NeuCube

Neurocomputing Software/Hardware Development Environment for Spiking Neural Network Applications in Data Mining, Pattern Recognition and Predictive Data Modelling

Knowledge Engineering and Discovery Research Institute
(http://www.kedri.aut.ac.nz)

Auckland University of Technology,

Auckland, New Zealand

July 2015

Disclaimer

NeuCube is a modular development systems to help graduate students, researchers and practitioners to create new, more efficient solutions to problems of data mining, pattern recognition, event prediction and decision support when dealing with complex and large data, especially temporal or/and spatio/spectro temporal data (SSTD) across domain applications. NeuCube uses the third generation of neural networks – the spiking neural networks (SNN) and the available neuromorphic hardware,

NeuCube is being developed and owned by the Knowledge Engineering and Discovery Research Institute (KEDRI, www.kedri.aut.ac.nz) funded by the Auckland University of Technology Strategic Research Investment Fund (SRIF).

Using NeuCube for teaching and research will require a licence and a small charge depending on the version, to recover the costs of its development and to allow the developers to further improve it.

Commercial use will require a special licence as NeuCube is a patent protected. Communication for obtaining any licence should be done through the person in charge (see the NeuCube web page: http://www.kedri.aut.ac.nz/neucube).

Contents

1. INTRODUCTION:

NeuCube is a software/hardware development environment for spiking neural network (SNN) prototype systems for data mining, pattern recognition and predictive data modelling with complex and large data, especially temporal or/and spatio/spectro temporal data (SSTD). A created application system with the use of NeuCube has the architecture of a spatio- temporal data machine (STDM). In this respect NeuCube is not a 'magic bullet' that can easily solve any of the above problems. Instead, it is a sophisticated framework of methods that facilitates the design and the implementation of efficient solutions to these problems through careful and precise selection and testing of most suitable methods and parameters for a STDM. This process can be slow, but the results can be very impressive in both accuracy and data understanding.

A STDM has an input part to encode input data into spiking sequences, a SNNcube that learns the input data in an unsupervised mode to capture spatio-temporal patterns, and an evolving output part for classification or regression tasks that is trained in an incremental, adaptive way in a supervised or in a semi-supervised mode to classify (calculate) the SNNcube patterns into output classes (or regression values). Additional modules can be added, such as a dynamic parameter model (e.g. a gene-regulatory network).

STDM are based on the principles of evolving connectionist system (ECOS) and neuromorphic computations. This architecture is first proposed in (Kasabov, 2014), initially for brain data modelling – Figure 1. NeuCube is PCT patent protected (Kasabov et al, PCT patent, 2013).

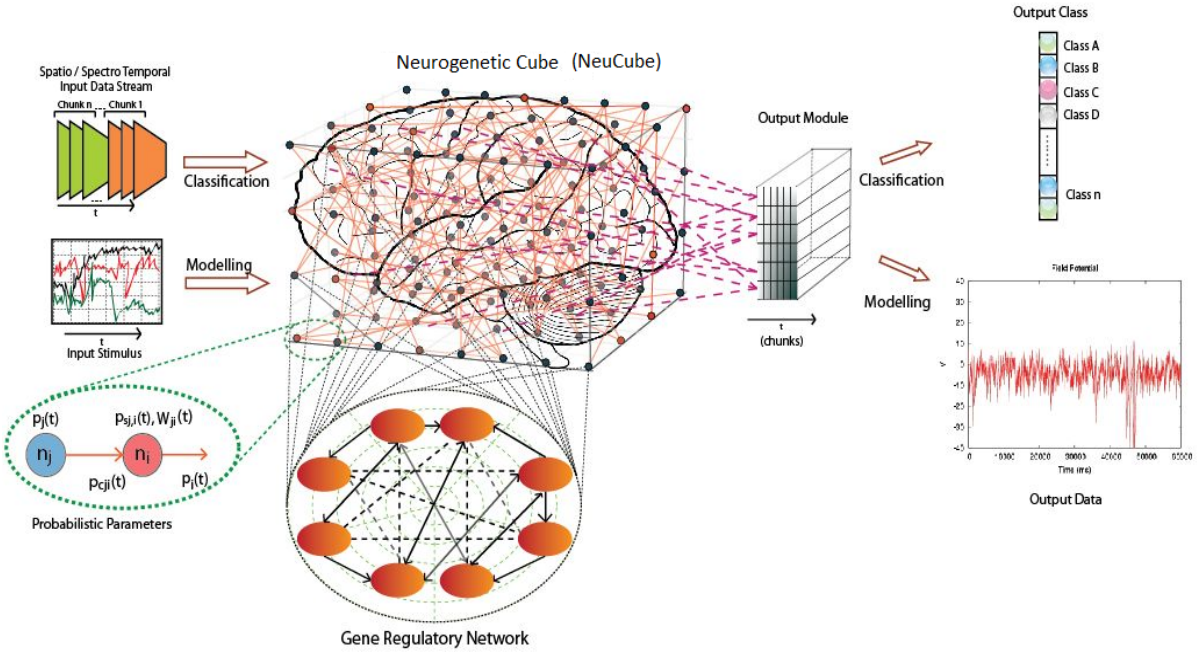

Figure 1: A functional diagram of the first NeuCube architecture introduced in (Kasabov, 2014; PCT 2013)

The above architecture was developed further as a multi-modular software/hardware development system for wider applications (Kasabov et al, 2015):

- Brain data modelling: EEG, fMRI, EMG, DTI; neurogenetic data; integrated brain data;
- Brain-Computer Interfaces;
- Robot control (e.g. neuro-rehabilitation robots with China Academy of Sciences);

- Prediction of neurodegenerative disease progression (e.g. Alzheimer's Disease data);
- Personalised event prediction (e.g. stroke occurrence; CVD);
- Ecological event prediction from temporal climate data;
- Audio/Visual data processing;
- Moving object recognition;
- Hazardous environmental event prediction (e.g. risk of earthquakes);
- Radio astronomy data modelling (Pulsar detection) – SKA;
- Bioinformatics (e.g. spatio-temporal protein folding and functions);
- Financial and business data modelling and prediction;
- Others

Figure 2 presents an example of a multi-modular NeuCube development system, where different modules perform different tasks, but all of them are integrated through a common communication protocol. A brief description of the modules from Figure 2 is given below.

Module M1 is a generic prototyping and testing module, where a SNN application system (called here Prototype Descriptor) can be developed for data mining, pattern recognition and event prediction from temporal or spatio/spectro-temporal data (SSTD, or as it is called here Data) After creating and testing a prototype descriptor in M1, it is saved in the I/O **module (M5)** and can be used in the other modules.

Module M2 is a PyNN-written simulator for small and large scale applications that can read the created and tested in M1 or in other NeuCube modules application system described as a Prototype Descriptor and its relevant Data, and run it in a fast mode. While this module can run independently, it also allows to run a Prototype Descriptor on a specialised neuromorphic hardware (e.g. SpiNNaker; the INI ETH chip etc.), indicated in Figure 2 as M**odule M3**. Module M3 makes it possible to execute a NeuCube prototype descriptor in a parallel manner, for fast on-line and real time applications.

Module M4 is a program written in Java for 3D visualisation of a NeuCube Prototype Descriptor. It allows for a virtual navigation through the 3D structure of a NeuCube model (prototype descriptor). It can also be used for visualization through the specialised Oculus VR 'goggles'. Dynamic visualisation is possible when the SNN model is observed in action.

Module 5 is for Input/output of data and for interaction between all NeuCube modules.

Module M6 which is written in Java, has all the functionality of module M1, but it has specific additional functions added for prototyping and testing of neurogenetic data models where in addition to brain data a SNN model can include relevant genes and proteins to study the problem in hand.

Module M7 facilitates the creation and testing of a personalised SNN system. It has all functionality of module M1 but also some special functions for personalised modelling (see Kasabov and Hu, 2010; Kasabov, USA patent 2008).

Module M8 has all functionalities of module M1, but it includes also some specific functions that allow for integrated brain data modelling including EEG, fMRI and DTI data together.

Module M9 is an optional module for data encoding optimization and event detection. This module will include state of the art data encoding technique for mapping analog signal to trains of spike.

Module M10 will provide additional feature of online learning for real time data analysis and prediction.

In this manual NeuCube v1.1 and NeuCube-M1 is used interchangeably. Both of them means one and the same. The NeuCube v1.1 is a limited trial version of the NeuCube Neurocomputing system. Following limitations on the size of dataset is imposed in this version.

1. Maximum number of class:10
2. Maximum number of samples:100
3. Maximum number of features: 64
4. Maximum number of neurons in SNNcube: 2000

Further detailed information on the NeuCube architecture and its applications are given in (Kasabov et al, 2015). Various applications of NeuCube are published in the references provided at the end of the Manual.

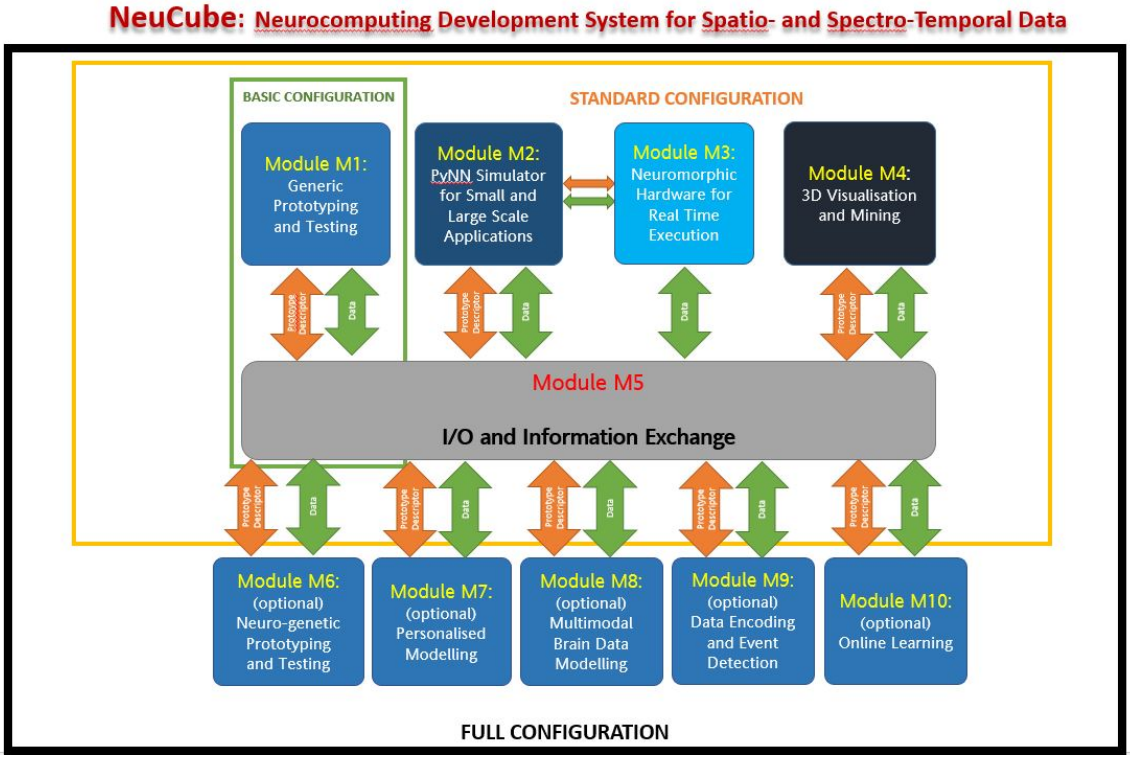

Figure 2: An example of a multimodular NeuCube development system for SNN applications

An exemplar configuration of a NeuCube development system is shown in Figure 3, where modules M1, M2, M3 (a small SpiNNaker board) and M4 are demonstrated.

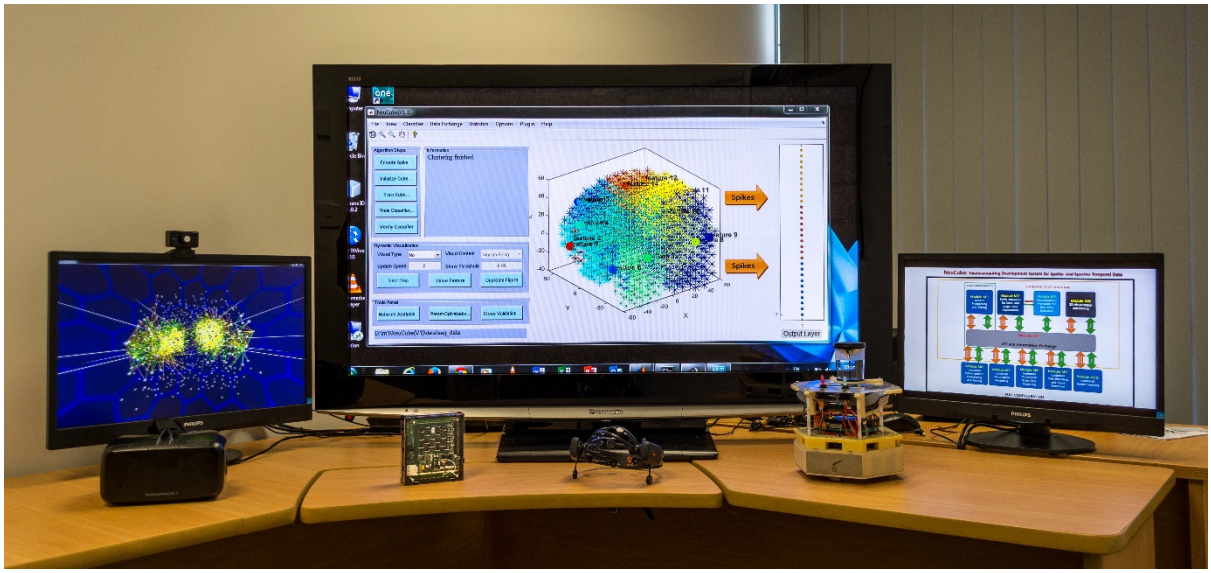

Figure 3: An exemplar configuration of a NeuCube development system is shown, where modules M1, M2, M3 (a small SpiNNaker neuromorphic hardware board) and M4 are demonstrated

The following is a time graph for the development and the release of different versions of the NeuCube development environment.

NeuCube v1.1 --> NeuCube v1.2. --> NeuCube v1.3

NeuCube v2.1. NeuCube v2.2

NeuCube v3.1 NeuCube v3.2

NeuCube v4.1

NeuCube v1.1 will contain Module M1 will be a limited trial version and will be distributed as a windows executable file, while NeuCube v1.2, v1.3 and further versions (beyond 2015) will include additional functionalities.

NeuCube v2.1 will contain Modules M1 and M4, while NeuCube v2.2 and further versions will have new functionalities added.

NeuCube v3.1 will contain Modules M1, M4, M2 and M3 (optional), while NeuCube v3.2 and further versions will have new functionalities added.

NeuCube v4.1 will contain all modules shown in Figure 2, while further versions of it will have new functionalities added.

This manual briefly summarises the functionalities of the NeuCube-M1 module that constitutes the NeuCube v1.1.

2. NEUCUBE INSTALLATION:

NeuCube-M1 distribution can be installed and run on a Windows 7/Windows 8 Operating System. A minimum memory of 4 GB is recommended. Internet connection is required for the installation.

NeuCube-M1 can be installed by running the NeuCube(v1.1)_installer.exe from the distribution folder. Double clicking on NeuCube(v1.1)_installer.exe runs the installer. The steps are self-explanatory and can be performed following the step by step procedure. If you do not have the Matlab Compiler Runtime(8.2/2013b) already installed in your computer, the installer automatically downloads and installs the Matlab Compiler Runtime(MCR). MCR is a large file (approx. 486 MB), and it may take some time (depending on the internet connectivity) for the installation procedure to be completed. Once the installation is completed, you can run the NeuCube-M1 module by double clicking NeuCube from 'Start' menu or from 'desktop' (if you have added a 'shortcut to desktop' during installation).

3. NEUCUBE USER INTERFACE:

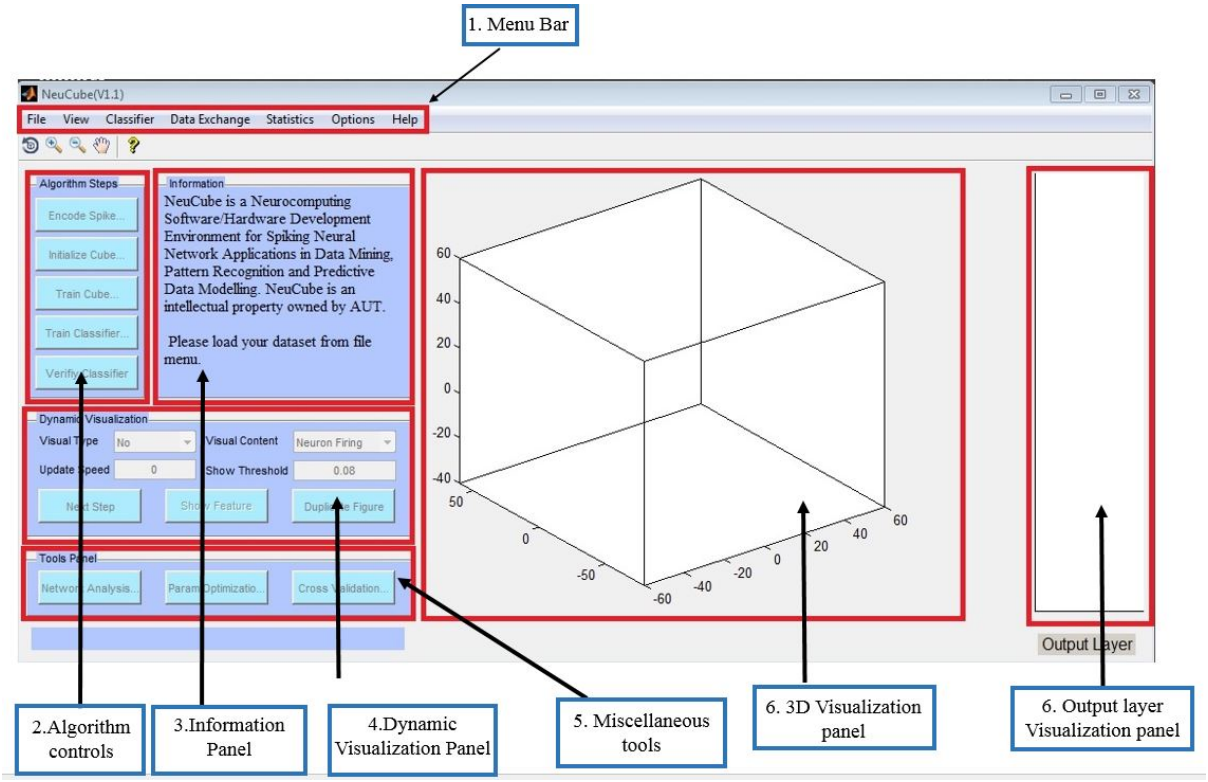

Figure 4: NeuCube-M1 module User interface and panel description

Figure 4 shows NeuCube-M1 module user interface and panel description in v1.1. It consists of seven different panels, which are:

1. **Menu bar**: The menu bar consists of several options for intra and inter modular information and data exchange, visualization of the 3D reservoir and the output layer.
2. **Algorithm controls** – Consists of controls to initiate the Stepwise learning process in the design and the testing of a STDM. This panel consists of a set of controls which are responsible for application of the core algorithms on the dataset.
3. **Information panel**: It is used to display information during a NeuCube model protyping.
4. **Dynamic visualization controls**: This panel consists of a set of UI controls, which can be used to visualize the learning of the SNNcube (controlled by panel 6) in real time during the 'train cube' phase of the process.
5. **Miscellaneous tools**: This panel initiates different functionalities, such as: 'network analysis toolbox'; 'parameter optimization'; and 'cross validation'.
6. **Visualization panel**: This panel visualizes the behaviour of the 3D SNNcube as connections and neuronal spiking activity.
7. **Output layer visualization panel**: This panel is used to visualize the behaviour of the output neurons created as a results of supervised training for classification or regression tasks.

4. DATA AND INFORMATION EXCHANGE:

NeuCube supports several data format, which are used for intra and inter modular data exchange. User should choose the file format appropriately to achieve efficiency in NeuCube-M1.

The NeuCube-M1 interacts with the external environment using different data descriptors. NeuCube architecture defines four type of data descriptor. They are the following:

1. Dataset descriptor: The Dataset descriptor consists of the data (and the metadata), that is to be learned and analysed. In majority of the cases, a dataset consists of a set of time series samples and the output label/value for the sample set. It is also possible to add miscellaneous information like 'feature name', 'encoding method' and other Meta information in the dataset.
2. Cube descriptor: The Cube descriptor consists all the information about the structure and learning in NeuCube. Some of the most important information stored in this descriptor are the spatial information of the input and reservoir neurons, structural information of the SNNcube and the state of the SNNcube during learning.
3. Parameter descriptor: The parameter descriptor stores all the user defined parameters including hyperparameters of data encoding algorithms, SNNcube training algorithms and classifier training algorithms.
4. Result descriptor: Result descriptor stores information about the experimental results produced by NeuCube.

5. Descriptor type	Mat	JSON	CSV
Dataset	Yes	Yes	yes
Cube	Yes	Yes	No
Parameter	Yes	Yes	No
Result	Yes	No	Yes

Table 1: Supported file format for descriptors

NeuCube-M1 supports three different file formats, Mat (binary), JSON (structured text) and CSV (comma separated plain text). Table 1 shows the supported file formats for each of the descriptor type. As a heuristic rule, mat format is recommended for achieving fast I/O. The CSV files are the recommended choice import/export of dataset and results. The JSON format is recommended for inter modular communication. The Dataset, Cube and Parameter files can be imported or exported during the lifetime of the experiments run in NeuCube-M1.

5. PROTOTYPE MODEL DESIGN AND TESTING, ILLUSTRATED ON A DEMO POBLEM

A NeuCube prototype model design and testing is a stepwise process, which is described in this section via a DEMO. The data files that are used in the DEMO can be found in data>wrist_movement_eeg.

Dataset: The dataset used in this DEMO, corresponds to a task of wrist movement in left, right and central direction. This task was performed on a single subject and EEG data was sampled from 14 channels at a sampling rate of 128 Hz, while the subject performed the task. 20 independent trials of 1 sec duration were collected while the subject performed each movement task. The DEMO dataset consists of the following files:

- Sample files (Mandatory): Each sample file (sam1.csv, sam2.csv, sam3.csv…….sam60.csv) contains data of one sample. Each sample corresponds to a data matrix arranged in comma separated format. The rows correspond to ordered time points, and the columns are the channel (feature).
  Sample files in any user defined dataset should follow the following naming convention:
  1. Sample files should begin with 'sam' followed by digits and must have a csv extension. Some valid sample filenames are sam1.csv, sam22_xyz.csv and others.
  2. The digits immediately following 'sam' defines the order of the sample. For example if the sample files are sam23.csv, sam10.csv, sam32.csv, the order of samples are interpreted as sam10.csv, sam23.csv, and sam32.csv.
- Target file (Mandatory): The target file is a file beginning with 'tar'. In the DEMO, tar_class_labels.csv corresponds to the target file. The target file in this problem stores the class label of each sample (ordered) in a column. It must be noted, a data folder should have only one target file.
- SNNcube coordinate file (Optional): This file describes the spatial coordinates of the neurons in the SNNcube. Every row in the SNNcube coordinate file corresponds to a SNNcube neuron and the columns correspond to the x, y, and z coordinate. The brain_coordinate.csv file is used as the SNNcube coordinate file in this DEMO. The (x, y, z) coordinates of the spiking neurons are the same as the coordinates of brain areas according to the Talairach brain template that makes it possible to represent in the same template any person's brain data. In the DEMO we used 1485 spiking neurons in the SNNcube where each neuron correspond to $1cm^3$ spatial resolution.
- Input coordinate file (Optional): This file describes the spatial location of the input neurons (features). Every row in Input coordinate file corresponds to an input neuron and the columns are the x, y, z coordinate. The eeg_mapping.csv file is used as Input coordinate file in this DEMO**Error! Bookmark not defined.**.

Loading a dataset

The NeuCube stepwise process begins by Loading data (from the File option in the main menu bar) for a chosen TASK of classification or regression, or for a recall of a trained model on new data. As shown in Figure 5, data can be loaded from the menu bar by clicking file->load dataset->classification/regression/recall.

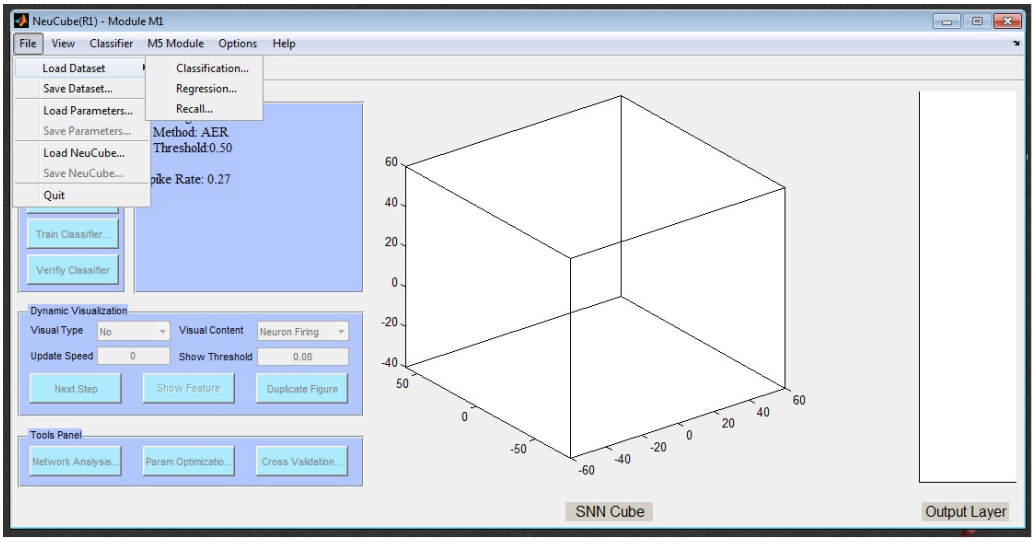

Figure 5. Loading a dataset

This example is a classification task, hence click on classification. This brings up the dialogue box shown in Figure 6. Click the 'browse' button to browse the data folder containing the sample and target files. As described before, the data folder should have one or more sample file and one target file. For the DEMO selecting the 'wrist_movement_eeg' folder and pressing 'OK' loads the sample in the NeuCube-M1 environment. Figure 7 shows the metadata corresponding to the DEMO dataset shown in the information panel after the data is loaded successfully.

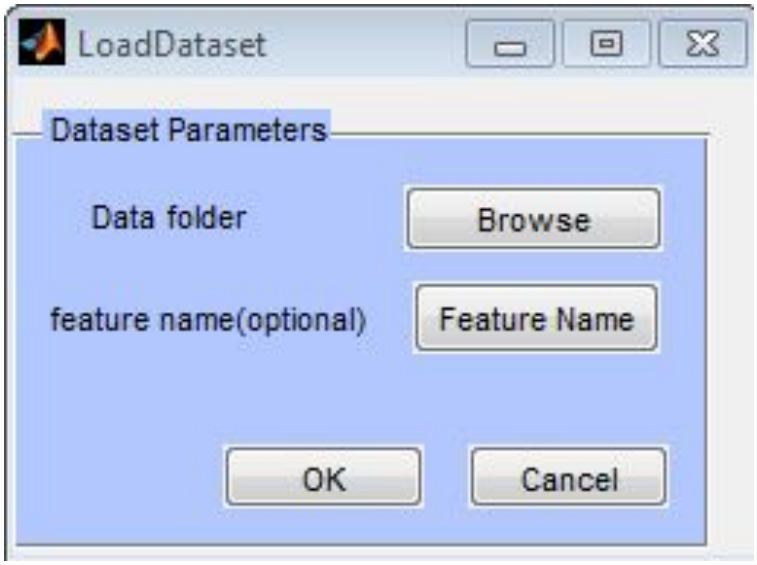

Figure 6: dataset parameters

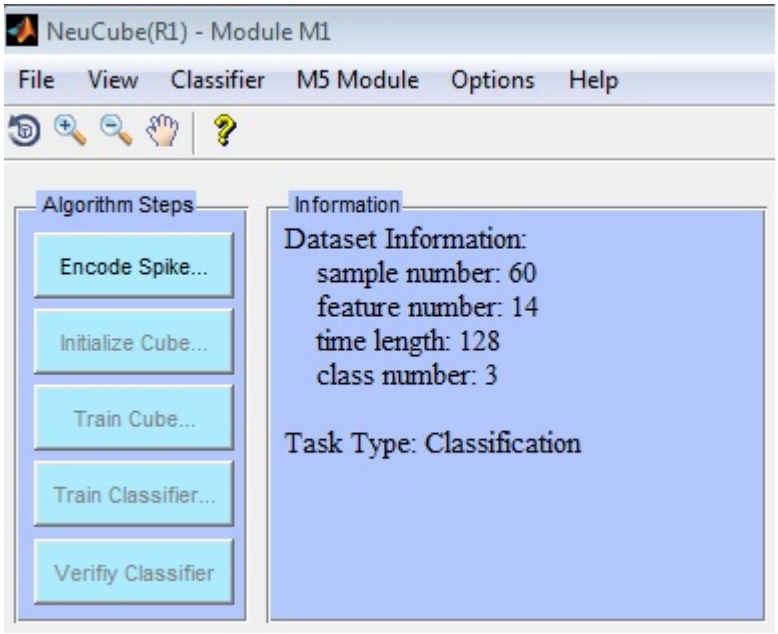

Figure 7. Information panel after data loading

In NeuCube v1.1 the size of the input data samples (the length of the temporal data points) is fixed (e.g. 128 for the DEMO problem here), but in principle the NeuCube approach allows for samples of variable time length and also for a variable number of input variables (features) used in each sample and for missing values. These additional options will be implemented in the next versions of the NeuCube v1.x.

Data Encoding

The next step after data loading is to encode the real valued input data into trains of spike, which will be used by the learning algorithms to learn spike patterns in the SNNcube (unsupervised), and then – in the output module (classifier/regressor) . Clicking on the 'Encode Spike' button generates a new UI panel as shown in Figure 8. This panel is used for following purpose:

1. Choose the encoding algorithm: NeuCube-M1 implements different Encoding method, which can be chosen from the 'Encoding method' dropdown menu. The hyperparameters like 'Spike Threshold', 'Window Size' etc. can be tuned as per the chosen algorithm.
2. Choose dataset Split: This panel is also used to specify the (random) split of the dataset for training and validation (Training Set Ratio). Default parameters can be used for the DEMO purpose. The 'Training Time Length' and 'Validation Time Length' define the percentage of the temporal length of the samples used for training and validation correspondingly (e.g. 0.8 means 80%; 0.7 means 70%, etc.). This option allows for a testing to be done on shorter temporal samples, when the model is expected to predict an event, say based only on 70% of new input data.
3. Encoding Visualization: This subpanel is used for specifying options for data encoding visualization. NeuCube-M1 in v1.1 supports visualization of raw and encoded data for one feature and one sample.

   For DEMO purpose, default values can be used. Pressing 'OK' encodes the data and after completion, the data encoding is visualized in the visualization panel as shown in Figure 9. The graph on top shows the raw input data for the chosen sample and feature and the bottom graph shows the positive and negative spike train generated from the raw data.

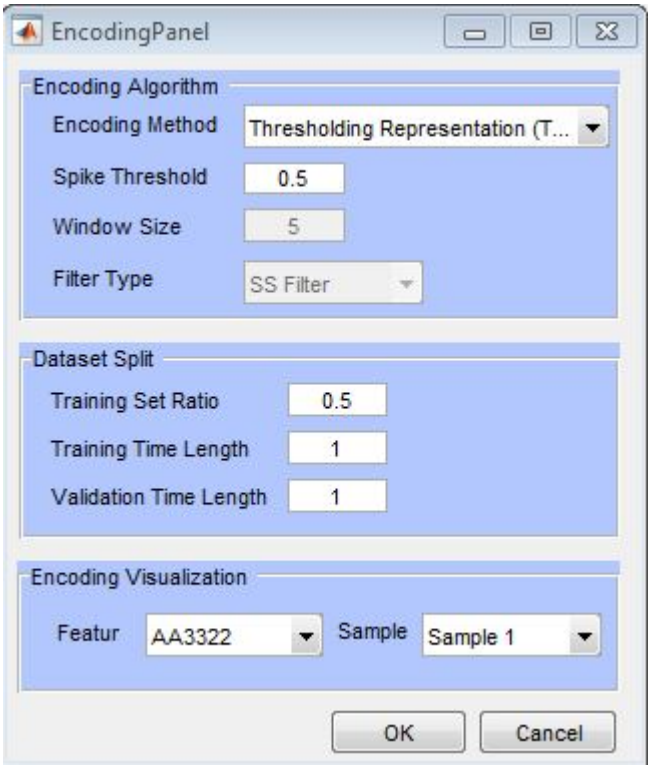

Figure 8. Encoding panel

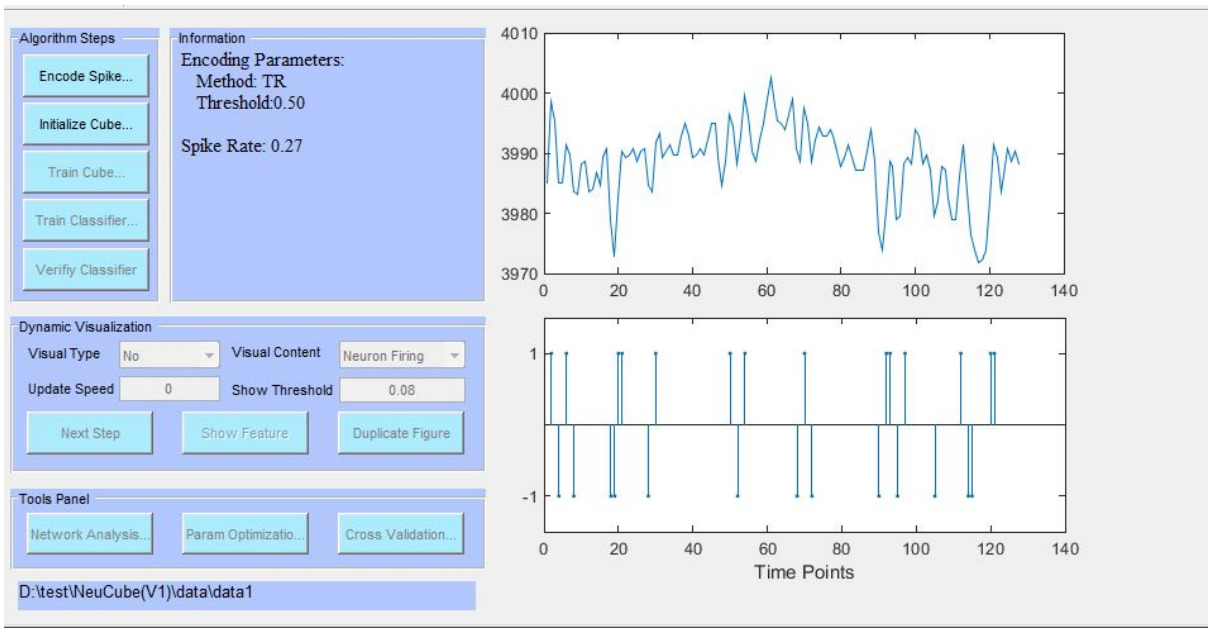

Figure 9: Spike encoding visualization

Cube Initialization

Once the data is encoded, the next step is to initialize the SNNcube, which can be done by clicking the 'Initialize Cube' button. This evokes the cube initialization panel as shown in Figure 10. The subpanel highlighted in red corresponds to the properties of the neurons of the cube. Coordinates of the neurons in the SNNcube can be defined automatically (using a graph method), manually (by specifying the neuron number and its x, y and z coordinates), or by loading a map from a file. This is allowed in the 'Neuron coordinates' drop down. The subpanel highlighted in blue corresponds to the properties of the input neurons, where the spike trains for each input variable

are entered for unsupervised training of the SNNcube. Coordinates of the input neurons can be mapped automatically, loaded from a file, or input manually by using the 'given by' option.

The connection between the neurons in the SNNcube are initialised using the small-world connectivity approach, where a radius is defined as a parameter for connecting neurons within this radius (SWC parameter) with small weight values attached to the connections which are 80% positive. Another parameter LDC (long distance connectivity) can be used to initialise connections beyond the radius of SWC.

In this DEMO we will load a mapping for the reservoir neurons, by choosing 'load from a file' option and choosing the 'brain_coordinate.csv' through Browsing. Similarly the input neuron mapping is chosen by loading the 'eeg_mapping.csv'. Click 'OK' once the parameters are set for initialization.

Once the initialization is finished, the '3d visualization' panel shows the initialized cube as shown in Figure 11. Clicking on the 'Show feature' button shows the spatial location of the input neurons in the 3D SNNcube.

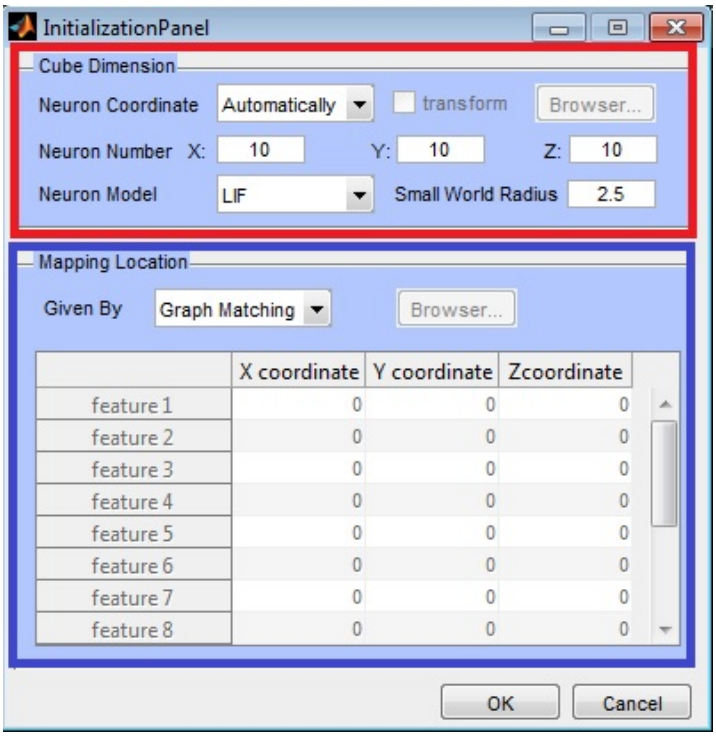

Figure 10. SNNcube Initialization and mapping panel

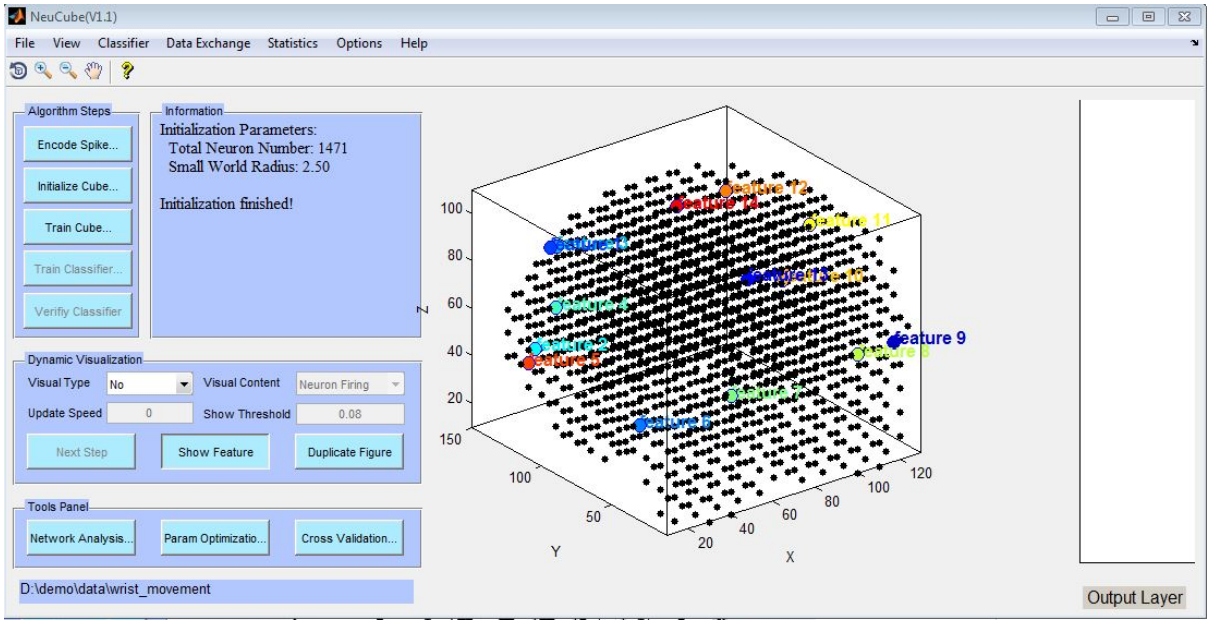

Figure 11. SNN cube after initialisation

Training of the SNN cube

The next step is the unsupervised training of the SNNcube. Clicking on 'Train Cube' initializes the UI for setting hyperparameters for the training as shown in Figure 12. The hyperparameters are explained below in brief:

- Potential leak rate: This parameter defines the leak in membrane potential of a spiking neuron, when the neuron does not fire.
- Threshold of firing: This parameter defines, the threshold membrane potential, beyond which the neuron fires a spike.
- Refractory time: This parameter defines the absolute time (in time units), during which there is no firing of spike. This refactory period begins after a neuron has fired a spike.
- STDP rate: This parameter defines the learning rate of the STDP learning
- Training round: Describes the number of iterations for unsupervised learning in the cube.
- LDC probability: Defines the probability of creating a long distance connection.

Once the hyperparameters for the unsupervised learning are set, click 'OK' to start the unsupervised learning. This step might take significant time based on the size of the dataset and the computer platform used.

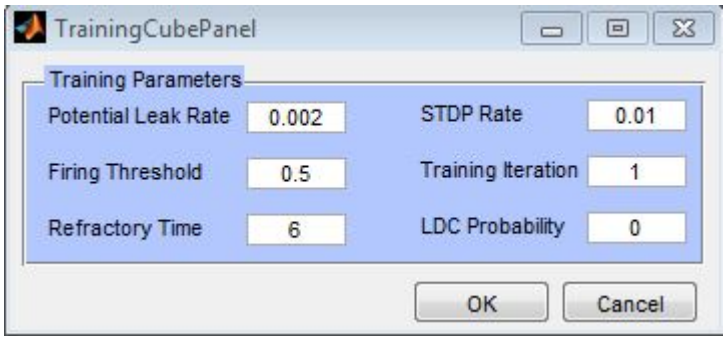

Figure 12. Training SNNcube panel

Dynamic visualization of learning:

The unsupervised learning process can be visualized dynamically online, while the system is learning, or can be saved to a movie for later usage and analysis by using the 'dynamic visualization' panel as shown in Figure 13. The visualization can be rendered in a 'continuous', or 'stepwise' fashion. The 'visual content' dropdown list specifies the type of activity to be rendered like 'Neuron firing', 'Synaptic evolution' or 'Weight changing'. The 'show threshold' option sets the threshold value for the connections to be shown.

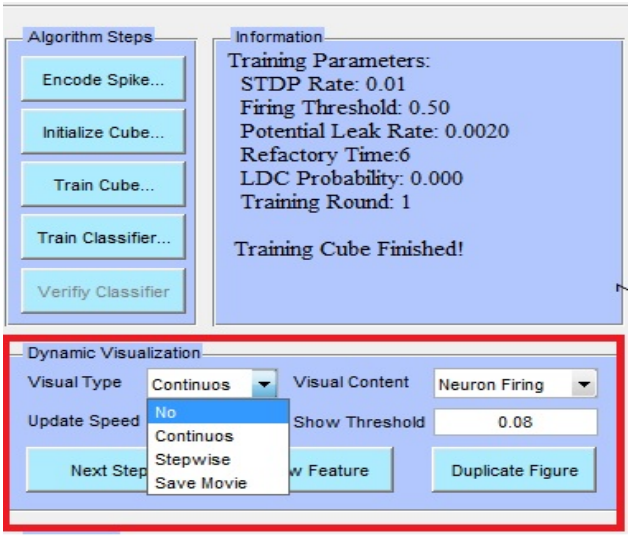

Figure 13. Dynamic visualization panel

Analysis of the cube:

Once the unsupervised learning finishes, there are several options in NeuCube for connectivity analysis and visualization.

Analysis/visualization of the SNNcube connectivity:

The final state of the cube at the end of learning can be analysed and visualized by clicking 'view' in the menu bar (Figure 14).

1. Clicking 'show connection' and choosing a 'threshold' displays the connection above a threshold value as shown in Figure 15 for the DEMO dataset.
2. Clicking 'Activation level' shows the membrane potential of the neurons. Figure 16 shows the spike activation level of the neurons. Detailed explanation about the meaning of each figure can be found on the information panel.
3. Clicking 'Spikes Emitted' shows a histogram of positive and negative spikes emitted by all the SNNcube neurons. Figure 17 shows the spike emission histogram generated for the DEMO dataset.
4. The 'Neuron Weight' option allows user to specifically choose a neuron id and visualize the connection weights of the neurons connected to the chosen neuron.

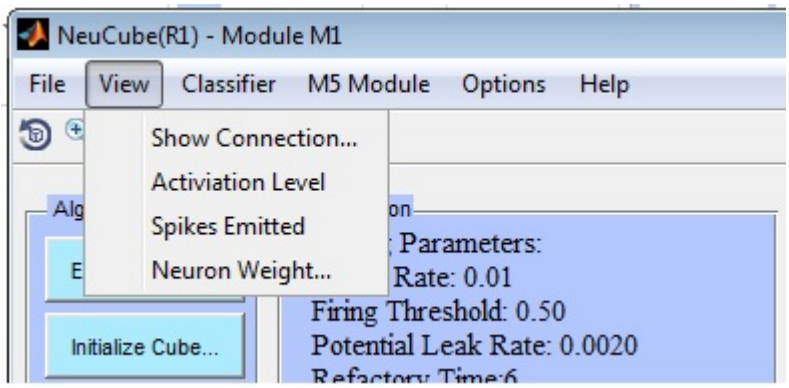

Figure 14. Module M1 visualization menu

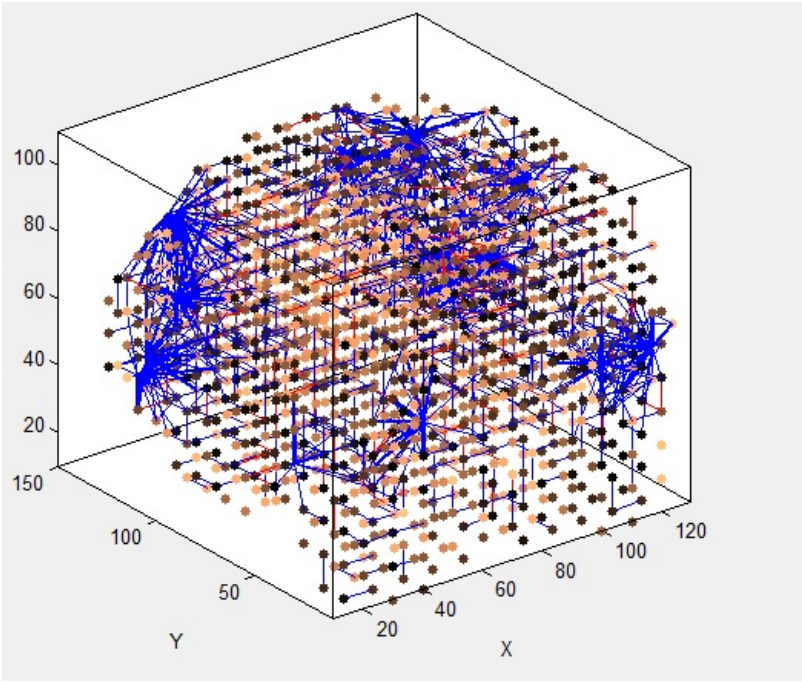

Figure 15. A snapshot of the connnections in the SNNcube after unsupervised learning. The connections represent spatio-temporal relationships between input data variables over time.

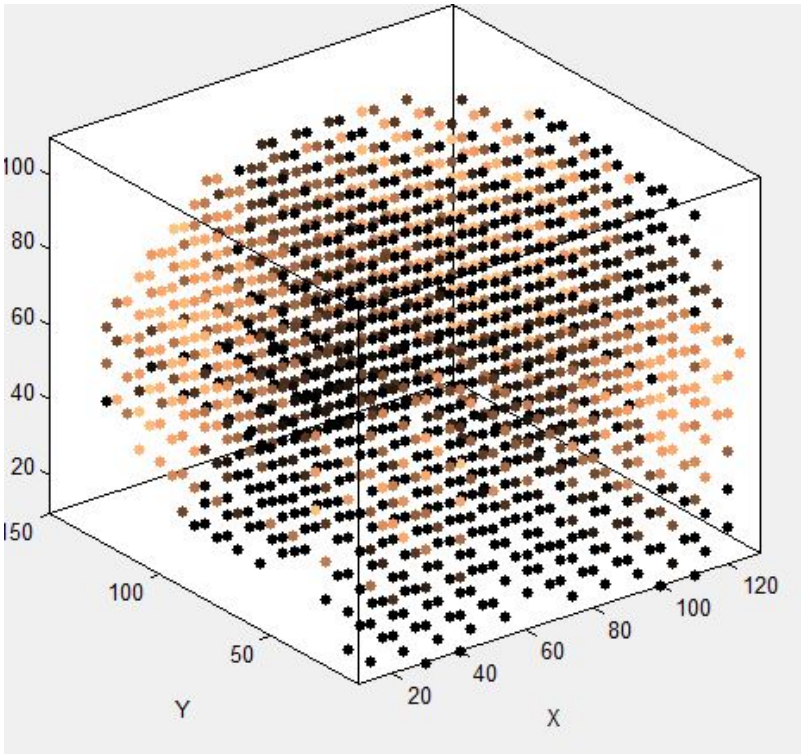

Figure 16. Activation level of the neurons in the SNNcube after unsupervised learning. The brighter the colour of a neuron, the higher its activation level is in terms of number of spikes emitted.

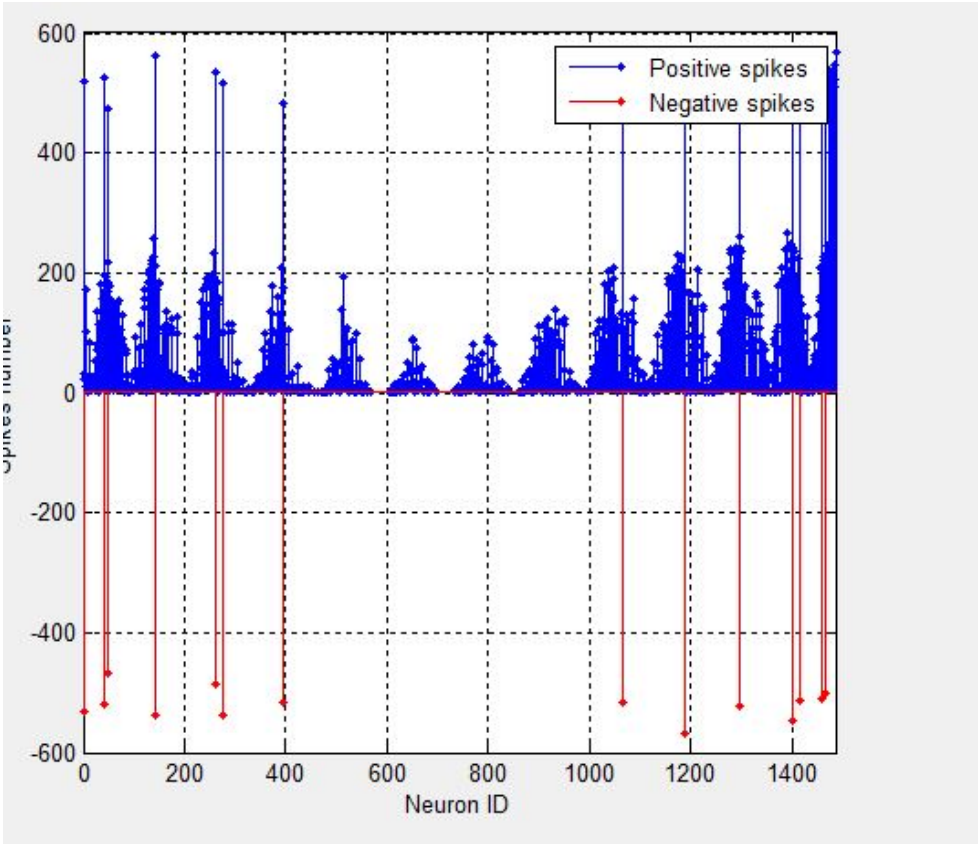

Figure 17: Positive and negative spike emission histogram for all the SNNcube neurons

Analysis through the 'network analysis panel:

Analysis of the learned SNNcube network can be performed using the network analysis toolbox, which is initiated by clicking 'Network analysis' button in the 'tools panel'. Figure 18 shows the UI for the network analysis panel. Network analysis toolbox allows for several kind of analysis, which is described in detail below:

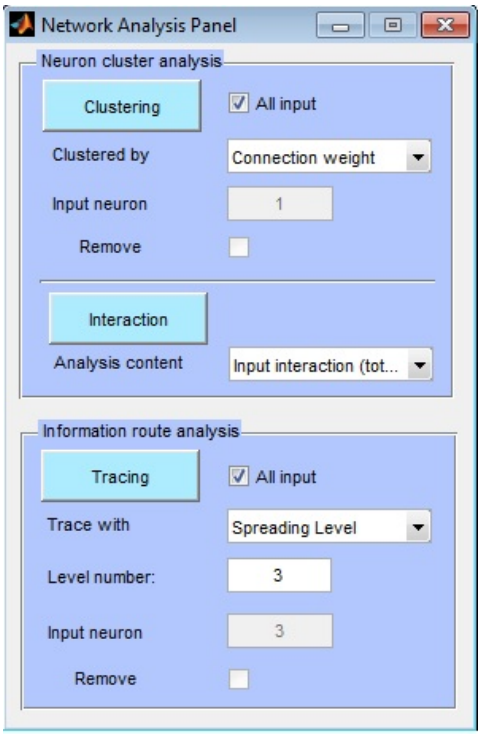

Figure 18: Network analysis panel

- **Neuron cluster analysis**: This option is used to analyse clusters of neuron surrounding input neurons. The clustering of the neurons can be done by two methods.
    - Connection weights[1]: For example Figure 19 shows the clustering by 'connection weights' for the learned cube with the EEG dataset. This is achieved by choosing 'connection weights' from the 'clustered by' option and clicking on the 'clustering' button.

[1] Connection weight is the synaptic weight between a pair of neuron. It is adjusted during unsupervised learning to reflect the interaction between the neurons

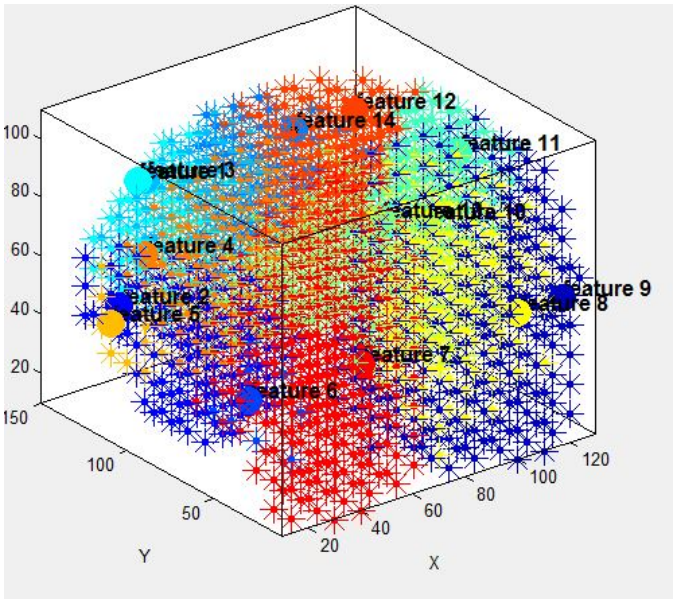

Figure 19. Clustering by connection weights for each input variable that represents a cluster centre.

- Spike communications[2]: For example Figure 20 shows the clustering by 'spike communications' for the learned cube with the exemplar EEG dataset. This is achieved by choosing 'spike communication' from the 'clustered by' option and clicking on the 'clustering' button.

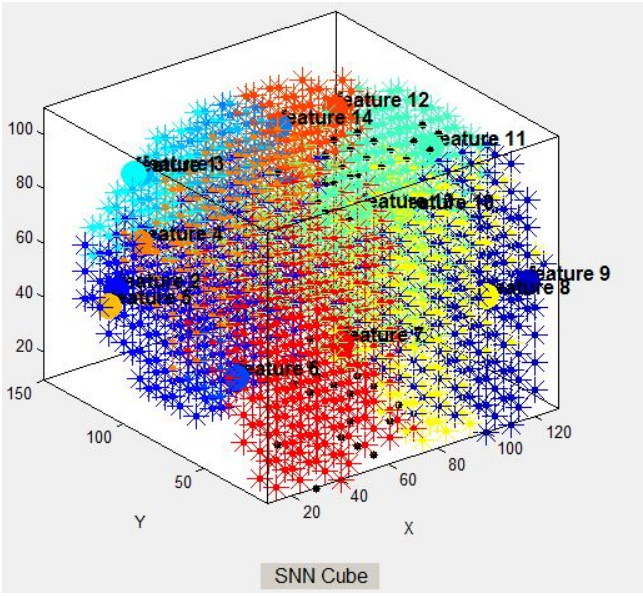

Figure 20. Clustering by spike communication

Once a clustering is performed interaction between the neurons can be performed by choosing the 'analysis content' dropdown and clicking on 'interaction' button. 'Interactions' are analysed using the following metrics:

- Input interaction (total): Shows the total interaction between the input neuron clusters given by the cluster analysis (connection weight/spike communication) explained previously.

[2] Spike communication is the spike amount communicated between a pair of neuron

- Input interaction (average): Shows the average interaction between the input neuron clusters given by the cluster analysis (connection weight/spike communication) explained previously.
- Neuron proportion: Shows the percentage of neurons in the cube which belong to an input neuron cluster.

For example Figure 21 shows the average one to one interaction between the input neurons based on average input interaction. The thicker lines signifies more interaction.

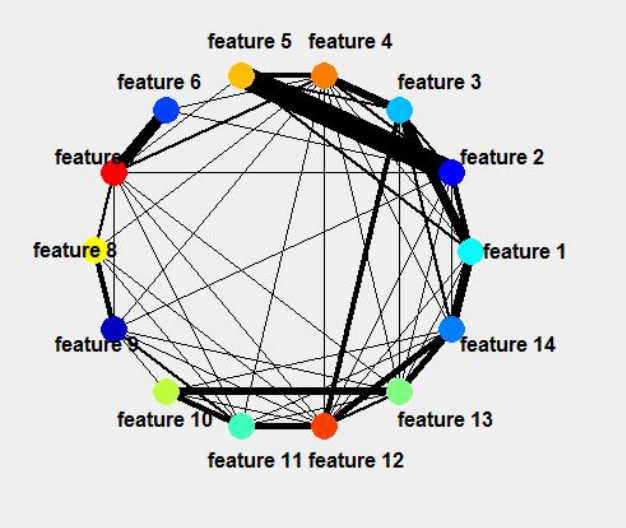

Figure 21. Spike interaction based on 'average input interaction'

Information route analysis: This option is used for analysing information propagation route of the spikes. This analysis is based on the concept of 'rooted tree'. A rooted tree is defined as a directed tree having a single root node (neuron). Figure 22 shows an example of a rooted tree. The neuron in yellow, is the root neuron. A neuron's 'parent neuron' is defined as a neuron which is one step higher in hierarchy and lying on the same branch. For example, the 'parent neuron' of any green neuron is the 'red neuron' it is connected to. Similarly a 'child neuron' is defined as a neuron which is one step lower in hierarchy and lying on the same branch. For example, the 'child neuron' of any red neuron is the 'green neuron' it is connected to.

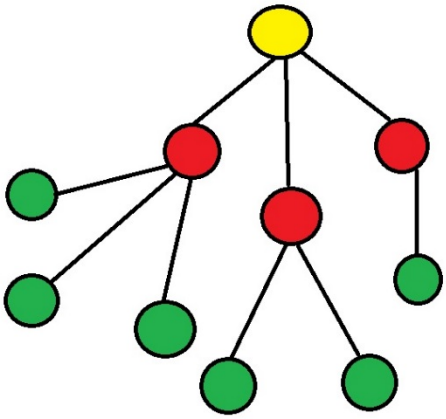

Figure 22: Rooted tree structure

The type of information can be chosen by selecting the 'trace with' drop down. Different methods of analysis is described below in brief:

- Max spike gradient: This visualization shows a tree rooted by the input neuron, where a child neuron is chosen to be connected to a parent neuron, if it receives spike from its parents.
- Spreading level: This visualization shows a tree from the input neuron to its neighbourhood which reflects the spreading of the spikes. The 'level number' parameter defines the neighbourhood of spread. For example choosing 'level number' as 2, means the figure shows spike distribution from input neuron to two layers of neighbouring connected neurons. Figure 23 shows an example of spreading visualization for input neuron 3 spread up to level 3.

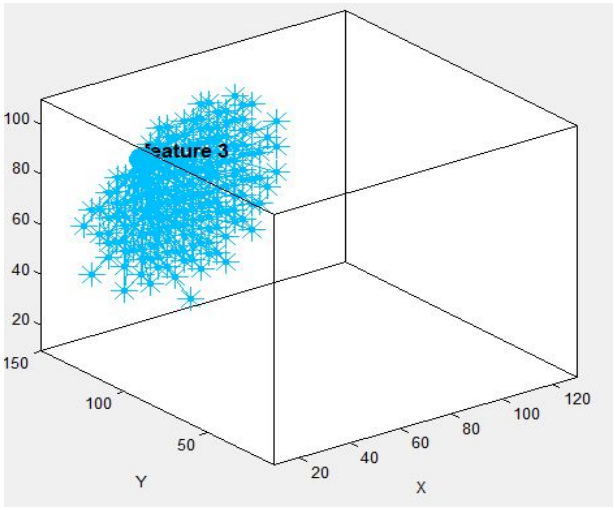

Figure 23. Information route analysis traced by 'spreading level' for input neuron number 3

- Information amount: This visualization shows a tree rooted by the input neuron, where a child neuron is chosen to be part of the tree, only if it receives a defined percentage of spikes from its parent neuron. The percentage can be specified by the 'information' box, where 0.1 means 10% spikes. Figure 24 shows a neuronal cluster from input neuron number 5, where every child neuron has received at least 10% of spikes from the parent neuron.

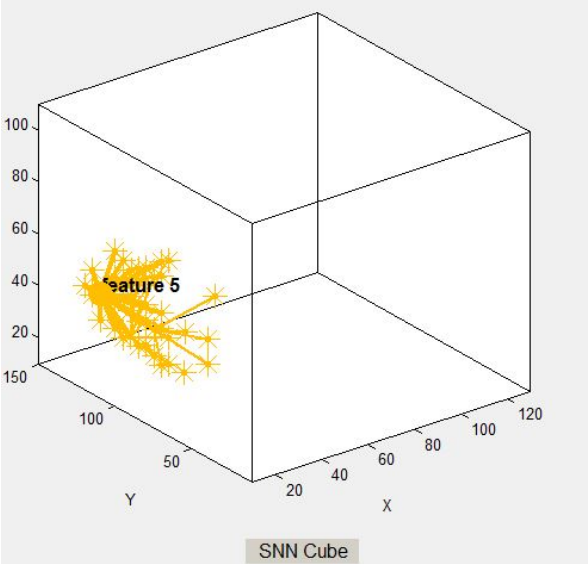

Figure 24. Information amount cluster for feature (input neuron) 5.

Training classifier:

This step trains a model that takes the output spikes of the trained SNN cube as input and performs supervised learning to perform classification and regression. The UI for training classifier is initiated by 'Train classifier' button. Figure 25 shows the 'training classifier' panel, which is used to choose the classification algorithm and corresponding hyper-parameters. For this DEMO the default parameter set is kept. Click 'OK' to start supervised learning. One may use the dynamic visualization panel controls to dynamically visualize the learning as discussed in page 17.

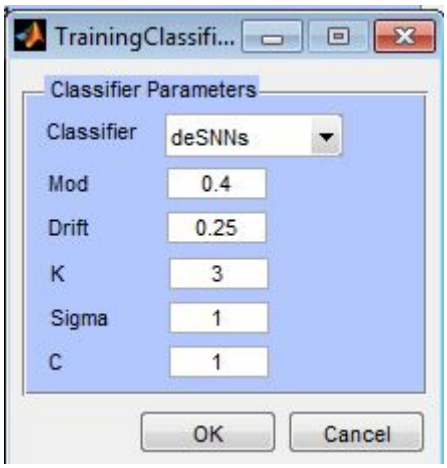

Figure 25. Classifier/regressor training panel

Verify classifier

This step is used to verify the accuracy of the model built by deSNN learning. Clicking on 'Verify classifier' begins the verification procedure. At the end of verification the output result is visualized graphically as shown in Figure 26. The 2D plot is a sample id vs class label plot, where the legends describe the true and predicted class labels. The 'result information' table lists down sample-wise Truth and prediction label. Overall and class-wise accuracies are shown at the end. The true and predicted class labels can be exported to a csv file by clicking 'Export Results'.

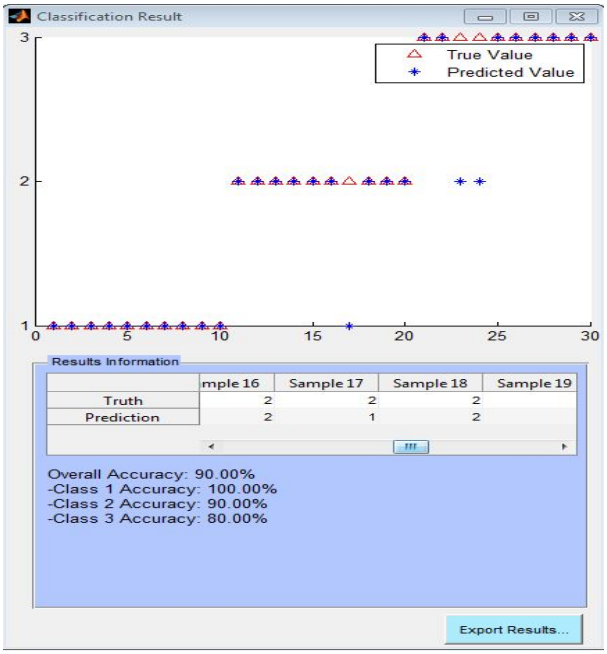

Figure 26. Output classification result panel for the verification of the classifier on the testing data.

Output layer visualization:

The output layer is shown in 'output layer visualization panel' which is described in page 9. The output layer consists of a layer of neurons, where each neuron corresponds to a sample. The output neurons are ordered from bottom to top across columns. For example in Figure 28 the first sample corresponds to the first blue neuron at the bottom, and the last sample corresponds to the top red neuron. The 'output layer visualization' panel can be controlled from the 'classifier' tab from the menu bar as shown in Figure 27. Each option under 'classifier' tab is described below:

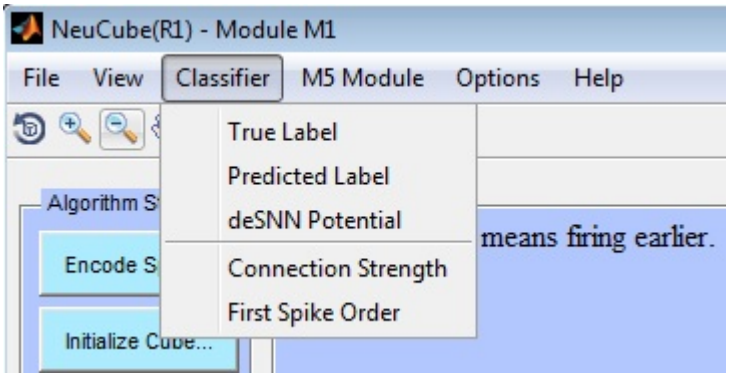

Figure 27. Output layer visualization control

- *True label*: Clicking on true label displays the true label of each sample from the training data set in a colour coded way. Figure 28 show the true labels for the EEG dataset, where red is true class 1, green is true class 2 and blue is true class 3.

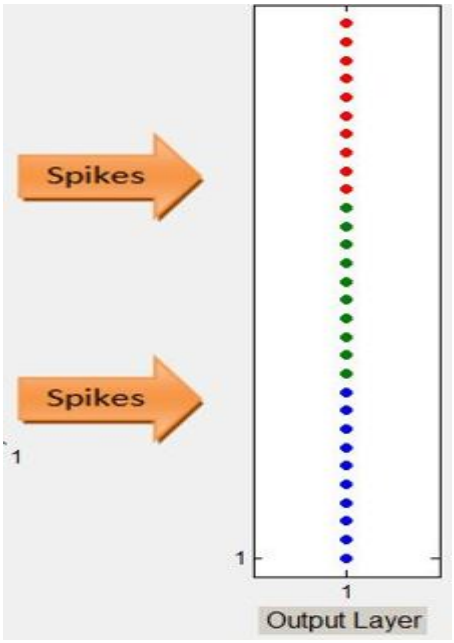

Figure 28. True labels shown in output layer

- *Predicted label*: Clicking on predicted label displays the predicted label of each sample from the test/validation data set in colour coded fashion. Figure 29 shows the predicted labels for the EEG dataset, where red is predicted class 1, green is predicted class 2, and blue is predicted class 3.

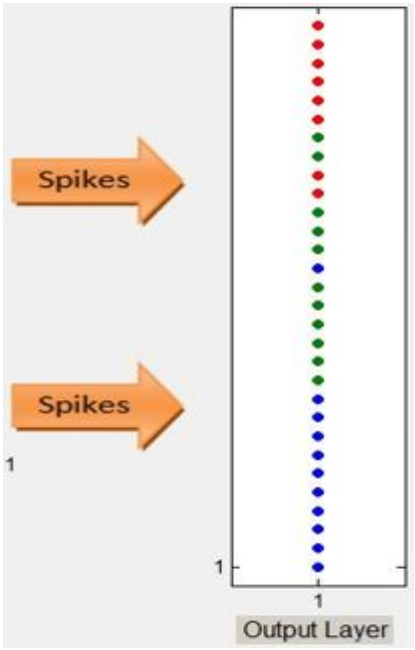

Figure 29. Classification output results after testing the trained classifier. In the DEMO we used 50% of the data (30 samples, 10 of each class) for training and 50% for testing (This was selected in the initialisation menu - Train Set Ratio parameter) . It is seen that 2 test samples that belong to class 1 (red) are wrongly classified into class 2 (green).

- *deSNN potential*: Clicking on deSNN potential displays the membrane potential of the output neuron (sample). As shown in Figure 30 brighter neuron signifies higher potential.

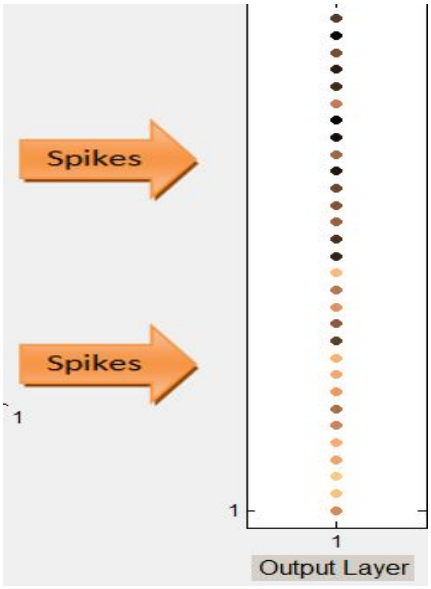

Figure 30. deSNN membrane potential of each output neuron (in this case study they are 30, 10 for each class) in the output layer after supervised training

- *Connection strength*: clicking on 'connection strength' enables the user to visualize the strength of connections between the SNN cube neurons for every output neuron (sample). As shown in Figure 31, clicking on one of the neurons in the output layer shows the connection strength of the neurons in the cube for that particular output neuron (sample). Brighter neurons are more strongly connected.

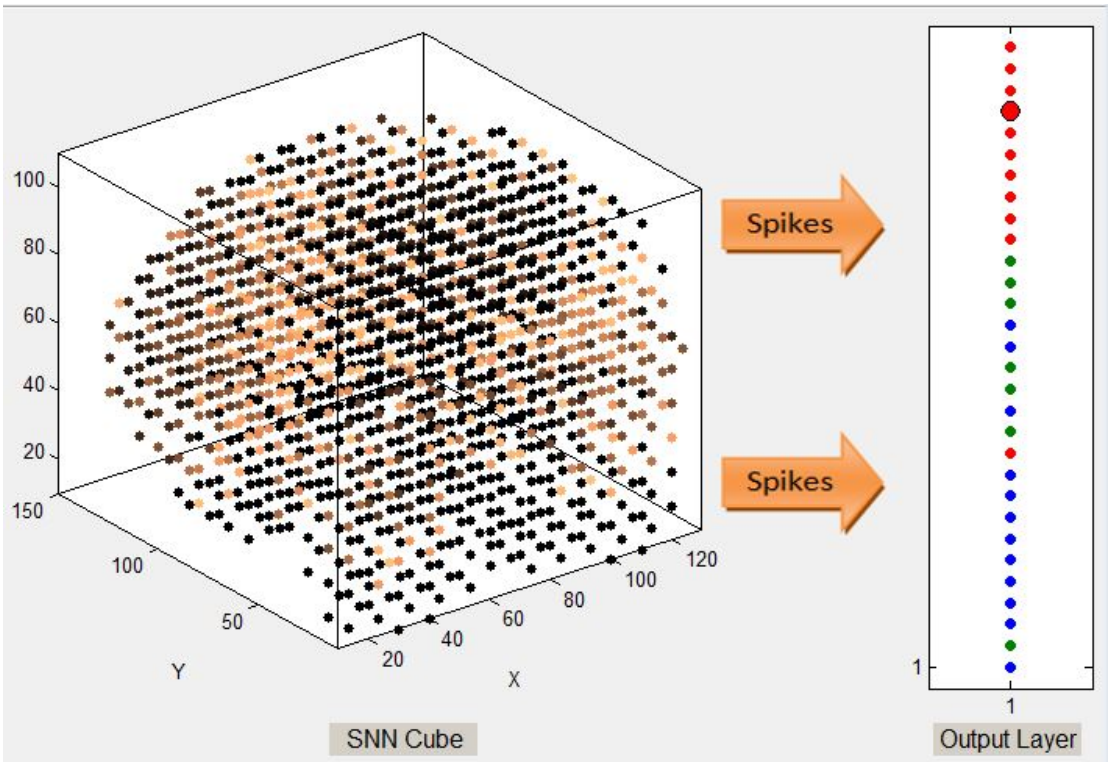

Figure 31: Connection strength for the fourth sample (highlighted by the big neuron in the output layer)

- *First spike order*: clicking on 'first spike order' enables the user to visualize the spiking order of the neurons in the SNN cube for each output neuron (sample). As shown in Figure

32, clicking on one of the neurons in the output layer shows the firing order of the neurons in the cube for that particular output neuron (sample). Brighter neurons fire earlier.

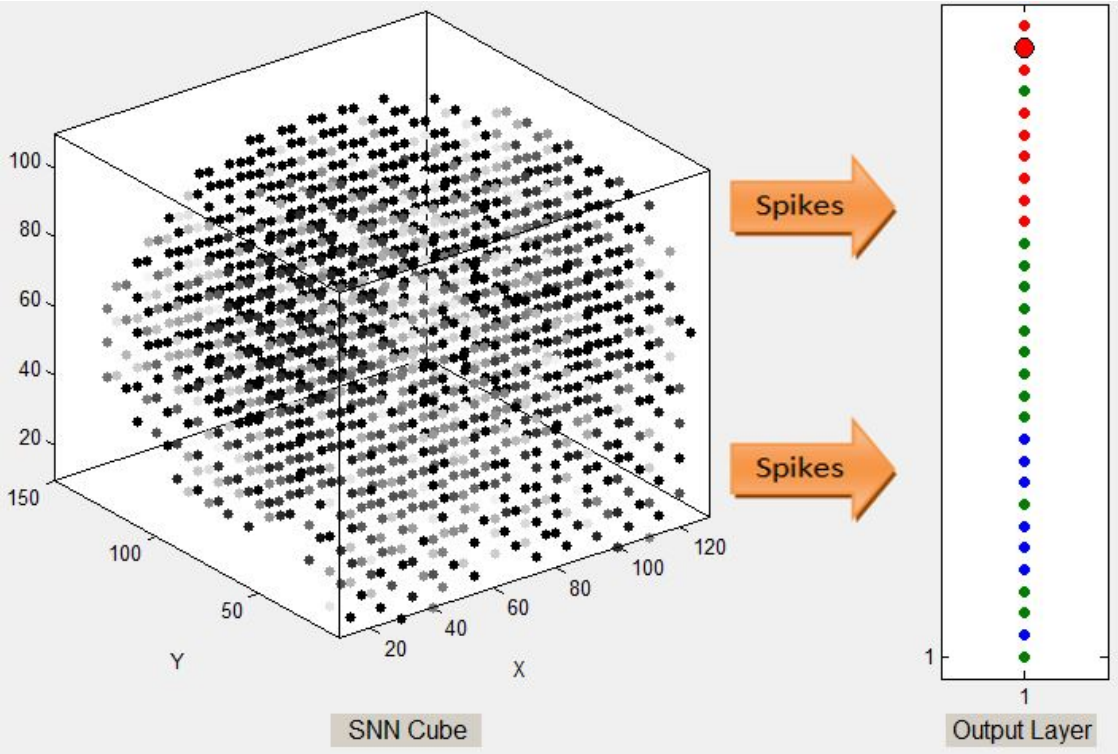

Figure 32. Firing order for the second sample (highlighted by the big neuron in the output layer)

Cross validation and Parameter optimization:

Cross validation: cross validation is a function which is wrapped around the unsupervised and supervised learning. At every fold the cube is initialised, trained unsupervised and trained supervised with different combination of data. Cross validation can be performed by clicking on the 'cross validation button' in the 'tool panel'. This initiates the UI as shown in Figure 33. The fold number parameter defines the number of iterations of training and validation.

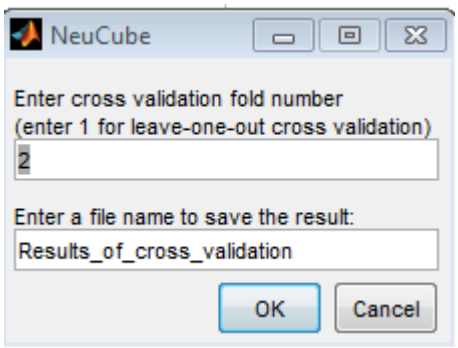

Figure 33. Cross validation UI

Parameter optimization: Parameter optimization toolbox allows to search for optimal set of hyperparameters that minimizes the test accuracy of the model (either for classification or for regression). The computational time for parameter optimisation depends on the number of parameters to be optimised and the size of the NeuCube model. Clicking on 'Param Optimization' initializes the UI for parameter optimization (Figure 34).

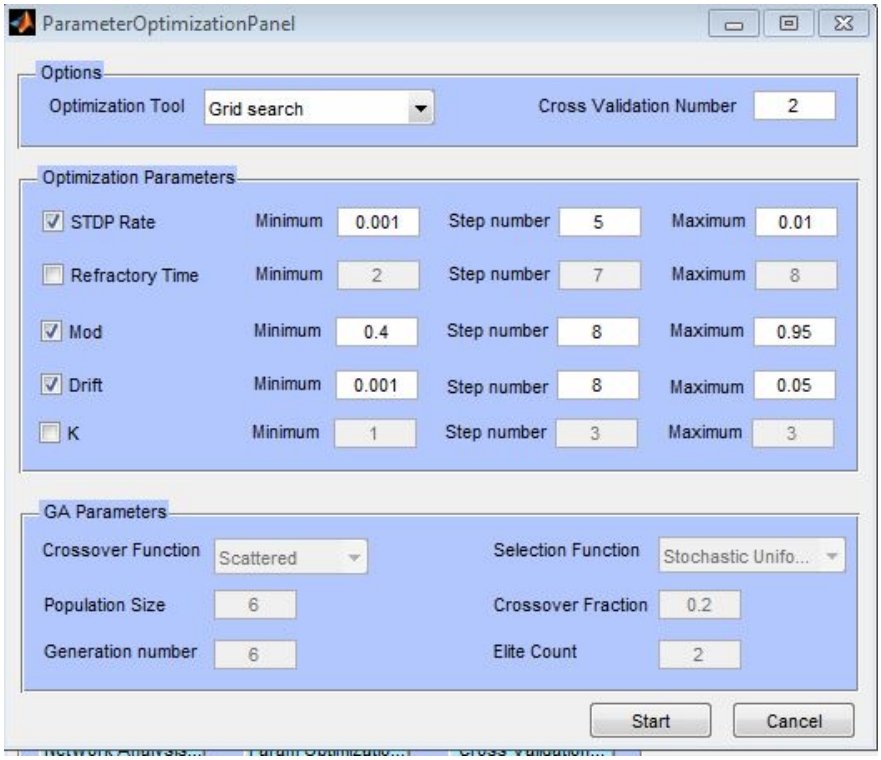

Figure 34. Parameter optimization panel

Parameter optimization in NeuCube Module M1, can be performed using various methods, such as: Grid search; Genetic Algorithm; Differential Evolution; Quantum Inspired Evolutionary Algorithms, PSO, etc. In NeuCube v1.1, two methods (Grid search and Genetic algorithm) are implemented and can be chosen from 'Optimization Tool' dropdown. NeuCube v1.1 offers 5 parameters for optimization, however the later release will include more parameters for optimization:

1. **Exhaustive grid search**: This is an exhaustive search method, based on grid based combination of parameters. The 'Optimization parameters' subpanel can be used to specify the parameters to be optimized by checking on the checkboxes. For example in Figure 34 STDP Rate, Mod and Drift is chosen to be optimized. Each parameter is searched within a range, specified by the 'Minimum' and 'Maximum' value. The 'step number' specifies the number of steps to be used for moving from minimum to maximum. Once these parameters are set, clicking 'Start' begins the parameter optimization by grid search.
2. **Genetic algorithm**: This is a nature inspired algorithm to search for optimal parameters of NeuCube. Choosing 'Genetic Algorithm (GA)' from 'optimization method' dropdown enables the user to choose the parameters for the genetic algorithm. As described in the exhaustive grid search, the parameters can be chosen by checking the checkboxes, and bounds of the parameters can be set up using 'maximum' and 'minimum' values. Contrary to exhaustive search GA is not a fixed step approach and does not require the step number to be specified. GA parameters that can be used to modify the behaviour of GA are described below in brief:
    - Crossover function: Crossover options specify how the genetic algorithm combines two individuals, or 'parents', to form a crossover 'child' for the next generation. An individual represents the set of parameters (called here genes) of a NeuCube model (called a 'chromosome').

  - Scattered: creates a random binary vector and selects the parameters (genes) where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form a child's chromosome.
  - Single point: chooses a random integer n between 1 and Number of variables and then,
    - Selects vector entries numbered less than or equal to n from the first parent.
    - Selects vector entries numbered greater than n from the second parent.
    - Concatenates these entries to form a child's chromosome vector.

  - Double point: chooses a random integer n between 1 and Number of parameters and then:
    - Selects vector entries numbered less than or equal to n from the first parent.
    - Selects vector entries numbered greater than n from the second parent.
    - Concatenates these entries to form a child vector.
- Selection function: Selection function specifies how the genetic algorithm chooses parents for the next generation.
  - Stochastic uniform: lays out a line in which each parent corresponds to a section of the line of length proportional to its scaled value. The algorithm moves along the line in steps of equal size. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size.
  - Remainder: Remainder selection assigns parents deterministically from the integer part of each individual's scaled value and then uses roulette selection on the remaining fractional part.
  - Uniform: Uniform selection chooses parents using their evaluated fitness value (e.g. the classification accuracy of the NeuCube model). Uniform selection is useful for debugging and testing, but is not a very effective search strategy.
  - Roulette: Roulette selection chooses parents by simulating a roulette wheel, in which the area of the section of the wheel corresponding to an individual is proportional to the individual's expectation. The algorithm uses a random number to select one of the sections with a probability equal to its area.
  - Tournament: Tournament selection chooses each parent by choosing Tournament size players at random and then choosing the best individual out of that set to be a parent.
- Population size: specifies how many individuals (NeuCube models) will be created and tested for fitness at each generation. With a large population size, the genetic algorithm searches the solution space more thoroughly, thereby reducing the chance that the algorithm returns a local minimum that is not a global minimum. However, a large population size also causes the algorithm to run more slowly.
- Generation number: Specifies the maximum number of generations for the genetic algorithm to perform.
- Crossover fraction: specifies the fraction of the next generation, other than elite children, that are produced by crossover.

- Elite count: specifies the number of individuals that are guaranteed to survive to the next generation. Set Elite count to be a positive integer less than or equal to the population size.

Figure 35 shows an example plot of fitness value (error) vs number of generations. It can be clearly seen how the model accuracy increases with time (generation).

Once the optimization method and parameters are chosen clicking 'Start' initiates the parameter optimization run. During the process, running time and estimated time remaining for the optimization can be visualized in information panel as shown in Figure 36. This is dynamically updated periodically. At the end of parameter optimization, the best parameters are stored as a Matlab file inside the local 'result' folder. This parameter can be used and loaded into the NeuCub-M1 environment later.

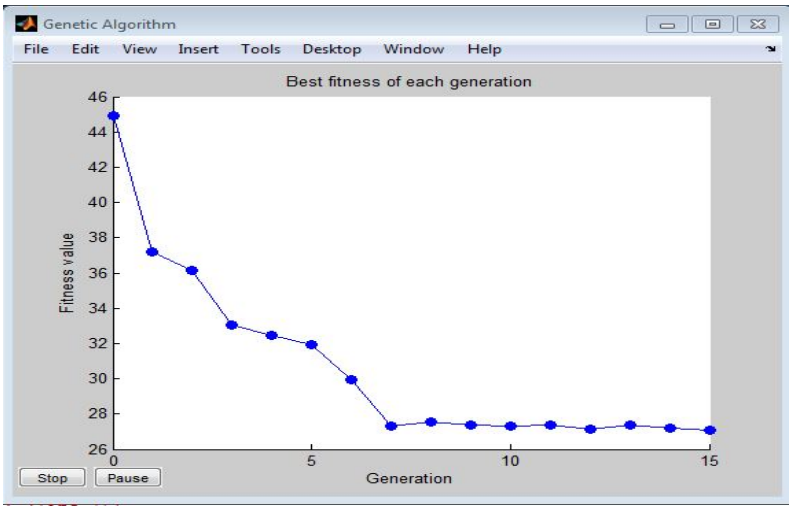

Figure 35. Example of parameter optimisation of a NeuCube model for classification using a GA optimiser. The classification error of the model (related to fitness) is decreasing with every generation of the GA where different parameter values are selected for the model.

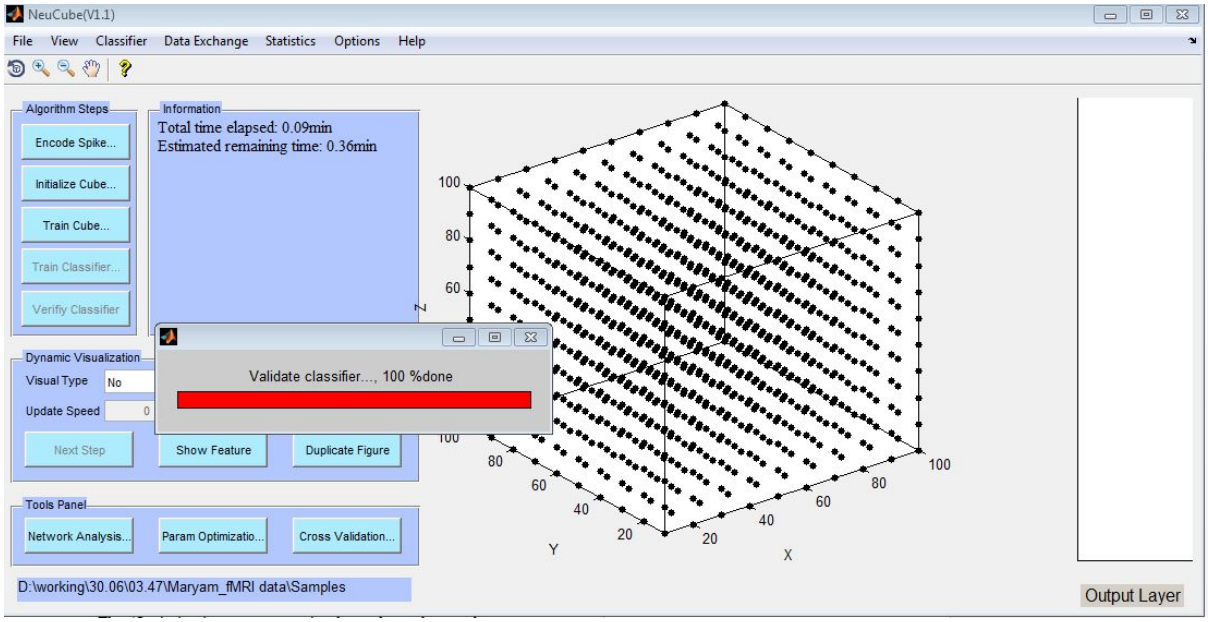

Figure 36: Example of parameter optimization running in NeuCube-M1. The information panel shows the total time elapsed and estimated remaining time for the optimization

Exporting statistics and results:

It is possible to export some information as numbers for further analysis by clicking on 'Statistics' dropdown in menu bar as shown in Figure 37. It is possible to download SNNcube connection weights, Output connection weights, SNNcube neuron activation levels, and spike emission statistics by clicking 'Cube Weight', 'Outlayer Weight', 'Activation Level' and 'Spike Emitted' respectively. The statistics are exported to csv files.

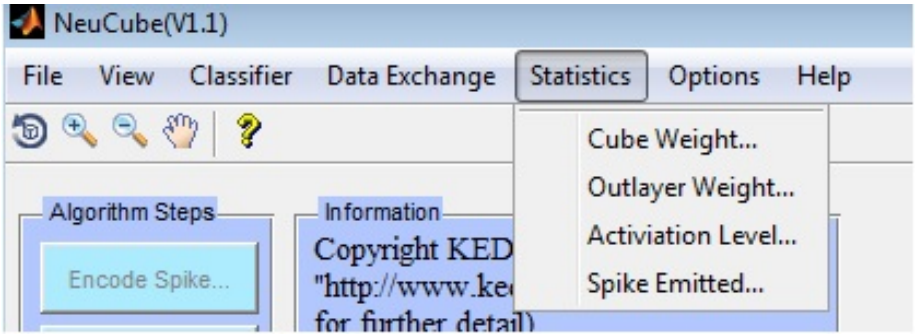

Figure 37: Statistics options

6. EXAMPLE ON REGRESSION ANALYSIS

A DEMO dataset for regression analysis can be found in data>share_price folder. This dataset consists of 50 samples. Each sample consists of 100 timed sequence of daily closing price of 6 different shares (Apple Inc., Google, Intel Corp, Microsoft, Yahoo, and NASDAQ). The target values representing the closing price of NASDAQ at the next day, are arranged in a column in the target file. For data (like the financial dataset) which does not have any natural spatial ordering, NeuCube automatically assigns spatial location based on graph matching algorithm. Hence this dataset do not require any additional coordinate files. Data can be loaded from File>Load data>Regression and choosing the financial_dataset folder. Once the data is loaded, algorithmic steps can be performed stepwise, as discussed in the previous section. For initialization of the SNNcube, use 'Automatically' and 'Graph matching' options from the 'neuron coordinate' and 'given by' dropdown respectively, as shown in Figure 38.

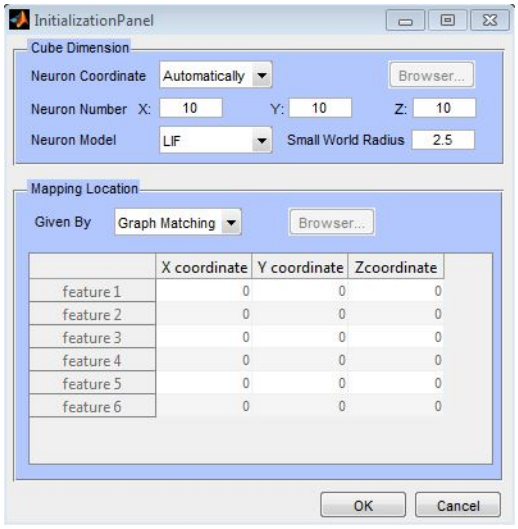

Figure 38: Initialization options for regression DEMO

Figure 39 shows the regression result produced by NeuCube-M1 on the DEMO regression dataset. The graph plots the true and predicted value of the validation samples. It also provides Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), as a measure of performance on the validation set.

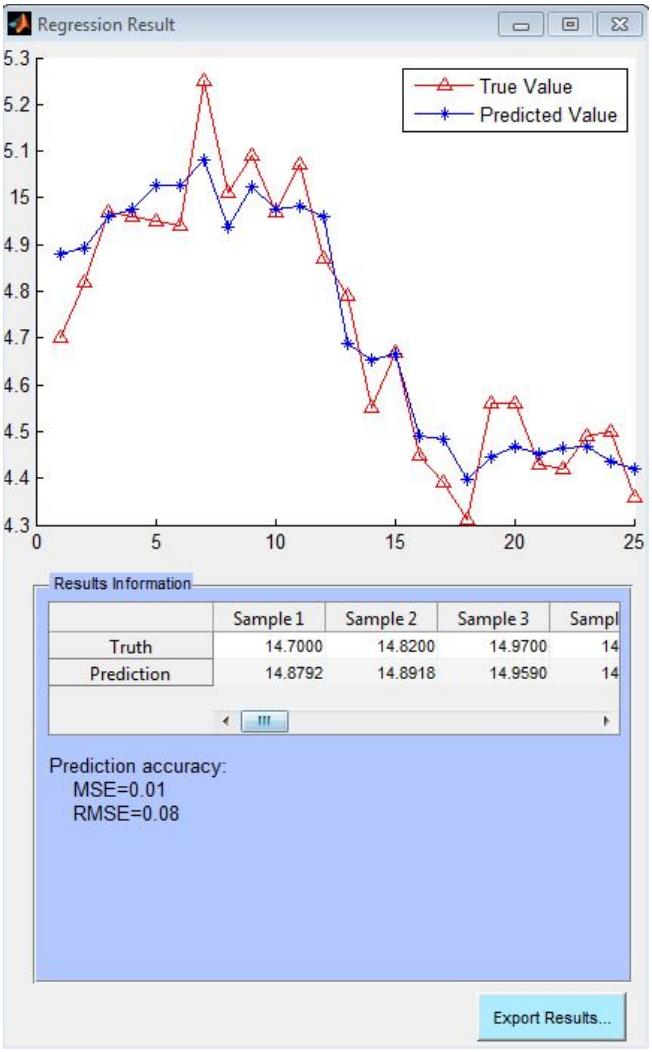

Figure 39: Regression result panel

7. RECALL

NeuCube-M1 module can be used to perform recall on new samples. For the purpose of recall, following resources are necessary:

1. Data Folder: A dataset for recall should only have a set of samples in one folder. If a target file is present in the folder, the software ignores it.
2. Cube file: Recall requires a cube file. The cube file contains description of the NeuCube model. A cube file can be exported from 'file>save NeuCube' option after 'training classifier'.
3. Parameter file: Recall requires the parameter file used during training of the model. A parameter file can be exported from 'file>save Parameter' option after 'training classifier'.

A recall operation can be performed in following steps:

1. Load the recall data from 'file>Load Data>Recall'.
2. Once the dataset is successfully loaded, perform spike encoding step by clicking 'Encode Spike'.
3. Load the cube and the parameter file saved previously, by clicking 'file>Load NeuCube' and 'file>Load Parameter' respectively.
4. Click on 'Verify Classifier' to predict the output on the recall samples.

8. REFERENCES

1. N.Kasabov et al, Design methodology and selected applications of evolving spatio- temporal data machines in the NeuCube neuromorphic framework, Neural Networks, 2015
2. Kasabov, N. Evolving connectionist systems for adaptive learning and knowledge discovery: Trends and Directions, Knowledge Based Systems, 2015, (2015), http://dx.doi.org/10.1016/j.knosys.2014.12.032.
3. Elisa Capecci, Grace Y. Wang , Nikola Kasabov, Analysis of connectivity in a NeuCube spiking neural network trained on EEG data for the understanding and prediction of functional changes in the brain: A case study on opiate dependence treatment, Neural Networks, (2015), http://dx.doi.org/10.1016/j.neunet.2015.03.009.
4. Maryam Gholami Doborjeh, Grace Y. Wang, Nikola Kasabov, A Neucube Spiking Neural Network Model for the Study of Dynamic Brain Activities during a GO/NO_GO Task: A Case Study on Using EEG Data of Healthy Vs Addiction Treated Subjects, IEEE Trans. NNLS, submitted, 2015.
5. Nikola Kasabov, Maryam Gholami Doborjeh, Spatio-Temporal Brain Data Mining with a NeuCube Evolving Spiking Neural Network Model on the fMRI Case study, IEEE Transactions of Neural Networks and Learning Systems, submitted 2015.
6. Enmei Tu, Nikola Kasabov, and Jie Yang, Mapping Temporal Variables into the NeuCube Spiking Neural Network Architecture for Improved Pattern Recognition, Predictive Modelling and Understanding of Stream Data, IEEE Transactions of Neural Networks and Learning Systems, submitted, 2014.
7. Kasabov, N., E.Capecci, Spiking neural network methodology for modelling, classification and understanding of EEG spatio-temporal data measuring cognitive processes, *Information Sciences*, 294, 565-575, 2015, DOI: 10.1016/j.ins.2014.06.028, 2014..
8. Kasabov, N. NeuCube: A Spiking Neural Network Architecture for Mapping, Learning and Understanding of Spatio-Temporal Brain Data, *Neural Networks* vol.52 (2014), pp. 62-76, http://dx.doi.org/10.1016/j.neunet.2014.01.006
9. Tu, E., Cao, L., Yang, J., & Kasabov, N. (2014). A novel graph-based k-means for nonlinear manifold clustering and representative selection. Neurocomputing. doi:10.1016/j.neucom.2014.05.067
10. Kasabov, N., Feigin, V., Hou, Z. -G., Chen, Y., Liang, L., Krishnamurthi, R., Parmar, P. (2014). Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke. Neurocomputing, 134, 269-279. doi:10.1016/j.neucom.2013.09.049
11. N. Murli, N. Kasabov, and B. Handaga, Classification of fMRI Data in the NeuCube Evolving Spiking Neural Network Architecture, Proc. ICONIP 2014, Springer LNCS, 2014..
12. M. G. Doborjeh, E. Capecci and N. Kasabov, Classification and Segmentation of fMRI Spatio-Temporal Brain Data with a NeuCube Evolving Spiking Neural Network Model, Proc. SSCI, IEEE Press, 2014.
13. E. Tu, N. Kasabov, M.Othman, Y. Li, S.Worner, J.Yang and Z. Jia, NeuCube(ST) for Spatio-Temporal Data Predictive Modelling with a Case Study on Ecological Data, Proc. WCCI 2014, Beijing, 7-13 July 2014, IEEE Press.
14. D. Taylor, N.Scott, N. Kasabov, E.Capecci, E. Tu, N. Saywell, Y. Chen, J.Hu and Z.Hou, Feasibility of NeuCube SNN architecture for detecting motor execution and motor intention for use in BCI applications, Proc. WCCI 2014, Beijing, 7-13 July 2014, IEEE Press.
15. M. Othman, N.Kasabov, E.Tu, V. Feigin, R.Krishnamurthi, Z.Hou, Y. Chen and J.Hu, Improved Predictive Personalized Modelling with the use of Spiking Neural Network System and a Case Study on Stroke Occurrences Data, Proc. WCCI 2014, Beijing, 7-13 July 2014, IEEE Press.
16. Hu, J., Hou, Z., Chen, Y., Kasabov, N., & Scott, N. (2014). EEG-Based Classification of Upper-Limb ADL Using SNN for Active Robotic Rehabilitation. In 2014 5th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (pp. 409-414). Sao Paolo, Brazil: IEEE. doi:10.1109/BIOROB.2014.6913811

17. N. Kasabov, J.Hu, Y. Chen, N.Scott, and Y. Turkova, Spatio-temporal EEG data classification in the NeuCube 3D SNN Environment: Methodology and Examples, Proc. ICONIP 2013, Springer LNCS, vol.8228, pp.63-69.
18. Y.Chen, J.Hu, N.Kasabov, Z. Hou and L.Cheng, NeuroCubeRehab: A Pilot Study for EEG Classification in Rehabilitation Practice Based on Spiking Neural Networks, Proc. ICONIP 2013, Springer LNCS, vol.8228, pp.70-77.
19. N. Scott, N. Kasabov, and G.Indiveri, NeuCube Neuromorphic Framework for Spatio-Temporal Brain Data and Its Python Implementation, Proc. ICONIP 2013, Springer LNCS, vol.8228, pp.78-84..
20. N.Kasabov, V.Feigin, Z.Hou, Y.Chen, Improved method and system for predicting outcomes based on spatio/spectro-temporal data, PCT patent, WO2015/030606 A2, priority date: 26.08.2013.
21. N.Kasabov, Data Analysis and Predictive Systems and Related Methodologies – Personalised Trait Modelling System, PCT/NZ2009/000222, NZ Patent, USA Patent 13/088,306, Filed: April 15, 2011, Priority: Sept.2008.
22. Kasabov, N., & Hu, Y. (2010, December). Integrated optimisation method for personalised modelling and case studies for medical decision support. International Journal of Functional Informatics and Personalised Medicine, 3(3), 236-256. doi:10.1504/IJFIPM.2010.039123

9. DEVELOPERS TEAM AND CONTACT PERSONS

Prof. Nik Kasabov, Director KEDRI - overall design of the NeuCube architecture.

Dr. Enmei Tu, Research Fellow - developer of the main NeuCube-M1 module.

Neelava Sengupta (nsengupt@aut.ac.nz) – contact person to report errors in the NeuCube software and the Manual and also developer of module M1, M8 and M9.

Nathan Scott – developer of modules M2 and M3 for neuromorphic implementation.

Dr. Stefan Marks – developer of module M4 for 3D dynamic visualisation in a VR scenario.

Israel Espinosa Ramos – developer of module M6 or neurogenetic modelling.

Elisa Capecci – Co-developer of module M6 for neurogenetic modelling.

Vivienne Breen – Co-developer of module M7 for personalised modelling.

Maryam Gholami Doborjeh (mgholami@aut.ac.nz) – contact person for tutoring on NeuCube and for EEG and fMRI data modelling.

Akshay Raj Gollahalli- Co-developer of Module M10.

Reggio Hartono- Co-developer of Module M10.

Joyce D'Mello (jdmello@aut.ac.nz) – KEDRI Admin Manager

Dr. Enrico Tronchin (etronchin@aut.ac.nz) – AUT Commercialisation Manager

Address for correspondence:

2 Wakefield Street, AUT Tower, 7th floor, KEDRI

Phone: +64 9 9219504;

Website: http://www.kedri.aut.ac.nz/neucube

10. Acknowledgements

The NeuCube development system is funded by Auckland University of Technology SRIF fund and partially by the MBIE of New Zealand for strategic alliance with China. The intellectual property of NeuCube is owned by AUT. Other people who took part in early development of NeuCube and its pilot applications are: Nelson Chen, James Hu, Professors Z. Hou, J. Yang, V. Feigin, D. Taylor, Dr. G. Wang, M. Othman, N. Murli, F. Alvi, M. Fanghella, W. Bhattacharjee, L. Zhou, C. McNabb.