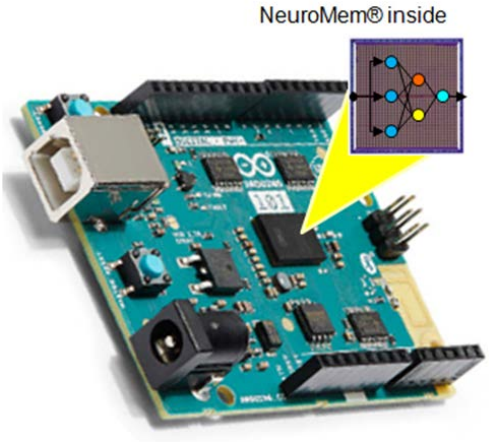
CurieNeurons library



Version 1.4.2
Revised 11/09/2017

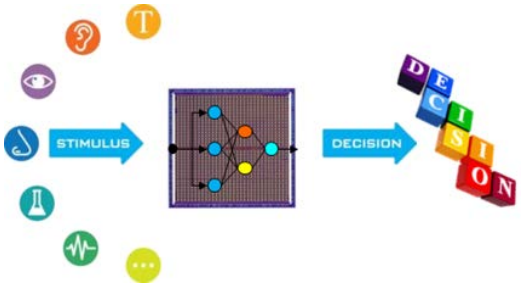
GENERALVISION

Contents

1 Introduction

The Intel Quark SE embedded in the Curie module features 128 NeuroMem neurons offering the following unique capabilities:

- ☐ The neurons learn by examples
  - ◘ No programming
  - ◘ Training can be done off-line or the fly
- ☐ Continuous monitoring at low-power
- ☐ Can detect novelty or anomaly
- ☐ Knowledge portability

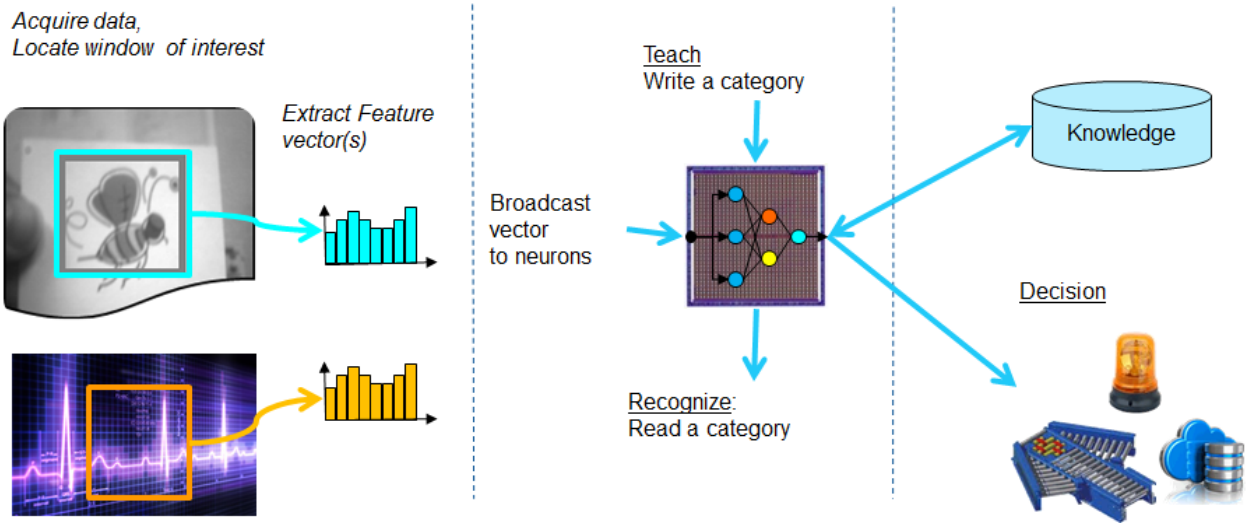


The NeuroMem neurons are very easy to use through the CurieNeurons library.

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. All you must do is focus on the quality of the input signals, the selection of relevant and discriminant feature extractions.

The NeuroMem neurons handle the automatic learning of your examples and associated categories, the recognition of new patterns, the detection of uncertainty cases if any, the detection of drifts or decrease of confidence, the reporting of novelties and/or anomalies.

Finally, you must format the response of the neurons to convert it into a global decision or action for your application, without forgetting to save the knowledge built by the neurons for backup purposes or distribution to other systems.

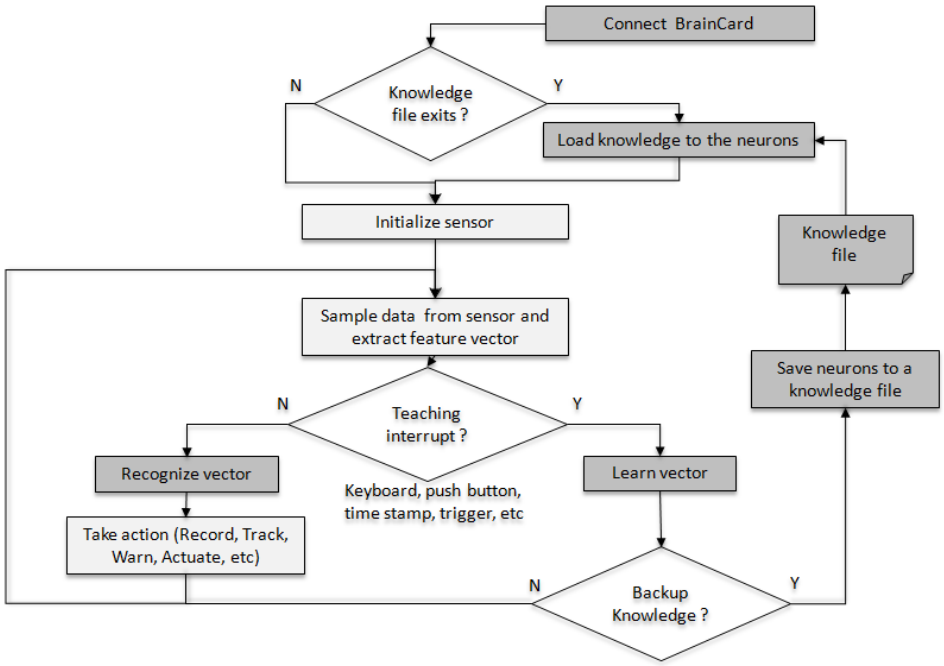


- ☐ 4 basic functions
  - ◘ Learn vector
  - ◘ Recognize vector
  - ◘ Save knowledge (read neurons' content)
  - ◘ Restore knowledge (write neuron's content)
- ☐ Additional functions
  - ◘ Tuning and expansion options (available in the CurieNeurons Pro version)

2 Getting Started

The data collected through the sensors can be broadcasted to the neurons for learning and recognition. Learning can be triggered by external user inputs, time stamps, but also the ability of the neurons to detect drifts from learned models. Learning can also be done “off line” on data previously collected and saved. Recognition can consist of using the neurons to classify input patterns or identifying novel or abnormal patterns. Depending on your application, the output of the neurons can control actuators, trigger a selective recording or transmission or else. Applications include identification, surveillance, tracking, adaptive control and more.

Typical workflow



How do the neurons learn and recognize?

Our NeuroMem RBF tutorial is a simple application demonstrating how the neurons build a decision space autonomously by learning examples and how they recognize new vectors with ability to report cases of unknown and uncertainties too. For the sake of a simplicity, the decision space is limited to a 2D space but the mechanism is the same to recognize a (X1, X2) vector as well as an (X1, X2, … X127) vector which is the maximum length supported by the neurons of the QuarkSE.

For more information of the NeuroMem technology, please refer to the NeuroMem Technology Reference Guide.

What is a feature vector?

The neurons are agnostic to the data source which means that they ready to learn and recognize feature vectors extracted from text (parsing), discrete measurements, audio signal, video signal, digital images, etc. The only constraint is that the vector must be formatted as a byte array of maximum length 128 in the case of the Intel® QuarkSE.

This means that the CurieNeurons are not limited to classifying the signals of the 6-axis mems of the Curie module, but can also be interfaced to biosensors, cameras, etc.

Can I do image recognition?

There are several camera modules available for Arduino which means that you can very well program an application using the CurieNeurons to learn and recognize objects or events. The neurons of the Curie module are capable of learning patterns of 128-bytes long, so when dealing with pixel areas larger than 128 pixels it is common to use a subsampling to generate a feature vector of 128-bytes. Such function is included in the example CurieNeurons_andArduCam1. Other common feature extractions are a color histogram and a composite profile (horizontal + vertical). They are included in the example “CurieNeurons_andArduCam2. Surely, you can invent your own too!

3 The function libraries

The functions of the CurieNeurons library are described in the technical manual TM_NeuroMem_API.pdf.

```
int begin();
void getNeuronsInfo(int* neuronSize, int* neuronsAvailable, int* neuronsCommitted);
void forget();
void forget(int Maxif);
void clearNeurons();

int learn(unsigned char vector[], int length, int category);
int classify(unsigned char vector[], int length);
int classify(unsigned char vector[], int length, int* distance, int* category, int* nid);
int classify(unsigned char vector[], int length, int K, int distance[], int category[], int nid[]);

void readNeuron(int nid, int* context, unsigned char model[], int* aif, int* category);
void readNeuron(int nid, unsigned char neuron[]);
int readNeurons(unsigned char neurons[]);
int writeNeurons(unsigned char neurons[]);
```

Advanced functions with full access to the registers of the neurons and in particular the following powerful options:

- The ability to segment the neural network into context
- Choice of the L1 or LSup norm for the distance calculation
- Option to classify using the Radial Basis Function or K-Nearest Neighbor mode

These powerful features must be used with caution, so make sure to refer to the NeuroMem Technology Reference Guide for a clear understanding of how the neurons behave.

```
void setContext(int context, int minif, int maxif);
void getContext(int* context, int* minif, int* maxif);
void setRBF();
void setKNN();
int Read();
void Write();
```

4 The examples

A series of short and academic test programs are supplied for the Arduino IDE. In addition to the short descriptions below, detailed comments are included in the source code and print statements executed by the code.

Test_SimpleScript

Simple script stimulating the neurons to learn and recognize patterns generated programmatically. Detailed description of this script is available at http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf.

Test_SimpleScript2 (library Pro only)

This script is similar to the Test_SimpleScript but also demonstrates additional capabilities of the neurons which are accessible in the CurieNeurons library Pro, including but not limited to the use of the KNN, the use of the LSup/L1 norms for the calculation of distances, and the use of multiple contexts to learn and classify vectors representing different dimensions or data types within the same NN.

In addition to the script described in http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf , you can practice with:

- Switching the NN from Radial Basis Function mode to K-Nearest Neighbor mode and comparing results between the two classifiers using a same knowledge and same input vector. KNN always gives a response for the closest match while RBF can report cases of “unknown” and cases of uncertainties.
- Training the neurons with vectors belonging to 3 different contexts with the 3rd context activating the Lsup norm.

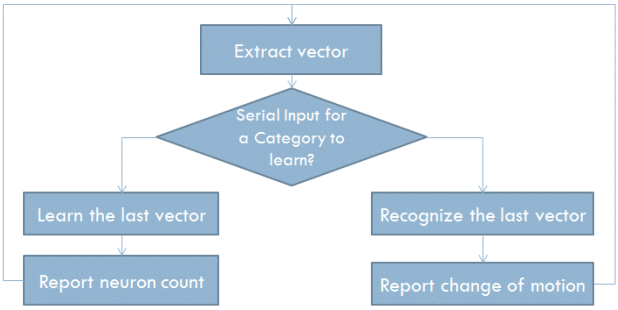
For more information about the RBF/KNN classifiers, the definition of Contexts, the use of different Norms, please refer to the NeuroMem Technology Reference Guide.

Test_Neuron_andIMU1

This script interfaces with its 6-axis IMU. The script assembles signals from the accelerometer and gyroscope into a simple feature vector broadcasted continuously to the neurons for recognition. Learning is performed by entering a category value through the serial input. Refer to the General Vision movie tutorial.

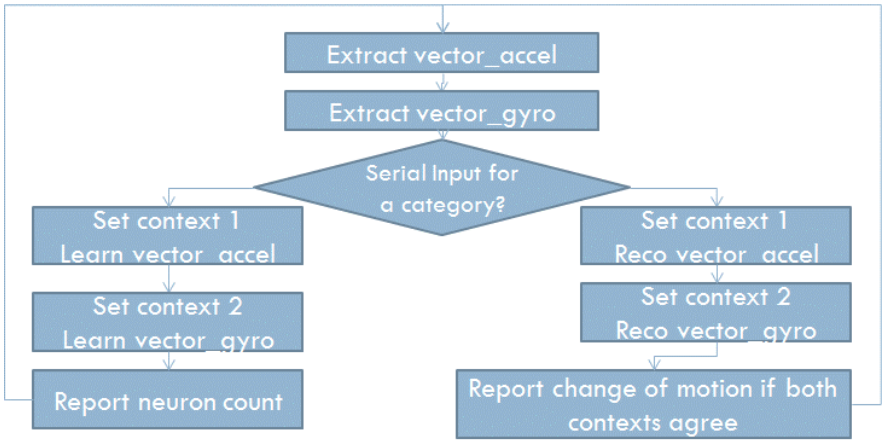
In the Arduino code, the LOOP continuously reads the IMU signals and extracts a simple feature vector which is a normalized subsampling of the 6 axes. The recognition of this vector starts automatically as soon as the neurons have some knowledge. The neurons build the knowledge as soon as you start teaching examples.
When you enter a category through the serial port, the program associates it to the last feature vector. When teaching a vertical motion for example, you want to teach more than once to make sure the neurons learn an example of a vertical up-to-down and vertical down-to-up, obviously at selected speed and amplitudes.

The script mentions 2 categories (vertical and horizontal) for the sake of simplicity, but feel free to define your own categories combining not only a direction, but also other motion such as circular, a range of amplitude and speed, etc. Remember that the category value can range between 0 to 32767 which means that it can encode multiple criteria (ex: bit[2:0] for direction, bit [5:3] speed range, etc.).



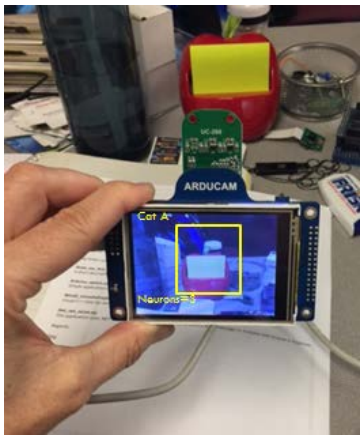
Test_Neuron_andIMU2 (library Pro only)

Test_Neurons_andIMU2 is similar to the Test_Neurons_andIMU1, except that it illustrates the ability of the NeuroMem neurons to handle multiple networks in a same chip for more robust decision making. The signals of the accelerometers and gyroscope are assembled into two separate feature vectors and associated to 2 different contexts. Learning a motion builds 2 decision spaces at once and consequently commits more neurons. The script displays a positive response only if both sub-networks agree with the classification, thus producing a more conservative but accurate response than in the script Test_Neurons_andIMU.



Remark1: This example is very academic and assemble a pattern which should be more sophisticated for real-life system taking a calibration into account, integrating a sampling rate adequate for the type of motion and profiling the waveforms more selectively using distances between peaks and zero crossing, etc.

Test_NeuronsandArduCam1



Hardware requirements

- Arduino/Genuino 101
- ArduCam Shield V2 with camera module
-

Library requirements

- ArduCAM at https://github.com/ArduCAM/Arduino
- UTFT4ArduCAM_SPI at https://github.com/ArduCAM/Arduino

Description

- Display video frame and the region of interest being monitored on the LCD
- Recognize continuously the region of interest
- When shutter button is depressed…
    - < 2 seconds ==> learn category 1 and increment its number optionally
    - > 2 seconds ==> learn category 0 or background to correct erroneous classification
    - Optionally for debug purposes, save to SD card the following:
        - Feature vectors and their taught category (appended in a same vectors.txt file)
        - Image (filename includes the taught category)
        - Knowledge file for portability to another Arduino/Genuino101
        - Be patient and make sure that the SD_LED is no longer blinking to continue

Hints

- The proper detection of the shutter being depressed is confirmed as soon as the region of interest freezes on the LCD screen
- If it is pressed too long, the example is taught as category 0 and the mention “forget” appears at the bottom of the screen.
- If the option to save is not commented, the vector will be appended to the vectors.txt file, but the number of neurons in the knowledge.dat file will not change (their influence fields might)

Careful

- The category to learn is incremented at each press of the shutter. This which limits the use cases and means that you have only one chance to teach an object of a new category.
- Note that you can easily change this in the code and limit the script to the teaching of a single category with a value of your choice (example: 1= “Good quality”) and the null category (0=“anything else” or “discard” category).

How to improve the speed

- Migrate the code executing the readout of the pixels from a FIFO and the feature extraction to the FPGA of the ArduCam Shield. Even better, bypass the use of the FIFO and generate the feature vectors on the FPGA as the video comes in
- Use another ArduCam board and manage the display with another hardware

How to improve the UI

- Variety of user inputs (BlueTooth, Push button, etc)
- Ability to select a category value at the time of teaching
- Display of category labels instead of numbers

- Ability to move the ROI with arrow cursors if the device if mounted on fixed fixture
- Automatically load a knowledge during setup if a file “default.knf” is found on SDcard

Possible use cases

- Monitor a cat door to ensure a raccoon is not sneaking in
- Monitor if a flame is present or not
- Inspect color parts (candies?)
- If Arduino shield with motors, learn a target centered in FOV and use neurons to control the motors so the camera always points at the target

Test_Neurons_andArduCam2 (Pro library only)

This script illustrates how easy it can be to teach multiple networks at once, learning different features extracted from the same objects and making a robust decision requiring a minimum of agreements between the networks.

It is exactly the same as the Test_Neurons_andArduCam1 except that it extracts three different feature vectors to learn and classify the objects: a subsample, a histogram of the colors and a profile of the lines and columns.

Recognition based on multiple features or contexts

The 3 features are calculated at the same time during the readout of the pixel values from the FIFO.

- Feature 1= pixel subsampling ➔ used to train neurons assigned to Context 1
- Feature 2= RGB histogram ➔ used to train neurons assigned to Context 2
- Feature 3= Composite profile➔ used to train neurons assigned to Context 3

Decision rule between contexts

The script reports a category on the screen only if the neurons of 2 out of the 3 contexts (at the minimum) agree on the same category. Otherwise, the script reports the object as “Unknown”.