NeuroMem library for Arduino IDE

Version 1.3

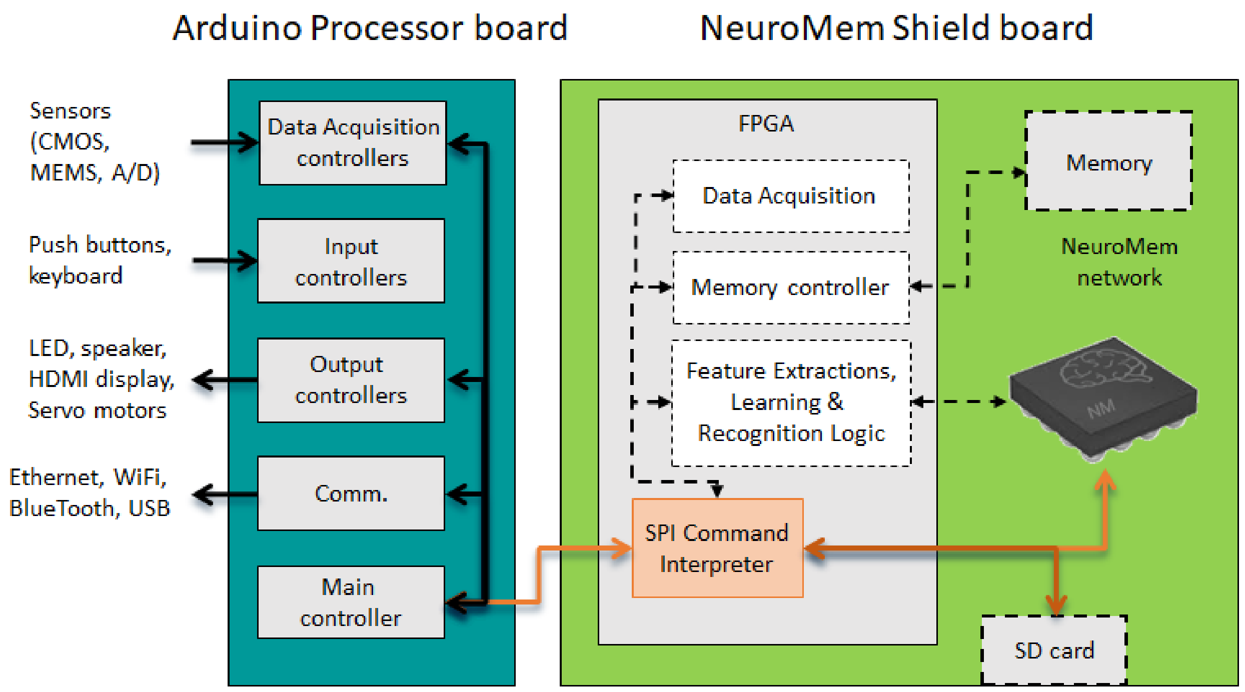
Revised 05/31/2018

GENERALVISION

CONTENTS

1 INTRODUCTION

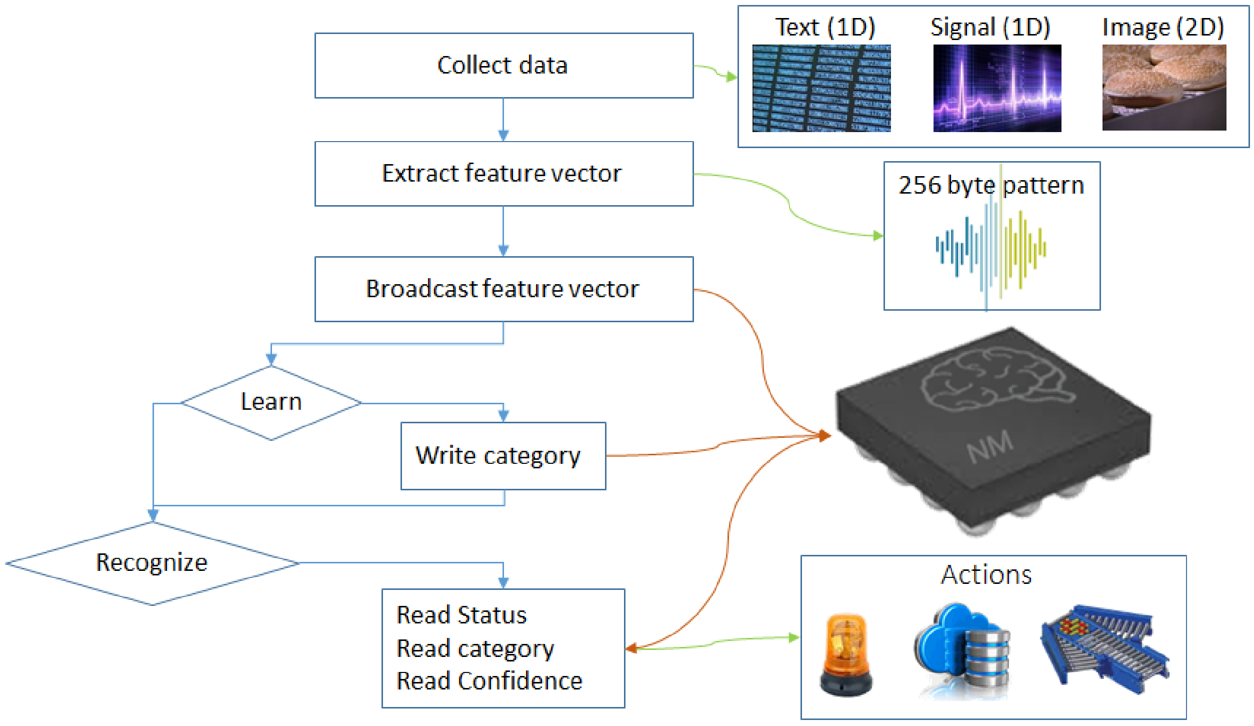
The NeuroMem library for Arduino is intended to access a NeuroMem network through compatible Arduino boards and Shields. Such boards must be populated with a network of one or more NeuroMem chips (CM1K or NM500) and a Field Programmable Gate Array configured to decode the NeuroMem Smart SPI protocol described in this manual. This protocol has been designed to dispatch SPI commands to the NeuroMem bus, but also other components such as sensors and SD card as presented below:



Step 1: Microcontroller
Sensor data collected through Arduino shields and other plug-in modules can be assembled into feature vectors(*).
Their learning can be triggered by external user inputs, programmable time stamps, but also by the detection of novelties made by the neurons themselves.
The pattern vectors are then broadcasted to the NeuroMem neurons for either learning or recognition(*).

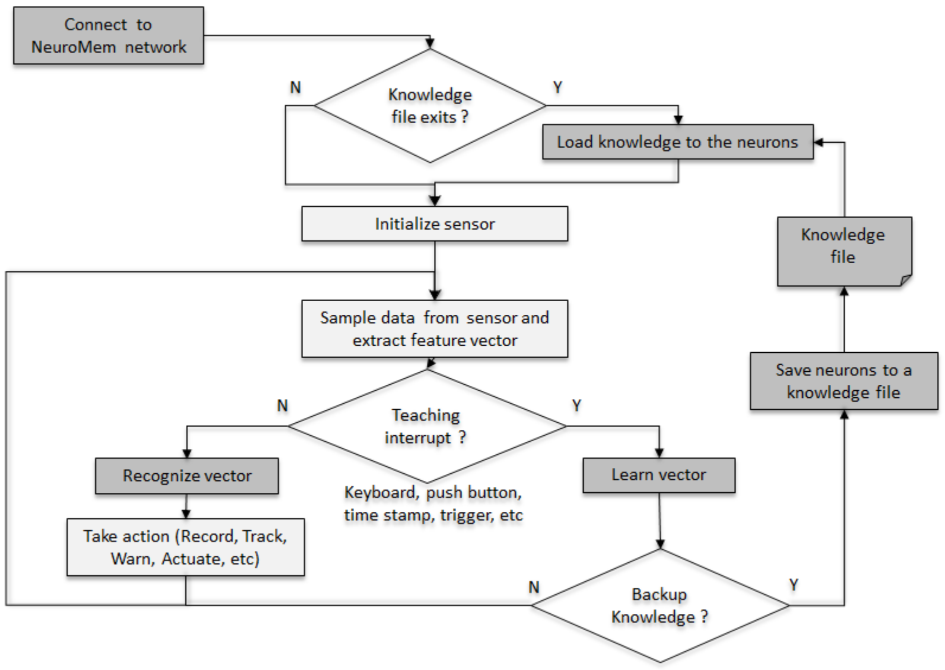
Step 2: NeuroMem library
The NeuroMem neurons handle the automatic learning of your examples and associated categories, the recognition of new patterns, the detection of uncertainty cases if any, the detection of drifts or decrease of confidence, the reporting of novelties and/or anomalies.
(*) can also be programmed in the FPGA to optimize speed performance

Step 3: Microcontroller
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(*).
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.



1.1 Typical application workflow

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.


Connect to NeuroMem network
Knowledge file exits ?
N
Y
Load knowledge to the neurons
Initialize sensor
Knowledge file
Sample data from sensor and extract feature vector
Save neurons to a knowledge file
Teaching interrupt ?
N
Y
Recognize vector
Learn vector
Keyboard, push button, time stamp, trigger, etc
Take action (Record, Track, Warn, Actuate, etc)
Backup Knowledge ?
N
Y

1.1.1 *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 CM1K.

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

1.1.2 *What is a feature vector?*

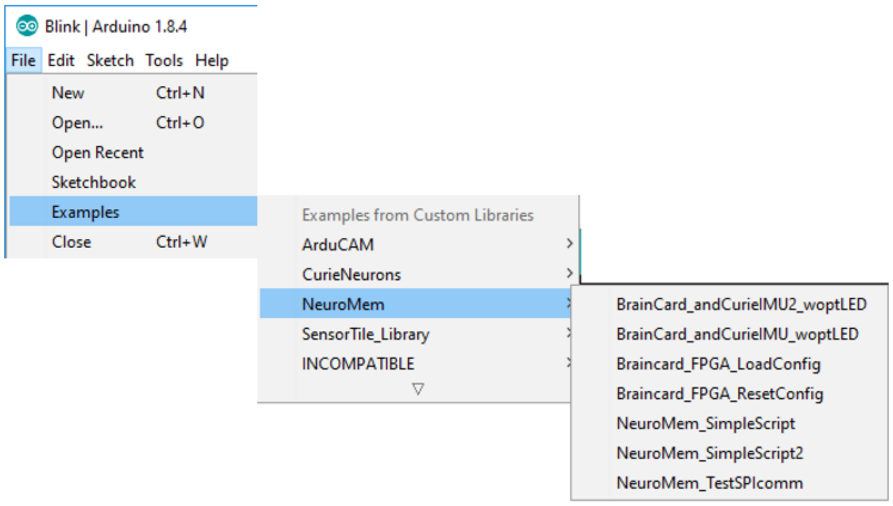
The neurons are agnostic to the data source which means that they ready to learn and recognize feature vectors extracted from text (parsing), discrete measurements, audio signal, video signal, digital images, etc. The only constraint is that the vector must be formatted as a byte array of maximum length 256.

In the case of images, a feature vector can be a subsampling of pixels within a region of interest, a histogram, some histogram of gradients within a patch of pixels, etc. In the case of an audio or biosensor signal, the feature vector can be a set of signal samples taken at a specific sampling rate, a histogram of peaks or zero crossing, or a power spectrum, etc.

1.2 Installation

Copy the NeuroMem folder to MyDocuments\Arduino\libraries
Launch the Arduino IDE
Under Tools/Board Manager menu, select your Arduino processor board (Uno, 101, STM32, etc.)
Under File/Examples/NeuroMem, you can access

- Generic examples for all NeuroMem-Smart devices (NeuroMem_xyz.ino)
- Hardware specific examples (filename starting with the board's name))



2 THE NEUROMEM LIBRARY

2.1 NeuroMemAI library

The functions of the NeuroMemAI library are described in the technical manual TM_NeuroMem_API.pdf. They have a dependency to the NeuroMemSPI library in order to access the hardware.

```
NeuroMemAI();
int begin(int Platform);
void forget();
void forget(int Maxif);
void clearNeurons();
int countNeuronsAvailable();

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

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

2.2 NeuroMemSPI library

The NeuroMem Smart protocol is described http://www.general-vision.com/documentation/TM_NeuroMem_Smart_protocol.pdf

```
NeuroMemSPI();
int connect(int Platform);
int FPGArev();
int read(unsigned char reg);
void write(unsigned char reg, int data);
void writeAddr(long addr, int length, int data[]);
void readAddr(long addr, int length, int data[]);
```

3 ACADEMIC EXAMPLES

3.1 Test_SimpleScript

Simple script stimulating the neurons to learn and recognize patterns generated programmatically.

Teach the neurons to learn a set of predefined vectors and practice with cases of recognition demonstrating how the neurons can be queried to report positive identification, or on the contrary uncertain identification, or unknown identification.

Detailed description of this script is available at http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf.

3.2 Test_SimpleScript2

Extension of the Test_SimpleScript demonstrating additional capabilities of the neurons, including 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.

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.

4 EXAMPLES OF MOTION LEARNING AND RECOGNITION

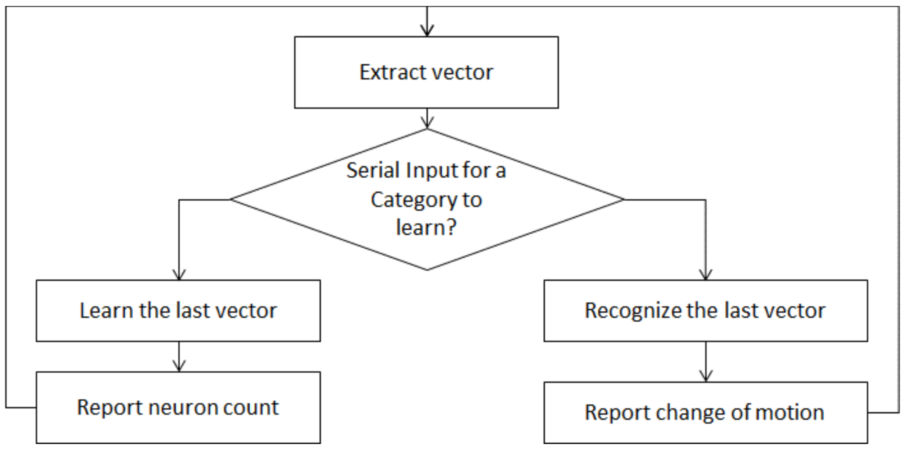
View Step-By-Step Tutorial

4.1 NeuroMem_andCurieIMU1

Hardware requirements

- Arduino 101

This script requires an Arduino/Genuino101 to use the 6-axis IMU of the Intel Curie module. It 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.

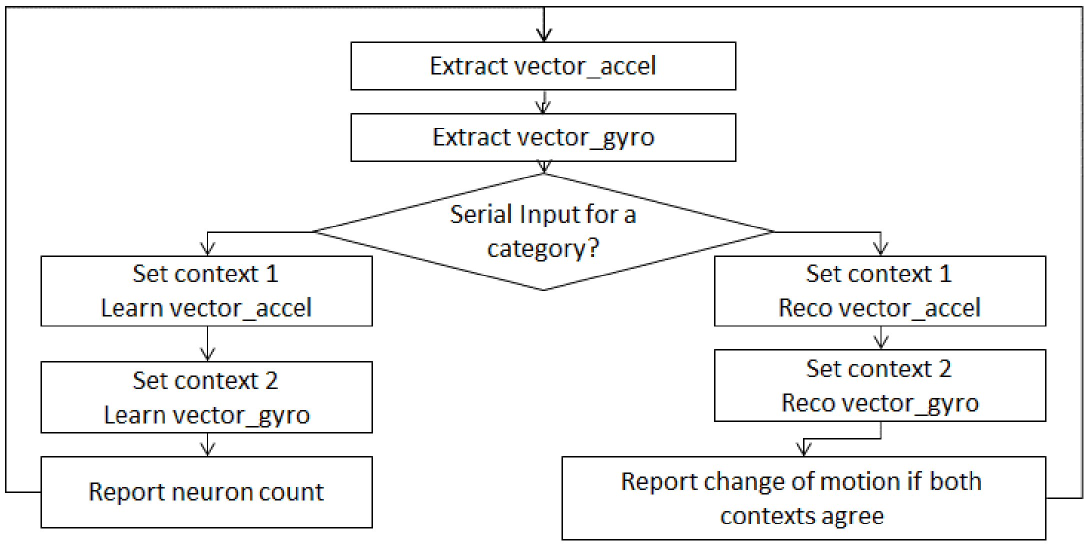
4.2 NeuroMem_andCurieIMU2

Hardware requirements

- Arduino 101

Test_Neurons_andIMU2 is similar to the Test_Neurons_andIMU1 and illustrates the ability of the NeuroMem neurons to handle multiple networks in a same chip for more robust decision making. This script requires an Arduino/Genuino101 to use the 6-axis IMU of the Intel Curie module.

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.


Extract vector_accel
Extract vector_gyro
Serial Input for a category?
Set context 1 Learn vector_accel
Set context 2 Learn vector_gyro
Report neuron count
Set context 1 Reco vector_accel
Set context 2 Reco vector_gyro
Report change of motion if both contexts agree

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.

4.3 NeuroShield_andIMU

Hardware requirements

- NeuroShield
-

Library requirements

- https://playground.arduino.cc/Main/MPU-6050
- https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
- https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/I2Cdev

This script uses the Invensense 6-axis IMU of the NeuroShield. It 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.

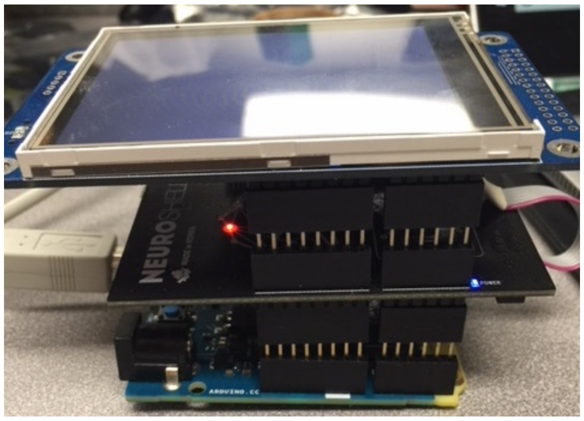
In the Arduino code, the LOOP continuously reads the IMU signals and extracts a simple feature vector which is a normalized subsampling of the 6 axes. The recognition of this vector starts automatically as soon as the neurons have some knowledge. The neurons build the knowledge as soon as you start teaching 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.

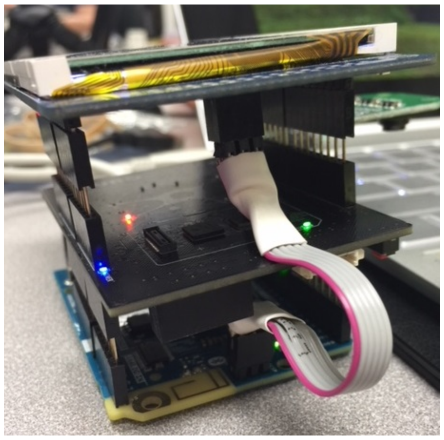
5 EXAMPLES FOR IMAGE LEARNING AND RECOGNITION

View [Step-By-Step Tutorial]()

Hardware requirements

- ArduCam Shield V2 with camera module
- Custom cable to connect the Arduino Standard SPI connector (6-pin) from the Arduino microcontroller board to the ArduCam board while going around the NeuroShield or BrainCard board which does not pass these SPI lines through. Refer to the images below.
- Note that you can also unsolder the 3x2 connector of the ArduCam shield and wire the signals MISO, MOSI and Clock to pin 11,12,13 of the P1 connector.





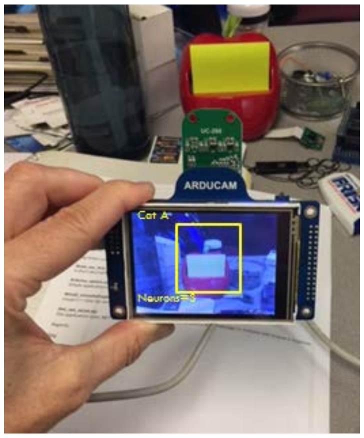
Library requirements

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

5.1 NeuroMem_andArduCam1

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).

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

Ideas 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

5.2 NeuroMem_andArduCam2

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 Neuromem_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

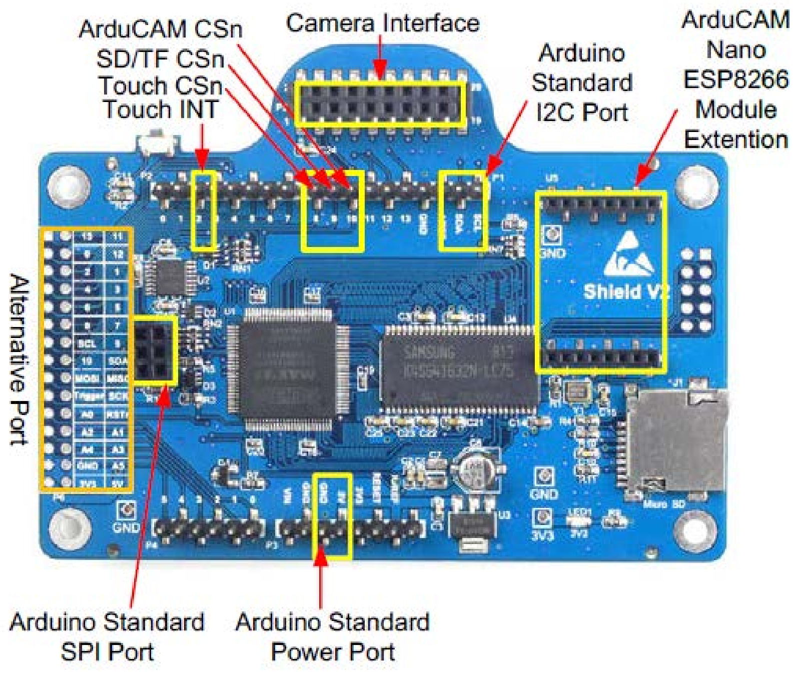
Decision rule between contexts
The script reports a category on the screen only if the neurons of 2 out of the 3 contexts (at the minimum) agree on the same category. Otherwise, the script reports the object as “Unknown”.

5.3 Connectivity

The ArduCam Shield use the ICSP connector as SPI access and unfortunately the NeuroShield does not offer a pass-thru for this connector, thus requiring an intermediate and non-aesthetic cable as shown above.

Alternative is to proceed with the following soldering between ICSP and P1 Arduino connectors:

Signal	ICSP	P1
MOSI	Pin4	D11
MISO	Pin1 (upper-right in front view)	D12
SCK	Pin3	D13


ArduCAM CSn
SD/TF CSn
Touch CSn
Touch INT
Camera Interface
Arduino Standard I2C Port
ArduCAM Nano ESP8266 Module Extention
Alternative Port
Arduino Standard SPI Port
Arduino Standard Power Port

7

6 TROUBLESHOOTING SPI

6.1 The Arduino board cannot be programmed properly

- Verify that it is properly powered
- Verify that the selected COM port is correct
- Unplug all shield boards before programming

6.2 The Arduino board is not responding to SPI communication

- Verify that it is properly powered
- Push the reset buttons
- Run the Test_SPIComm and diagnose the type of the error (single Read/Write, multiple Read/Write)
- Test a different SPI speed (refer to your processor documentation)
- Test powering the platform with an external power supply
- Verify that the SPI Chip Select pin is not used by another shield
- The default SPI clock divider is one-quarter the frequency of the system clock (4 Mhz for the boards at 16 MHz). Depending on your Arduino processor, you might want to increase it using the function SetClockDivider.