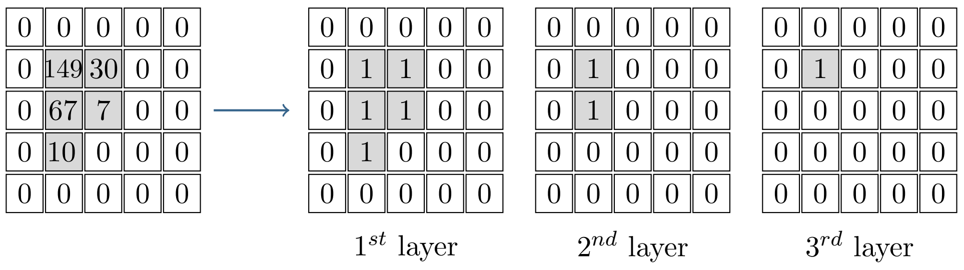

FIG. 9: Transformation of 2D pattern into a 3D pattern: one layer represent a charge of 50, so layer 4 and 5 are filled with zeros.

4.3. Data Analysis

Up to now, only shape and a reduced information of charge are taken into account. The concept of activation functions, described in chapter 3 3.10, enables passing more information to the network. To understand the role of several properties, it is meaningful to analyse the given simulation data for the total number of detecting pixels, x- and y- length, such as the maximal pixel length, maximal charge and the relation between total charge and global position in the pixel detector. Therefore, the rate of signal and background is calculated and fitted with a suitable function. The fits can later be included in the activation functions. How this can be done will be discussed in section 4 4.4.

4.3.1. Number of Signal Pixels

For the total number of pixels measuring energy, the rate for signal and background of all detected particles is calculated. In Fig 10 the blue line represents the signal and orange represents background. Fitting the signal with a function

$$f_1(n_{tot}) = \frac{a}{1 + \exp\left(b \cdot n_{tot} + c\right)} \tag{9}$$

yields the parameters $a = 0.55$, $b = 0.8$ and $c = -4.84$. In this analysis, it is important that the fitted function is bound to range of $[0, 1]$ in domain of n_{tot}, so that it can later be used as a probability for bit flipping.

4.3.2. x- and y-length

The same analysis is done for x- and y-length of a cluster. For the fitted function of x-pixel-length n_x, one gets

$$f_2(n_x) = 0.5 - \frac{0.5}{1 + \exp(-0.81 \cdot n_x)} \tag{10}$$

and

$$f_3(n_y) = \frac{0.5}{1 + \exp(2 \cdot n_y - 9.3)} \tag{11}$$

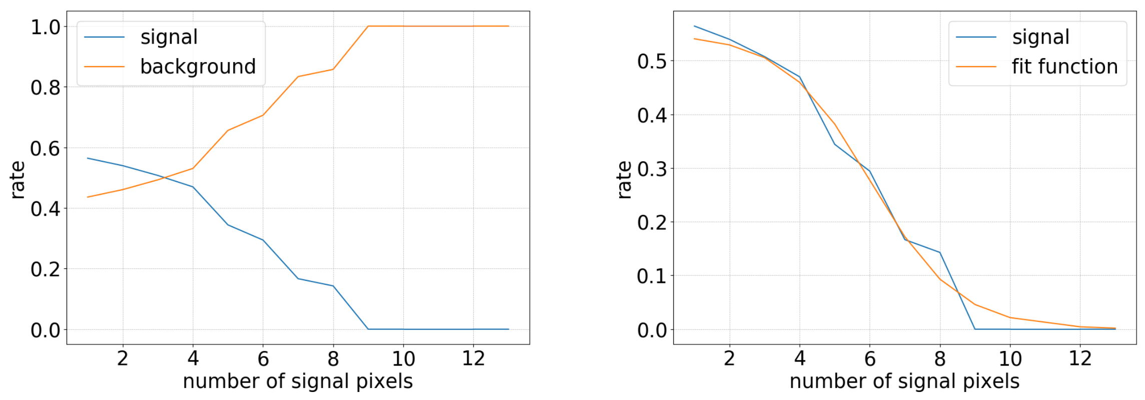

FIG. 10: **(a)**: Rate of signal (blue) and background (orange) depending on the total number of pixels n_{tot}, **(b)**: Fitted function $f_1(n_{tot})$ (orange) for the signal rate

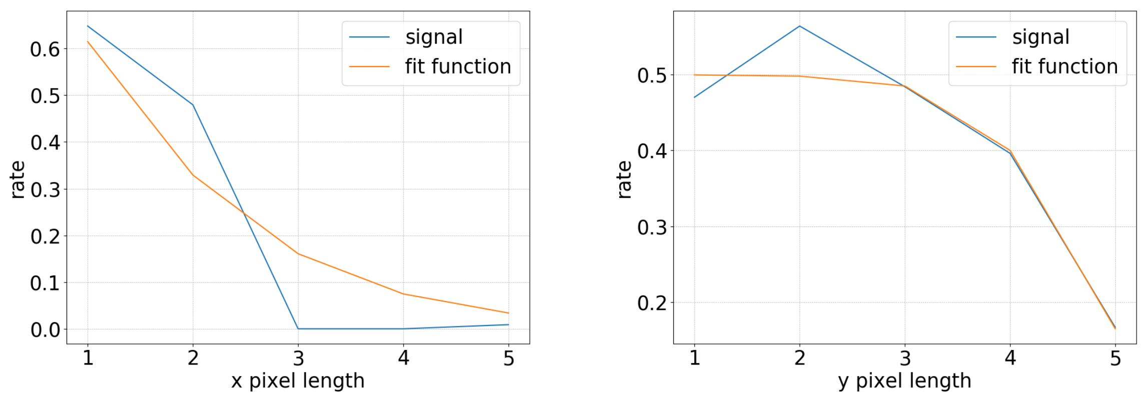

FIG. 11: Fit function for signal rate depending on **(a)** x-pixel-length n_x and **(b)** y-pixel-length n_y

for the y-pixel-length fit. In addition to x- and y-length, the maximal pixel length n_{max}, which is equivalent to the maximum of both values, is calculated and observed too. Again, signal rate and fit function are shown in Fig 12. The function is given by

$$f_4(n_{max}) = \frac{0.54}{1 + \exp(3.08 \cdot n_{max} - 13.34)} \tag{12}$$

4.3.3. Maximal Charge

By analysing the maximal charge, one obtains a clear trend of signals at higher and background at lower charges. In Fig 13 (a) rates of signal are shown in blue and background in orange. Looking only on the signal rate (Fig 13 (b)), the function

$$f_5(q_{max}) = 1 - \frac{1}{1 + \exp\left(-a \cdot q_{max} + b\right)} \tag{13}$$

can be fitted, such that $a = -0.027$ and $b = 0.5$.

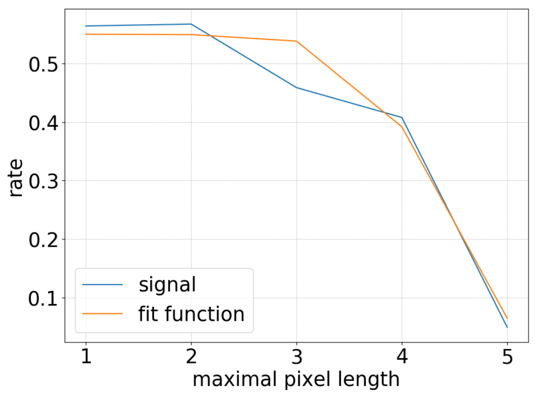

FIG. 12: Signal rate of all detected particles in dependence of the maximal pixel length and fit function f_4

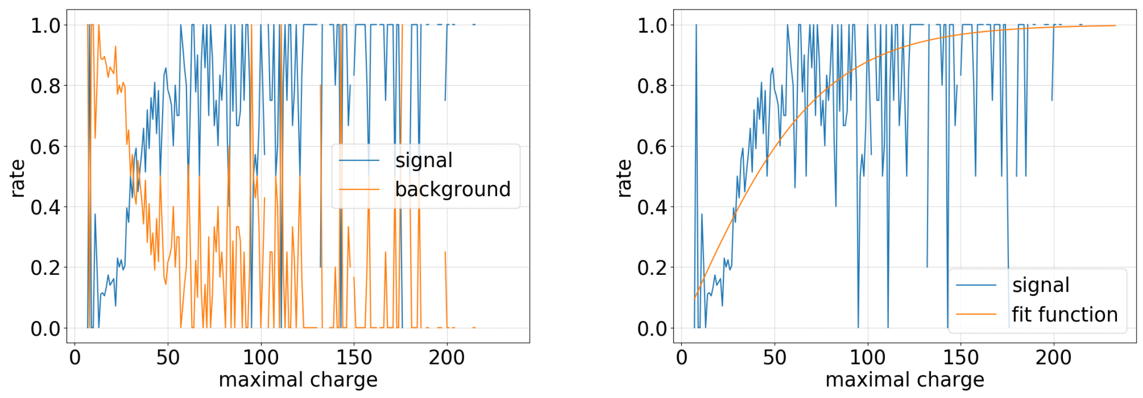

FIG. 13: **(a)**: Rate of signal (blue) and background (orange) depending on the maximal charge q_{max}, **(b)**: Fitted function $f_5(q_{max})$ (orange) for the signal rate

4.3.4. Total Charge and Global Position

As next step, the global position and hence the angle of incoming particles shall be considered. Since only simulation data is analysed, it can be assumed that an event happens at global origin. Out of the connection line between origin and the global position in the detector, the track length l is calculated. To include charge loss of particles per track length, total charge q_{tot} is divided by l. In Fig 14 the signal rate for this ratio is fitted by the function f_6:

$$f_6\left(\frac{q_{tot}}{l}\right) = 1 - \exp\left(-0.01 \cdot \frac{q_{tot}}{l}\right) \tag{14}$$

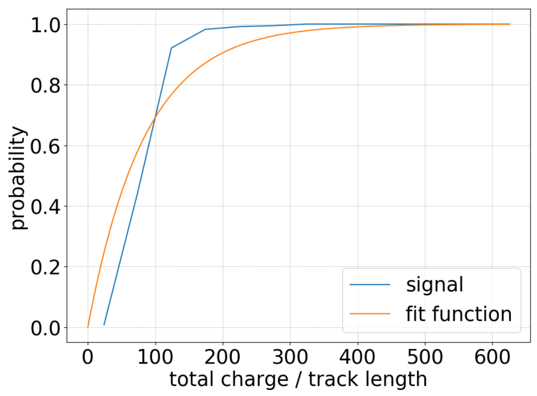

FIG. 14: Signal rate in dependence of total charge q_{tot} divided by track length and fit function $f_6(q_{tot})$

4.3.5. Zernike Moments

Dealing with a pixel patterns leads to the idea of image moments, which describe a weighted average of image pixel's intensities, often used in digital image processing. A special case is described by Zernike moments based on Zernike polynomials, defined by

$$Z_{nm}(x,y) = Z_{nm}(\rho,\theta) = R_{nm}(\rho)e^{im\theta} \tag{15}$$

where $n - |m|$ is even and

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-m)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!}\rho^{n-2s} \tag{16}$$

expresses the radial part [13]. Those form a complete orthogonal set of functions on the unit disk, which can be directly projected on an image by defining Zernike moments

$$A_{nm} = \frac{n+1}{\pi}\sum_x\sum_y f(x,y)Z^*_{nm} \tag{17}$$

where $f(x,y)$ contains the charge of each pixel.

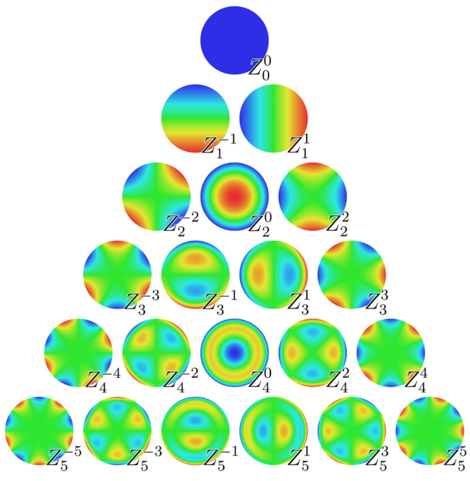

FIG. 15: The first 21 Zernike polynomials, ordered vertically by radial degree and horizontally by azimuthal degree[1]

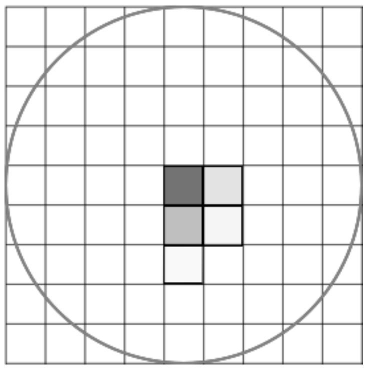

FIG. 16: 9 × 9 pattern with cut out disk of a diameter of 9 pixels: charge is represented by shades of grey, the highest measured charge is positioned in the centre

In Fig 15 the first 21 Zernike polynomials are illustrated. To adapt them to cluster analysis, one have to prepare the data first. Since the moments are invariant under rotation, only the central position has to be found. Therefore, raw local patterns are scanned for their highest charge. The 5×5 pattern is transformed to a 9×9 field with highest charge in the centre (Fig 16). Next, a disk of a diameter of 9 pixels is cut out and scaled to the unit disk. Out of this, finally Zernike moments of different orders are calculated. Moments up to order Z_3^m were concerned in this research. An appreciable separation between signal and background is observed for A_{00}, A_{20} and A_{33}.

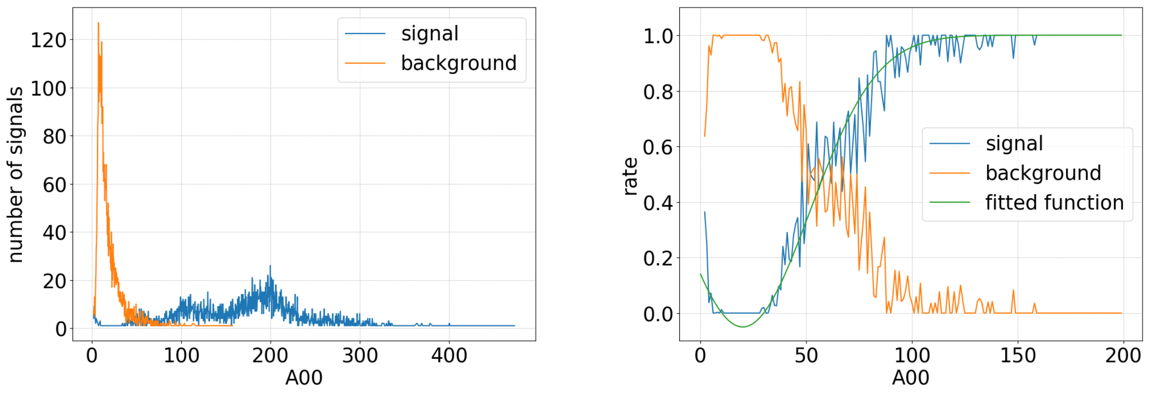

FIG. 17: **(a)**: Number of signal (blue) and background (orange) counts, **(b)**: Caltulated rate of signal (blue) and background (orange) with fitted function $f_7(A_{00})$ (green) for signal rate

Results of A_{00} are shown in Fig 17. The left figure shows a clear separation, where background (orange) have small values almost all under 100 and signals (blue) contribute to

[1] https://en.wikipedia.org/wiki/Zernike_polynomials#/media/File:Zernike_polynomials2.png

much larger A_{00} moments. On the right side, signal and background rates are calculated and signal rates are fitted by the function

$$f_7\left(A_{00}\right) = 1.05 \cdot \exp\left(-(0.02 \cdot A_{00} - 0.44)^2\right) \tag{18}$$

Since this function also should be treated as a probability for the activation function, for the domain with negative values the function is just set to 0.
The same procedure is done for A_{20}. There, one gets the fitted function

$$f_8\left(A_{20}\right) = 1.2 \cdot \exp\left(-(0.01 \cdot A_{20} + 0.55)^2\right) \tag{19}$$

Again, negative values are simply set to 0 to handle the function as probability. Moment

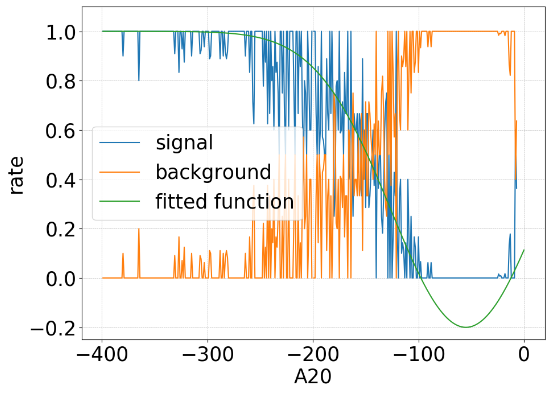

FIG. 18: Signal rate (blue), background rate (orange) and fitted function $f_8(A_{20})$ for the signal rate

A_{33} has also an imaginary part, so real and imaginary part are separately plotted in Fig 19. The fitted functions are defined by

$$f_9\left(\mathrm{Re}\left[A_{33}\right]\right) = \frac{1}{0.5 + 0.5 \cdot \exp\left(-2 \cdot \mathrm{Re}\left[A_{33}\right] - 0.5\right)} \tag{20}$$

$$f_{10}\left(\mathrm{Im}\left[A_{33}\right]\right) = 0.5 - \frac{0.5}{1 + \exp\left(-\mathrm{Im}\left[A_{33}\right] - 0.6\right)} \tag{21}$$

Other moments didn't show a good separation of particles, which is why they are neglected here. However, four additional input information for the network's activation function are found.

4.4. Activation Function

In addition to pattern information, the network is so far fed with, ten functions are found to describe the probability of being a signal for a measured particle with several properties. This can be included into the activation function, which influences bit flipping during node updating. In this case of separating two types of particles the network implementation is quite simple:
The Hopfield network is trained with two prototypes of signals and two prototypes of

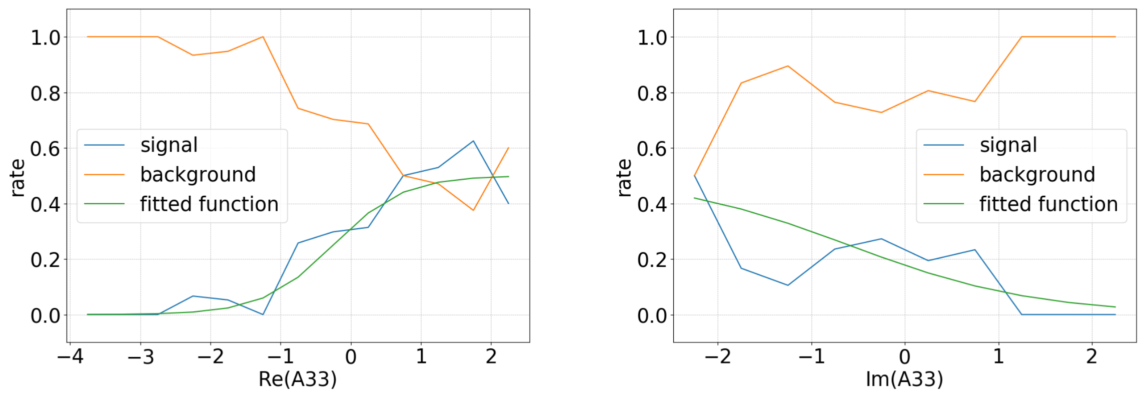

FIG. 19: Rate of signal (blue) and background (orange) for real **(a)** and imaginary **(b)** part of A_{33} with fitted functions $f_9(\mathrm{Re}\,[A_{33}])$ and $f_{10}(\mathrm{Im}\,[A_{33}])$ (green) for signal rates

background. Two weight matrices are defined, one for signals, the other for background. Both weight matrices are separately calculated as described in section 3 3.3 and 3 3.4, their weighted sum leads to the full weight matrix. The activation function defines the weights to sum these matrices. Without it, both are counted equally. The principle is shown in Fig 20. The activation function f specifies the probability of being a signal. It is a combined function of all fits of the previous chapters, weighted by their rate range of possible domain. This yields

$$f = \frac{1}{6.6}\left(0.548f_1 + 0.6f_2 + 0.352f_3 + 0.5f_4 + f_5 + 0.7f_6 + f_7 + f_8 + 0.5f_9 + 0.4f_{10}\right) \qquad (22)$$

That way, the trained network can use a modified weight matrix to update nodes to recognize input patterns.

$$\begin{tikzcd}
\begin{pmatrix} w_{11}^{signal} & \cdots & w_{n1}^{signal} \\ \vdots & \ddots & \vdots \\ w_{1n}^{signal} & \cdots & w_{nn}^{signal} \end{pmatrix} \arrow[dr, "f"'] & & \begin{pmatrix} w_{11}^{background} & \cdots & w_{n1}^{background} \\ \vdots & \ddots & \vdots \\ w_{1n}^{background} & \cdots & w_{nn}^{background} \end{pmatrix} \arrow[dl, "1-f"] \\
& \begin{pmatrix} w_{11} & \cdots & w_{n1} \\ \vdots & \ddots & \vdots \\ w_{1n} & \cdots & w_{nn} \end{pmatrix} &
\end{tikzcd}$$

FIG. 20: Signal and background matrices are summed, weighted by activation function f.

5. RESULTS

5.1. Network Performance

A Hopfield network was implemented as described in the previous chapters. During developing process for the most efficient network, it occurs that the Stokey learning rule, which predicts a better capacity and lower rate of "fake" states, didn't improve efficiency in this case. It was tested to store two patterns in a 2D pattern. In Fig 21 the weight matrices after each learning rule are depicted. In Hebb's learning rule only values of -2, 0 and 2 are possible. Maxima (yellow) and minima (purple) occur, if both pattern have the same value, at mixed states the value is 0 (green). Using the other learning rule obviously extrema at the crossing points, i.e. the differences between patterns, are emphasized much more. Since

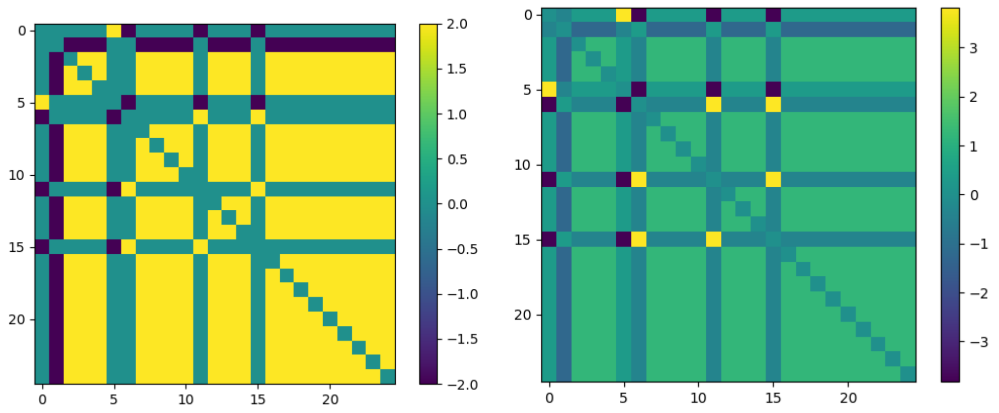

FIG. 21: Weight matrices after Hebb's learning (right) and Stokey's learning rule (left): Storkey's learning rule emphasizes extrema at crossings of saved pattern states and flattens all other states.

it didn't lead to an improvement of accuracy, but extended calculation time, the Hebbian learning rule was chosen.
For several parameters μ the 2D network was "post trained". Despite expectations, performance did not improve for the stored signal patterns. Thus, "post training" was discarded for the further development. Since no additional information is given to the network, signal and background pattern are to similar to separate from each other.
Another idea to optimize accuracy was to move clusters in a pattern's corner. Usually, as shown in chapter 4 4.3 4.3.2 and 4 4.3 4.3.3, cluster sizes are small, such that it seems difficult to recognize the same cluster on another position. So, clusters are moved to the top left corner of a pattern before recognition process.
In the next step, patterns were extended to the third dimension to $5 \times 5 \times 5$ and the first reasonable results with an accuracy[16] of 63.0 % were obtained.
Finally, the activation function was introduced to the network. First, one started to implement only the local properties' functions. These are properties of cluster shape and entries, like total number of signal pixels, x-length, y-length, maximal charge and total charge. The accuracy was increased to 76.5 %. Since the activation function showed a crucial improvement, the global property of position in the detector was included by adding function f_6. That lead to an accuracy of 95.7 %. Finally, also Zernike moments completed the activation function, which lead to an accuracy of 97.7 %. Because a ratio of 50:50 is not the usual

3D	local prop.	global prop.	Zernike mom.	accuracy
✓	x	x	x	63.0 %
✓	✓	x	x	76.5 %
✓	✓	✓	x	95.7 %
✓	✓	✓	✓	97.7 %

TABLE I: Accuracy results of included optimization

measuring ratio, one have to calculate efficiency ν_{signal} and background rejection $\nu_{background}$ to be able to compare it to other networks and methods of separating experimental input. The efficiency and rejection are given by

$$\nu_{signal} = \frac{\text{number of correctly identified signals}}{\text{total number of signals}} \tag{23}$$

$$\nu_{background} = \frac{\text{number of correctly identified background}}{\text{total number of background}} \tag{24}$$

For the final network with best performance it holds

$$\nu_{signal} = 96.83\,\%$$
$$\nu_{background} = 98.49\,\%$$

At this point, it should be mentioned again, that all test and training data were taken from simulations and are not the real data. Since only real background data are available, no statistics to adapted activation functions could be made out of real data. Nevertheless, testing real background data leads to a background rejection of

$$\nu_{background} = 95.60\,\%$$

These values are independent of the input's ratio of signal and background and can be compared to other analysis methods.

6. CONCLUSION AND OUTLOOK

Hopfield networks in general perform well for image recognition. In this case, it turned out to be very pattern specific and restricted in capacity. It became clear, that for signal separation more information than only cluster shape is needed. In this research a lot of optimization methods were concerned, such as an expansion of dimension showed an improvement of accuracy. But the best performance we only could achieve by implementing an activation function to determine a neuron's output by its weighted input. Obviously, this successfully provides the opportunity to feed additional data in form of a probability into the network.
Basically, the implemented method is a mixture of a neural network and previous data analysis, what makes it static. To enable a more dynamic method, one could also train the activation function. That would increase the time it takes to train the network, but makes it easier to adapt it to real data. However, many problems occur with it, such as guaranteeing convergence of fitting a function to a data set during training process. The functions' values also should stay between 0 and 1 to ensure the meaning of a probability. So this idea arises as a possible but complex opportunity to modify the static behaviour of the Hopfield network.
Compared to SOMs, a Hopfield network seem to perform better, due to ratio of efficiency and background rejection, but cannot conquer multilayer perceptrons [1]. Nevertheless, training this Hopfield network is based on a completely different approach, which also seems to be reasonable and can still be improved by fine tuning of the activation function. Since only simulated data was concerned, the network performs worse for real data. By analysing real signal and background data a more exact activation function can be found.

[1] K. Dort, University of Giessen, *Search for Highly Ionizing Particles with the Pixel Detector in the Belle II Experiment*, .

[2] S. Kaes, University of Giessen, *Multiparameter Analysis of the Belle II Pixeldetectors Data*, .

[3] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the national academy of sciences 79.8. (1982) 2554–2558.

[4] *SuperKEKB and Belle II*, https://www.belle2.org/project/super_ kekb_and_belle_ii .

[5] *Belle II Detector*, http://belle2.kek.jp/images/BelleII3D.pdf .

[6] *Hopfield Network*, http://web.cs.ucla.edu/ rosen/161/notes/hopfield.html (IJLTET) .

[7] X. Hu, Bejing Technology Group, *Storkey Learning Rules for Hopfield Networks*, .

[8] S. A., Imperial College, *Increasing the capacity of a Hopfield network without sacrificing functionality*, .

[9] A. Z. A. S. S. Alzaeemi, S. A. S., *Activation Function of Hopfield Neural Network in Agent Based Modeling*, Journal of Genetics and Genetic Engineering **2** (2018) 7–14.

[10] L. M. Folli V. and R. G., *On the Maximum Storage Capacity of the Hopfield Model*, Frontiers in Computational Neurocience (2017) .

[11] R. Bhiksha, Carnegie Mellon University Deep Learning, *S18 Lecture 20: Hopfield Networks 1*, .

[12] A. A. S. J. Pooja Agarwal, Abhijit J. Thophilus, *Leveraging Different Learning Rules in Hopfield Nets for Multiclass Classification*, Future Technologies Conference (FTC) (2017) .

[13] C. K. P. Bhaskara Rao, D.Vara Prasad, *Feature Extraction Using Zernike Moments*, International Journal of Latest Trends in Engineering and Technology (IJLTET) **2** (2013) .

[14] The weight matrix can also be normed by the number of nodes, both variants are possible.

[15] θ_i is usually 0 $\forall i$ for a simple Hopfield model

[16] Accuraccy is defined here as all correctly identified data normalized on all input data (50 % signal, 50 % background)

Implementation Details

The Hopfield network of this research was implemented as a class structure in C++. In test.cc file an instance of this class is generated, trained and used. In the following the most important methods are listed. An additional helper file provides important methods to deal with files and arrays and other calculations.

HopfieldNetwork Class

- `void setNumberOfReadPatterns(int num):`
  Sets the number of training pattern, which will be read out of `trainingFilename` located in DetectionData (set in constructor of class). It can be changed by `void setTrainingFilename(std::string filename)`
- `void setMove(bool ifmove):`
  Determines, if a cluster will be moved to the left upper corner of the pattern before training or recognition and is set to `false` by default.
- `void setUseActivationFunction(bool use):`
  Determines, if the activation function will be used or not. It is set to `true` by default.
- `void train():`
  Trains the network with Hebb's learning rule.
- `void generateOrderOfUpdatingNodes():`
  Generates a random chosen order of a cycle to update nodes.
- `std::string detect(int* arr):`
  Recognizes an input pattern. It gets an array of length 25 and return a string with the recognition result.
- `float weightedSum(int networkStatePattern[numberOfNodes], int node):`
  Calculates the weighted sum and is used in `detect` method.
- `int threshold(float vIn):`
  Compares the weighted sum `vIn` to threshold 0.
- `float activationFunction(float charge, int xlength, int ylength, int maxlength, int totalpixels, float tracklength, float a00, float a20, float rea33, float ima33):`
  Returns the activation function.