NeuroMem API
Open Source Library

PATTERN LEARNING AND
RECOGNITION
WITH
A NEUROMEM NETWORK

Version 4.1
Revised 05/04/2016

GENERAL
VISION

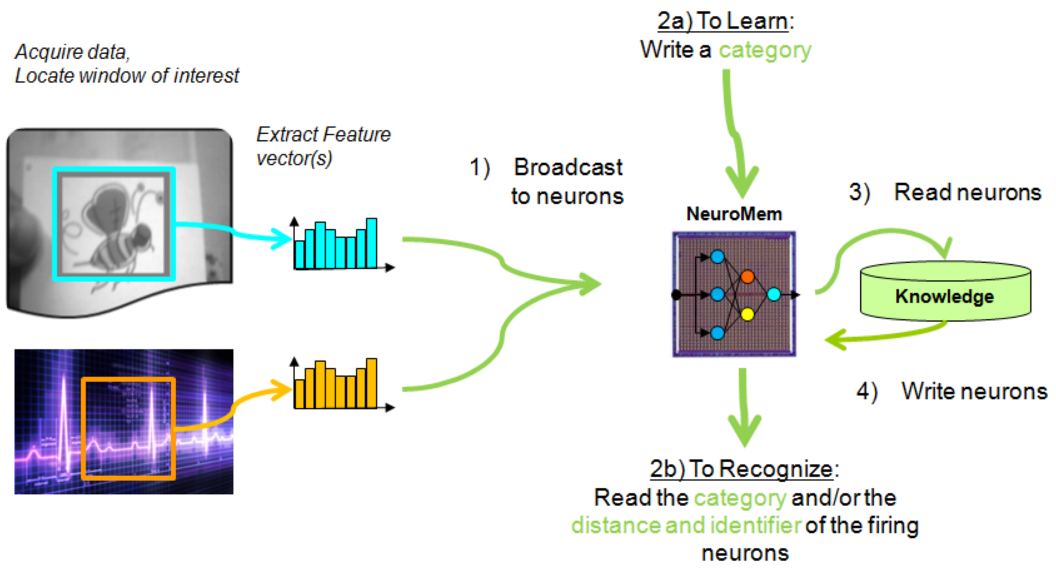
TABLE OF CONTENTS

INTRODUCTION

NeuroMem is a pattern recognition accelerator chip which is also trainable in real-time by learning examples. The interface to a chain of neurons of any size comes down to 4 simple main operations:

- Learn a vector
- Classify a vector
- Save the knowledge built by the neurons
- Load a knowledge into the neurons

Data coming from a variety of sources can be converted into a pattern vectors which are then broadcasted to the neural network for either learning or recognition. In the case of learning, the neurons decide autonomously if the input pattern and its associated category represent novelty and should be stored in the next available neuron, and also if existing neurons with conflicting responses shall correct their firing behavior. In the case of recognition, the neurons decide autonomously which one has the closest match and queue their responses per decreasing level of confidence.



The functions supplied in the NeuroMem API are all based on a Register Transfer Level (RTL) protocol. Higher level functions manipulating agnostic or specific data types can be built on this protocol. Depending on your hardware platform, this RTL protocol can also be used to address other components such as memory, on-board sensors, and more.

For detailed information about the neurons and their behavior, you can also refer to the NeuroMem Technology Reference Guide.

FUNCTION LIBRARY

Init()

Clear the entire content of the neurons (registers and memory). The first neuron of the chain is ready to learn.

void getNeuronsInfo(int* neuronSize, int* neuronsAvailable, int* neuronsCommitted)

Read the specifications of the silicon neurons:

- neuronSize, memory capacity of each neuron in byte
- neuronsAvalaible, number of neurons available
- neuronsCommitted, number of "trained" neurons, that is holding a pattern and its associated category

Forget()
Forget(int Maxif)

Set all the neurons to idle mode by clearing their category register. The first neuron of the chain is ready to learn. Note that this function does not clear the memory of the neurons.

Set the Maximum Influence Field to a value different than the default of 0x4000. The smaller Maxif, the more conservative the generalization capability of the neurons while learning. Maxif can range between 2 and 0xFFFF.

Int(ncount) Learn(byte vector[], int length, int category-to-learn);

Learning consists of broadcasting a vector and assigning its category-to-learn. Depending on the application, the category-to-learn can be defined by the user in a mode called supervised learning. It can also be assigned programmatically in a mode called unsupervised learning. After the broadcast, the firing neurons with a category other than category-to-learn reduce their Influence Fields to exclude the input vector from their similarity domain. This is also when a new neuron is committed if no firing neuron has the category equal to category-to-learn.

Inputs

A vector is a sequence of bytes with a length between 1 and 256. This sequence can derive from any type of data source such as measurements, signal, sound, images, videos, etc. It can be composed of raw data or represent a signature extracted from raw data and also called a feature.

Category can range between 1 and 32767. A category of 0vis used to teach a counter example with the intend to correct neurons firing erroneoulsy, but without commiting a new neuron.

Outputs

The function returns the total number of committed neurons.

Considerations before calling this function:

- Forget to reset the network
- Forget and change the Max Influence Field to learn in a more or less conservative manner

Int(nsr) Classify(byte vector[], int length);
Int(nsr) Classify(byte vector[], int length, int distance, int category, int nid);
Int(respNbr) Classify(byte vector[], int length, int K, int distance[], int category[], int nid[]);

The classification of a vector consists of broadcasting it to the neurons and reading the response of the “firing” neurons with successive readout of their distance, category, and optionally the neuron identifier registers. This is when the “Winner Takes All” inter-connectivity between the neurons takes place so each neuron knows if it holds the next smallest distance and its model is consequently the next closest match to the vector. Depending on your application, you can decide to execute this readout only once and get the closest match, or to iterate and get K readout or the response of the K closest firing neurons. It is also possible to limit the response to a simple classification status which is known immediately at the end of the broadcast.

Inputs

A vector is a sequence of bytes with a length between 1 and 256. This sequence can derive from any type of data source such as measurements, signal, sound, images, videos, etc. It can be composed of raw data or represent a signature extracted from raw data and also called a feature.

K indicates the maximum number of responses to read out, if applicable. K is a maximum if the network is set to RBF mode since there is no warranty that K neurons will fire. K is always applicable if the network is set to KNN mode.

Outputs

The first 2 instantiation of the function returns the Network Status Register and its lower byte can be decoded as follows

- NSR[7:0]=0, the vector is not recognized by any neuron (UNKnown)
- NSR[7:0]=8, the vector is recognized and all firing neurons are in agreement with its category (IDentified)
- NSR[7:0]=4, the vector is recognized but the firing neurons are in disagreement with its category (UNCertain)

The extended instantiation of the function returns the number of responses which is also the dimension of the distance, category and nid arrays. This number can be less than or equal to K if the network is set to RBF mode. This number is always equal to K if the network is set to KNN mode.

Distance is the distance between the input vector and the model stored in the firing neuron. It is calculated according to the selected norm (Bit 7 of the Global Context Register).

Category is the category of the firing neuron. It can range between 1 and 32767. Its bit 15 is cleared, so the degenerated status of the neuron, if applicable, is not reported.

Nid is the identifier of the firing neuron. Nid can range between 1 and the number of neurons available in the network.

In the extended instantiation of the function, the values of the distance array are in increasing order. The values of nid[k], distance[k] and category[k] are the neuron identifier of the kth closest firing neuron, its distance register and category.

Considerations before calling this function:

- Change the Network Status Register (NSR) to turn ON/OFF the KNN classifier

void ReadNeuron(int nid, int* context, byte model[],int* aif, int* category);
void ReadNeuron(int nid, byte neuron[]);

Saving the content of a specific neuron consists of switching the network to Save-and-Restore mode, pointing to the specific neuron of the chain and reading sequentially its memory and registers.

Inputs

Nid is the index of the neuron in the chain starting at 1. Nid can range between 1 and the number of committed neurons. If greater, the content of the last committed neuron is returned.

Outputs

context= Neuron context register
model[]= 256 bytes of neuron memory*
aif= Active Influence Field register
category= Category register

Neuron[] is an array of NEURONSIZE+8 bytes as described below.
NEURONSIZE is the memory capacity of each neuron expressed in bytes and returned by the getNeuronsInfo function (ex: 256 in the CM1K chip, 128 in the Intel® QuarkSE chip)

- Byte 0= 0 (discarded)
- Byte 1= NCR, Neuron Context Register
- Byte 2-NEURONSIZE+1= NEURONSIZE bytes of neuron memory*
- Byte NEURONSIZE+2 - NEURONSIZE+3: AIF, Active Influence Field
- Byte NEURONSIZE+4 - NEURONSIZE+5: MINIF, Minimum Influence Field
- Byte NEURONSIZE+6 - NEURONSIZE+7: CAT, Category

*Note that if a model stored in a neuron has a length L lesser than NEURONSIZE, only the first L values of the model[] are significant. The remaining values might be irrelevant or noisy unless the Init function was executed prior to starting the learning.

Int(count) ReadNeurons(byte neurons[]);

The NeuroMem neurons can be saved for backup purposes (in between training sessions for example) or for export and use on other devices.

Saving the content of the neurons consists of switching the network to Save-and-Restore mode, pointing to the first neuron of the chain and reading sequentially their memory and registers until a neuron with a null category is reached.

Outputs

The Neurons output array is a byte array which consists of a header followed by a series of records holding the content of the committed neurons.

Byte 0	**HeaderSize**
Byte 1	Reserved (for future use as a format identifier to decode the values stored in the header)
Byte [2, 3]	**MaxLength** or the maximum length of all the models retrieved from the neurons. This value is set to MAXVECLENGTH by default which is returned by the getNeuronsInfo function (ex: 128 for the Intel® Curie module and 256 for the General Vision CM1K chip). It can be set to a lesser value to optimize the speed of retrieval of the neurons' content if it is known that all the neurons hold models smaller than MAXVECLENGTH
Byte [4 ,7]	**NeuronCount**, or the number of committed neurons read by the function

A “neuron” record has a length of MaxLength + 8 bytes

Byte 0=	0 (discarded)
Byte 1=	NCR, Neuron Context Register
Byte [2 , MaxLength+1]=	MaxLength bytes of neuron memory*
Byte [MaxLength+2 , MaxLength+3]:	AIF, Active Influence Field
Byte [MaxLength+4 , MaxLength+5]:	MINIF, Minimum Influence Field
Byte [MaxLength+6 , MaxLength+7]:	CAT, Category

*Note that if a model stored in a neuron has a length L lesser than MaxLength, only the first L values of the model[] are significant. The remaining values might be irrelevant or noisy unless the Init function was executed prior to starting the learning.

The function returns the number of read neurons.

Int(ncount) WriteNeurons(byte neurons[]);

The NeuroMem neurons can be restored from backup files or files exported from another hardware platform. This operation erases any previous knowledge built and residing in the neurons.

Loading the content of the neurons consists of switching the network to Save-and-Restore mode, pointing to the first neuron of the chain and writing sequentially to their memory and registers. Execution of this function erases any previous knowledge held by the neurons.

Inputs

Neurons is a byte array describing the contents of the neurons to restore. It is expected to have the following format:

Byte 0	**HeaderSize.** A value of “0” signifies that the input array does not include any header and should be read as the byte 0 of the record of the first neuron.
Byte 1	Reserved (for future use as a format identifier to decode the values stored in the header)
Byte [2, 3]	**MaxLength** or the maximum length of all the models retrieved from the neurons. This value is set to MAXVECLENGTH by default which is returned by the getNeuronsInfo function (ex: 128 for the Intel® Curie module and 256 for the General Vision CM1K chip). It can be set to a lesser value to optimize the speed of retrieval of the neurons’ content if it is known that all the neurons hold models smaller than MAXVECLENGTH
Byte [4 ,7]	**NeuronCount**, or the number of committed neurons read by the function

A “neuron” record has a length of MaxLength + 8 bytes

Byte 0=	0 (discarded)
Byte 1=	NCR, Neuron Context Register
Byte [2 , MaxLength+1]=	MaxLength bytes of neuron memory*
Byte [MaxLength+2 , MaxLength+3]:	AIF, Active Influence Field
Byte [MaxLength+4 , MaxLength+5]:	MINIF, Minimum Influence Field
Byte [MaxLength+6 , MaxLength+7]:	CAT, Category

Outputs

The function returns the number of committed neurons once the input array has been loaded into the silicon neurons. This value is equal to 0xFFFF is the network is full meaning that the neurons array exceeds the network capacity.

void SetContext(int context, int minif, int maxif)

Select the context and associated minimum and maximum influence fields for the next neurons to be committed. The context can be associated to a sensor, a type of feature extraction, a scale, a combination of these parameters or else. For more details refer to the NeuroMem Technology Reference Guide.

Inputs

Context, context of the newly committed neurons
Minif, minimum influence field of the newly committed neurons
Maxif, maximum influence field of the newly committed neurons

SIMPLE SCRIPT

The following script is intended to illustrate the behavior of the neurons during learning and recognition operations. It can execute using the GVcomm.cpp interfacing to a NeuroStack board or to the simulation of a NeuroMem network.

In this example, the vectors are set to simple, short and fixed data set such as [11,11,11,11] and [15,15,15,15], such that their relationship is easy to understand and represent graphically. However in real applications, these vectors are extracted from real data and can be composed of up to 256 different values ranging between 0 and 256. For example, in the case of images, vectors can represent raw pixel values, grey-level or color histograms, etc. In the case of audio and voice data, vectors can be cepstral coefficients, etc.

Step 1: Learn

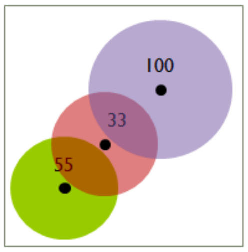
Learn vector1 [11,11,11,11] (1) with category 55
Learn vector2 [15,15,15,15] with category 33(2)
Learn vector3 [20,20,20,20] with category 100

(1) The learned vectors are purposely set to arrays of constant values so their representation and relationship are easy to understand. The distance between two "flat" vectors is indeed the difference between their constant value times their number of components. For example the distance between [11,11,11,11] and [15,15,15,15] is equal to 4 * 4. This simple arithmetic makes it easy to understand the different cases of recognition illustrated in this test script.
(2) The category of the second neuron is purposely set to a lesser value than the category of the first neuron to verify that if both neurons fire with a same distance, the category of the neuron on the 2nd chip is still the first the be read out

Fig1 is a representation of the decision space modeled by the 3 neurons where Neuron1 is shown in red, Neuron2 in green and Neuron3 in blue. In the following 2D graph, we limit the length of the models to 2 components instead of 4, so they can be positioned in an (X,Y) plot. X=1st component and Y=Last component, and a surrounding diamond shape with the side equal to their influence field.

Committed neurons= 3

NeuronID=1	Components=11, 11, 11, 11,	AIF=16,	CAT=55
NeuronID=2	Components=15, 15, 15, 15,	AIF=16,	CAT=33
NeuronID=3	Components=20, 20, 20, 20,	AIF=20,	CAT=100



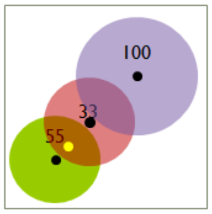
The influence fields of Neuron#0 and Neuron#1 overlap, as well as Neuron#1 and Neuron#2 overlap but differently since their distances from one another are different.

Step2: Recognition

The vectors submitted for recognition are selected purposely to illustrate cases of positive recognition with or without uncertainty, as well as cases of non recognition.

The program reads the responses of all the firing neurons, which is until the distance register returns a value 0xFFFF or 65553.

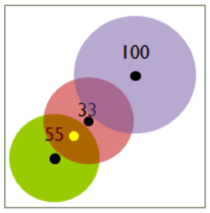
Case of uncertainty, closer to Neuron#1



Vector=12, 12, 12, 12
Neuron 1 and 2 fire. Vector is closer to Neuron1

Response#1	Distance= 4	Category= 55	NeuronID= 1
Response#2	Distance= 12	Category= 33	NeuronID= 2
Response#3	Distance= 65535	Category= 65535	NeuronID= 65535

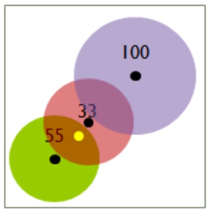
Case of uncertainty, equi-distant to Neuron#1 and Neuron#2



Vector=13, 13, 13, 13,
Neuron 1 and 2 fire. Vector is equi-distant to both neurons

Response#1	Distance= 8	Category= 33	NeuronID= 2
Response#2	Distance= 8	Category= 55	NeuronID= 1
Response#3	Distance= 65535	Category= 65535	NeuronID= 65535

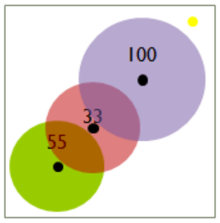
Case of uncertainty, closer to Neuron#2



Vector=14, 14, 14, 14,
Neuron 1 and 2 fire. Vector is closer to Neuron2

Response#1	Distance= 4	Category= 33	NeuronID= 2
Response#2	Distance= 12	Category= 55	NeuronID= 1
Response#3	Distance= 65535	Category= 65535	NeuronID= 65535

Case of unknown

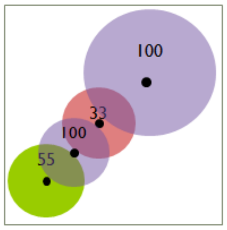


```
Vector=30, 30, 30, 30,
No neuron fire
 Response#1       Distance= 65535       Category= 65535      NeuronID= 65535
```

Step3: Adaptive learning

Learning a new vector to illustrate the reduction of neurons'AIFs.

Learn vector[13,13,13,13] with category 100. This vector is equidistant to both Neuron1 and Neuron2. Learning it as a new category, will force Neuron1 and 2 to shrink from the AIF=16 to an AIF=8 in order to make room for a new neuron which will hold the new model and its category.



```
Committed neurons= 4
NeuronID=1      Components=11, 11, 11, 11,     AIF=8,          CAT=55
NeuronID=2      Components=15, 15, 15, 15,     AIF=8           CAT=33
NeuronID=3      Components=20, 20, 20, 20,     AIF=20          CAT=100
NeuronID=4      Components=13, 13, 13, 13,     AIF=8,          CAT=100
```

Note that if the vector to learn was [12,12,12,12], Neuron1 would shrink to 4 and Neuron2 to 12.

FUNCTIONS OF THE NEUROMEM_API_GEEK

The NeuroMem API Geek gives you full access to the neuron registers so you can define your own advanced functions with the optimum number of access registers. You can among other things deploy applications using multiple contexts and supporting sensor fusion, multiple features consolidation, or cascade classifier and more. For details refer to the chapter “Network segmentation into context” in the NeuroMem Technology Reference Guide.

void GetContext(int context, int minif, int maxif)
void SetContext(int context, int minif, int maxif)

Read or write the current values of the Global Context and its associated Minif and Maxif registers

void SetRBF

Set the network to Radial Basis Function mode, more precisely as a Restricted Coulomb Energy classifier.

void SetKNN

Set the network to K-Nearest Neighbor mode.

Int NCOUNT()
Read the number of committed neurons
Note that this register is returned by the Learn, ReadNeurons and WriteNeurons functions.

Int NSR()
NSR(int value)
Read or write the Network Status Register.
Note that this register is returned by the Classify function and masked by the SetRBF and SetKNN functions

Int GCR()
GCR(int value)
Read or write the Global Context Register.
Note that this register is written and read by the SetContext and GetContext functions.

Int MINIF()
MINIF(int value)
Read or write the MINIF or Minimum Influence Field register.
Note that this register is written and read by the SetContext and GetContext functions.

Int MAXIF()
MAXIF(int value)
Read or write the MAXIF or maximum Influence Field register.
Note that this register is written and read by the SetContext and GetContext functions.

Int DIST()
Read the DIST or distance register.
Note that this register is read by the Classify functions.

Int CAT()

CAT(int value)

Read or write the CAT or category register.
Note that this register is written by the Learn function and read by the Classify functions.

Int NID()
NID(int value)

Read or write the NID or Neuron Identifier register.
Note that this register is read by the Classify functions.

RSTCHAIN(int value)

Write the RESETCHAIN register. Value is always 0.
Note that this register is only accessible in Save and Restore mode (NSR=16)

IntAIF()
AIF(int value)

Read or write the AIF or Active Influence Field register of the neuron in focus.
Note that this register is only accessible in Save and Restore mode (NSR=16)

IDX(int value)

Write the IDX or Component Index register of all the neurons at once.