Cross-correlation of brain neurons in 1D and 2D

Table of Contents

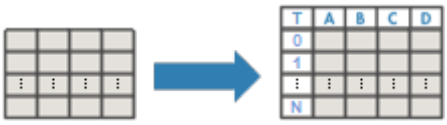

```
% Clean current MATLAB session
clear variables; close all; clc;
```

1. Import the measured data (channels) to MATLAB

In the *HOME* tab, there is a link to the App called *Import Tool* that allows to easily import various type of data. It can also be called from the command line using this function call:

```
% uiimport;
```

The data are organized as a matrix of 4 columns and N lines and there is a need to get a kind of table with headers and the time as well:



Once the right data import procedure has been defined, MATLAB code can be automatically generated as a *MATLAB script or function* to automate the import process.

```
% Constants declaration
FILENAME = '.\DATA\MA47_Baseline1_DFF.txt';
FIRST_CELL_INDEX = 1; % cannot be smaller than 1
LAST_CELL_INDEX  = inf;

% Get raw data from text file
channels = importData(FILENAME, FIRST_CELL_INDEX, LAST_CELL_INDEX);
% Get matrix dimensions
[NB_FRAMES, NB_NEURONS] = size(channels);
```

2. Add time to the imported data

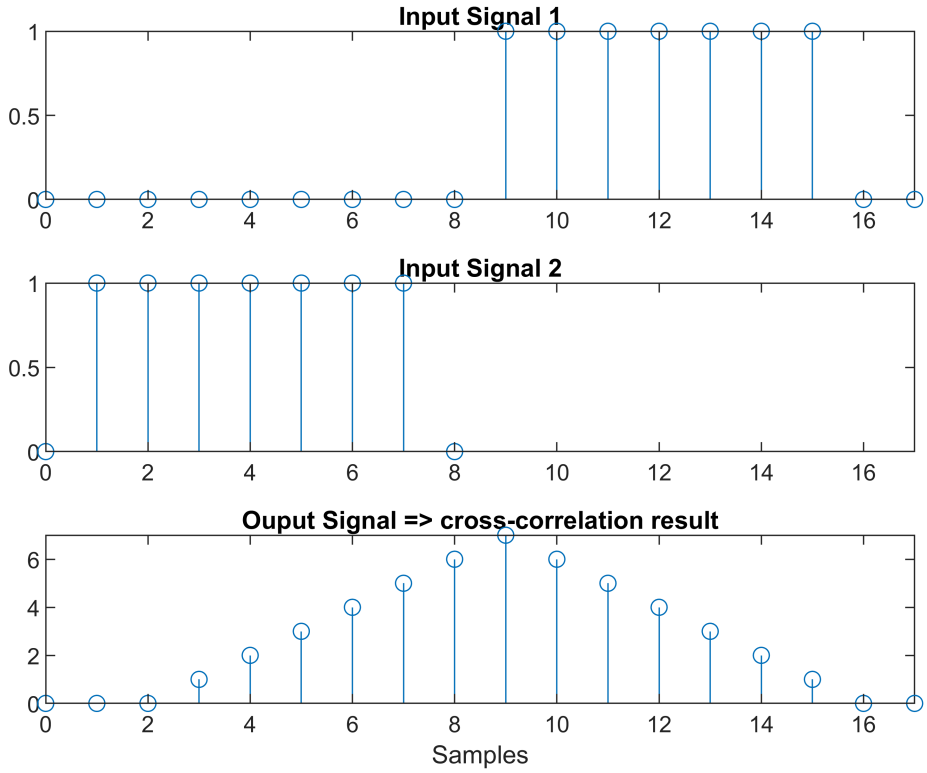
Because of the post-processing of the data in the time and frequency domains, it can be useful to also have the time as a vector within the data. Such dataset are defined as what is called *time series object*.

```
% Frame vector definition
FRAME_FS = 4; % Frame sampling frequency
DELTA_TS = 1/FRAME_FS; % Frame sampling time
T_FRAME_MIN = 0*DELTA_TS;
T_FRAME_MAX = NB_FRAMES*DELTA_TS;
t_frame = (T_FRAME_MIN:DELTA_TS:T_FRAME_MAX-DELTA_TS)';
% Post-processing of input data
series = timeseries(channels,t_frame); % Adding time info to measured data
```

3. Cross-correlation

In signal processing, image processing, statistics and so on, several definitions and implementations of a *correlation* exist. Here, the focus is put on the *cross-correlation* as it is defined for *unidimensional* signals in signal processing. It is a measure of similarity of two series as a function of the displacement of one relative to the other. It also allows to measure the *time delay* between two signals.

```
% Run this code in the Command Window
x = [0 1 1 1 1 1 1 1 0];
y = [0 1 1 1 1 1 1 1 0];
crossCorrAnimation(x,y,1);
```



3.1 For continuous functions

In the continuous time domain, for real continuous functions *f* and g, the cross-correlation is defined as:

$$h(t) = \int_{-\infty}^{\infty} f(t) \cdot g(t+\tau) \cdot dt \quad (1)$$

3.2 For discrete functions

In the discrete time domain, for real discrete functions *f* and g, the cross-correlation is defined as:

$$h[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[m+n] = \widehat{R}_{fg} \quad (2)$$

Depending on the application, if the cross-correlation is used on unidimensional signals or images or stochastic processes, a *normalization* or a *bias* factor can be applied. This factor is in general made of the number of samples and/or the *standard deviation* of the function *f*.

3.2.1 Biased and unbiased cross-correlation

Biased estimate of the cross-correlation:

$$\widehat{R}_{fg,biased}(m) = \frac{1}{N} \cdot \widehat{R}_{fg}(m) \quad (3)$$

Unbiased estimate of the cross-correlation:

$$\widehat{R}_{fg,unbiased}(m) = \frac{1}{N - |m|} \cdot \widehat{R}_{fg}(m) \quad (4)$$

3.2.2 Correlation coefficient

The correlation coefficient of two random variables is a measure of their linear dependence. If each variable has *N* scalar observations, then the *Pearson correlation coefficient* is defined as:

$$\rho(A,B) = \frac{1}{N-1} \cdot \sum \left(\frac{A_i - \mu_A}{\sigma_A}\right)\left(\frac{B_i - \mu_B}{\sigma_B}\right) \quad (5)$$

where μA and σA are the mean and standard deviation of A, respectively, and μB and σB are the mean and standard deviation of B.

The correlation coefficient *matrix* of two random variables is the matrix of correlation coefficients for each pairwise variable combination:

$$R = \begin{pmatrix} \rho(A,A) & \rho(A,B) \\ \rho(B,A) & \rho(B,B) \end{pmatrix} = \begin{pmatrix} 1 & \rho(A,B) \\ \rho(B,A) & 1 \end{pmatrix} \quad (6)$$

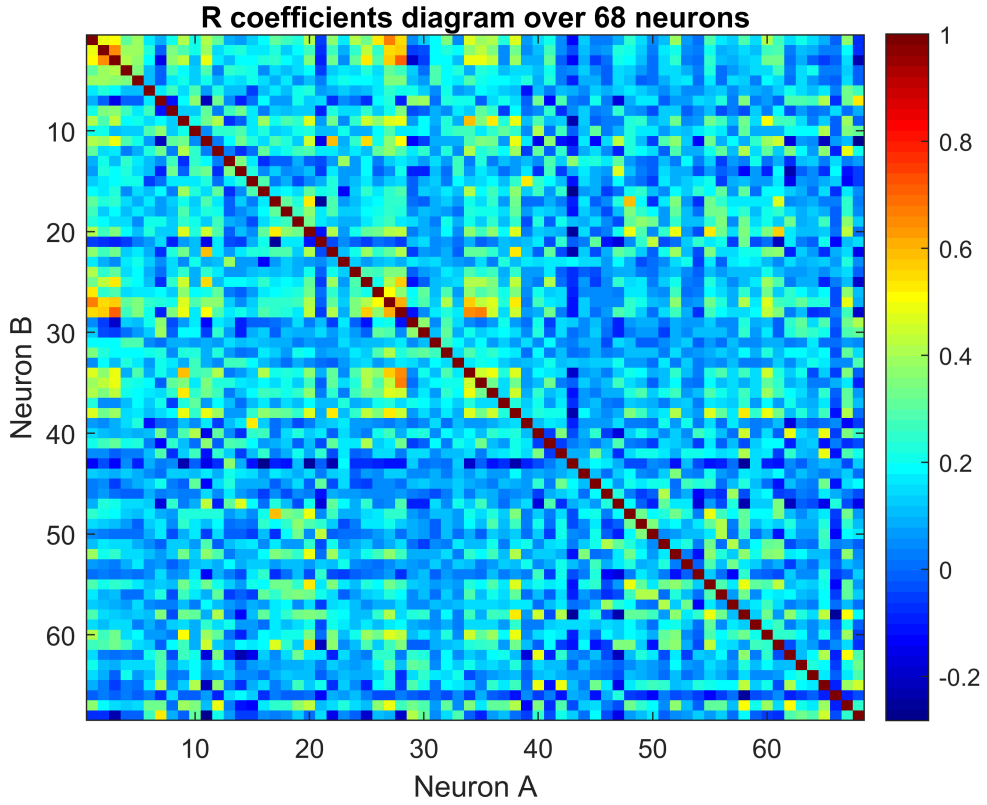
Note: $\rho(A,A)$ is the correlation coefficient of a variable A correlated with itself and its value is always 1. This is what is called the *auto-correlation.* The same applies to the coefficient $\rho(B,B)$.

4. Unidimensional neurons correlation

The intensity of several neurons has been measured over time and the goal is to see which ones are activated at the same time. This can be done by computing cross-correlations between all neurons and extracting the correlation coefficients.

Several functions can be used in MATLAB for this, like *corr(), xcorr(), corrcoef(), etc.* At the end it depends on the information that needs to be taken out of all this. To only see one value for each correlation in a *heatmap diagram*, the function *corrcoef()* is the best choice.

```
% Computation of the correlation coefficients
Rcoeff = corrcoef(series.Data); % Supported for code generation
% Rcoeff_matrix = corr(series.Data); % Not supported for code generation

% Graphical results
figure(2);
colormap(jet);
% Plot the heat diagram of the R coefficients
imagesc(Rcoeff);
% Display the colorbar beside the diagram
colorbar;
% Legend
title(['R coefficients diagram over ' num2str(NB_NEURONS) ' neurons']);
xlabel('Neuron A');
ylabel('Neuron B');
```



5. Bi-dimensional images neurons correlation

The intensity of several neurons on an image (256x256) has been measured over time and the goal is to see which ones are activated at the same time. This can be done by computing 2D cross-correlations between all images and extracting the correlation coefficients. A single function can be used in MATLAB for this and it is called *xcorr2()*.

An image is made of discrete pixels and the 2-D cross-correlation of an M-by-N matrix X, and a P-by-Q matrix Y, is a matrix, Z of size $M+P-1$ by $N+Q-1$. Its elements are given by:

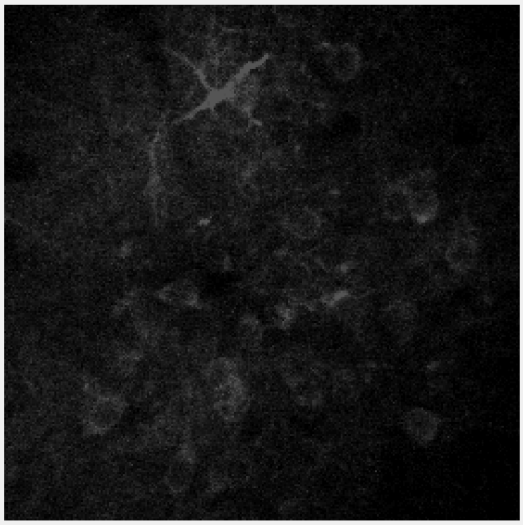
$$Z(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X(m,n)\cdot Y(m-k,n-l) \quad \text{, with} \quad \begin{matrix} -(P-1) \le k \le M-1 \\ -(Q-1) \le l \le N-1 \end{matrix} \qquad (7)$$

5.1 Initialization of data and visualization of a 2D neurons image

This shows several neurons on a selected image (256x256) after the data have been imported.

```
clear variables; close all; clc;
% Constants declaration
FILENAME      = '.\DATA\MA4_OPHN1_Baseline2_256_4x_xyz_Aligned.tif';
DATA_STRUCT   = imfinfo(FILENAME); % Extract TIFF file fields and data
NB_IMAGES     = 512;%length(DATA_STRUCT);
NB_DIMENSIONS = 2; % 2D images
X_RESOLUTION  = DATA_STRUCT(1).Width;
Y_RESOLUTION  = DATA_STRUCT(1).Height;
UINT8_MAX     = 255; % White value in uint8 for RGB
R_TRESHOLD    = 0.8;
WAIT_BAR_TEXT = ['Initializing data for ' num2str(NB_IMAGES) ' images...'];

% Frame vector definition
FRAME_FS    = 4; % Frame sampling frequency
DELTA_TS    = 1/FRAME_FS; % Frame sampling time
T_FRAME_MIN = 0*DELTA_TS;
T_FRAME_MAX = NB_IMAGES*DELTA_TS;
t_frame     = (T_FRAME_MIN:DELTA_TS:T_FRAME_MAX-DELTA_TS)';

% Initialization
image  = zeros(X_RESOLUTION,Y_RESOLUTION);
series = cell(NB_IMAGES,NB_DIMENSIONS);
bar_h  = waitbar(0,WAIT_BAR_TEXT,'Name','Initialize data',...
                 'CreateCancelBtn','setappdata(gcbf,''canceling'',1)');
display_data = true;

% Get raw data from TIFF file
for n=1:1:NB_IMAGES
  % Normalize the pixel encoding between 0 and 1 as a double
  image = double(imread(FILENAME,n))/UINT8_MAX;
  % Add the time info to each image
  series(n,:) = {t_frame(n),image};
  % Check for clicked Cancel button
  if getappdata(bar_h,'canceling')
```

```
    display_data = false;
    break;
  end
  % Display the remaining time to complete the full computation of data
  waitbar(n/NB_IMAGES,bar_h,WAIT_BAR_TEXT);
end
% Close the wait bar
delete(bar_h);

% Graphical results
if (display_data)
  % Test to extract one image and visualize it for debug purpose
  img = cell2mat(series(9,2));
  imshow(img);
  set(gcf,'Visible','on','Units','normalized','Outerposition',[0 0 1 1]);
else
  disp('The initialization of the data has been canceled');
end
```

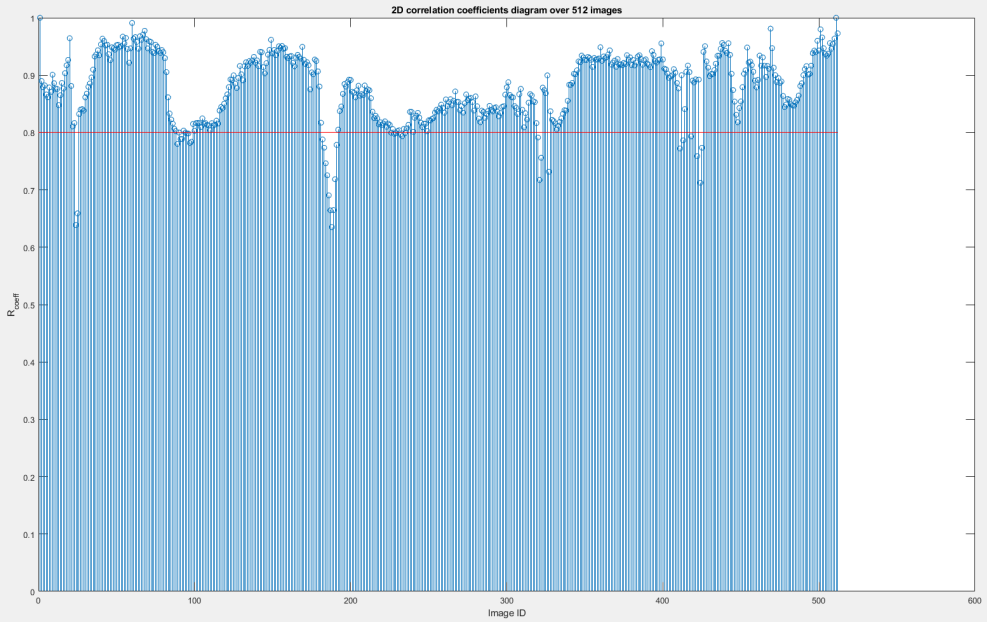


5.2 Computation of the bi-dimensional correlations

Notice how the spectral components can be identified clearly compared with the spectrogram approach. Fore more information on the resolution issues, see this URL:

```
WAIT_BAR_TEXT = ['Processing 2D correlations over ' num2str(NB_IMAGES) ' images...'];
% Initialization of output vectors
Vxcorr = cell(1,NB_IMAGES);
Vnorm  = zeros(1,NB_IMAGES);
bar_h  = waitbar(0,WAIT_BAR_TEXT,'Name','2D correlations',...
                 'CreateCancelBtn','setappdata(gcbf,''canceling'',1)');
display_data = true;
```

```
% Data processing and computation of the upper triangle cross-correlation matrix
for n=1:1:NB_IMAGES
  % Compute the upper triangle cross-correlation between the 1st and n^th images
  % As the result is symmetric along the main diagonal, it saves time and memory
  Vxcorr{1,n,:} = triu(xcorr2(cell2mat(series(1,2)),cell2mat(series(n,2))));
  Vnorm(n) = Vxcorr{1,n}(X_RESOLUTION,Y_RESOLUTION)/...
             Vxcorr{1,1}(X_RESOLUTION,Y_RESOLUTION);
  % Check for clicked Cancel button
  if getappdata(bar_h,'canceling')
    display_data = false;
    break;
  end
  % Display the remaining time to complete the full computation of data
  waitbar(n/NB_IMAGES,bar_h,WAIT_BAR_TEXT);
end
Vnorm = min(Vnorm,1); % Limit max correlation value to 1
% Close the wait bar
delete(bar_h);
% Graphical results
if (display_data)
  figure(1);
  stem(Vnorm); hold on;
  plot([0, NB_IMAGES],R_TRESHOLD*[1 1],'r');
  % Legend
  title(['2D correlation coefficients diagram over ' num2str(NB_IMAGES) ' images']);
  xlabel('Image ID');
  ylabel('R_{coeff}');
else
  disp('The processing of the images has been canceled');
end
```



2D neurons analysis using deep learning

Table of Contents

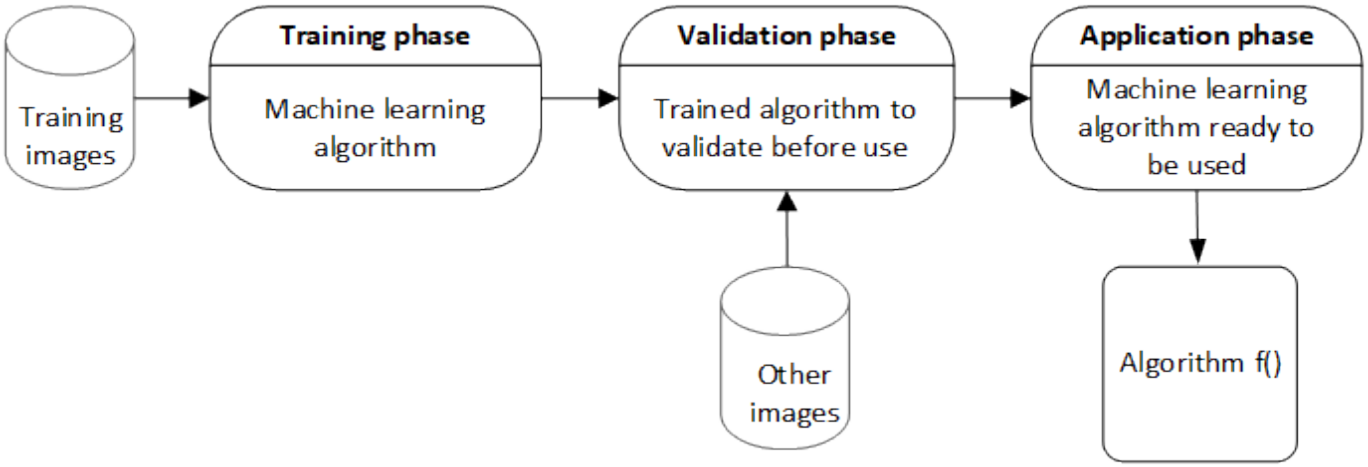

```
% Clean current MATLAB session
clear variables; close all; clc;
```

1. What is deep learning?

Deep learning is a branch of machine learning that teaches computers to do what comes naturally to humans: learn from experience. Machine learning algorithms use computational methods to “learn” information directly from data without relying on a predetermined equation as a model. Deep learning is especially suited for image recognition.

1.1 Deep learning workflow

First, training images have to be chosen and labeled correctly. In the *training phase*, a neural network architecture is selected based on the specific application and trained with the chosen labeled images. Once the neural network has been trained, it must go through the *validation phase* using images that were not used during the training phase. At the end, if the validation is successful, the trained algorithm can then be deployed to run in what is called the *application phase*. Here is a schematic of the different phases of a deep learning workflow:



2. Conversion of input data

The input raw data are represented by a *TIF* file containing many images. To properly process these images in MATLAB it is needed to split all of them into single images and save them in another format like *PNG* for example.

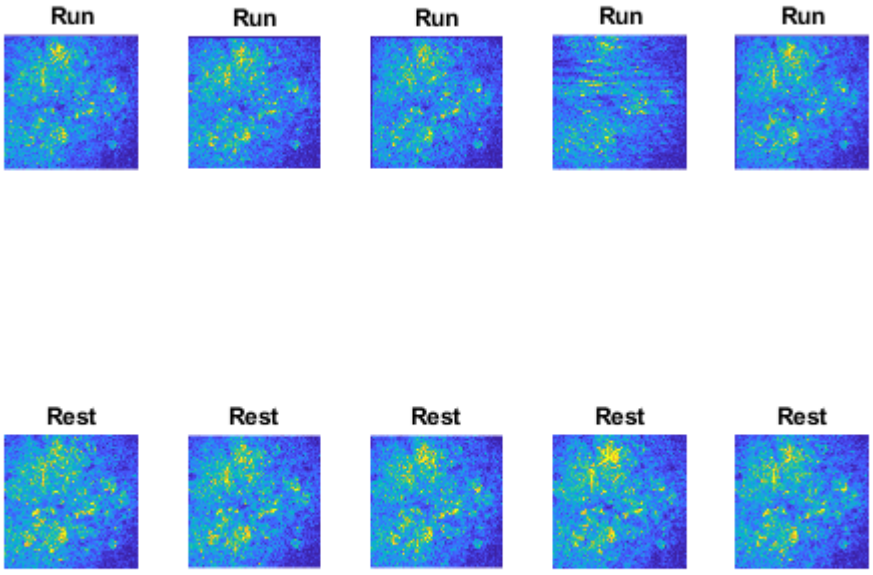
```
% Execute the following code only when the content of the main TIF file has changed
% Constants
DATA_FOLDER   = 'Data\';
DEEP_LEARNING = [DATA_FOLDER 'DL'];
NB_PIXELS     = 227;
% Check if the folder for the deep learning process exists
if ~exist(DEEP_LEARNING,'dir')
  mkdir(DEEP_LEARNING);
end
% Get images data from the main data file
imageFile = fullfile('Data','MA4_OPHN1_Baseline2_256_4x_xyz_Aligned.tif');
nbImages  = numel(imfinfo(imageFile));
waitBarText = ['Extract ' num2str(nbImages) ' images from the TIF file...'];
bar_h  = waitbar(0,waitBarText,'Name','Extract images',...
                 'CreateCancelBtn','setappdata(gcbf,''canceling'',1)');
display_data = true;
% Extract images from the main TIF file
for n=1:nbImages
  img = imread(imageFile,n);
  img = imresize(img,[NB_PIXELS,NB_PIXELS]);
  img = ind2rgb8(img,colormap);
  imwrite(img,fullfile(DEEP_LEARNING,"img"+sprintf('%06d',n)+".png"));
  % Check for clicked Cancel button
  if getappdata(bar_h,'canceling')
    display_data = false;
    break;
  end
  % Display the remaining time to complete the full computation of data
  waitbar(n/nbImages,bar_h,waitBarText);
end
% Close the wait bar
delete(bar_h);
```

3. Ground truth labeling

Putting labels on raw data has to be done before starting training the neural network. It consist of assigning a *state* to an image, like "ON" if a light is switched on, or "OFF" if the same light is switched off. In this case, it is needed to identify if the mouse is moving => "Run", or not => "Rest". This process has to be done manually, but the **Ground Truth Labeler** app can be used to help labeling ground truth data in a video or sequence of images.

```
% Execute the following code to create the ground truth only if Vnorm has changed
% Here it is either '1' => "Run" or 0 => "Rest"
% R_TRESHOLD = 0.8;
% mouseData  = timetable(seconds([series{:, 1}])',...
```

```
%                         Vnorm'>R_TRESHOLD,'VariableNames',"Activity");
% save("mouseData","mouseData");
% Contain the labeled ground truth data
load([DATA_FOLDER 'mouseData.mat'],'mouseData');
% Datastore to images
mouseImages = imageDatastore(DEEP_LEARNING);
mouseImages.Labels = categorical(mouseData.Activity,[false true],{'Run' 'Rest'});
% Split images between the training and verification data
[mouseTrainImgs,mouseTestImgs] = splitEachLabel(mouseImages,0.5,'randomized');
% Display some randomly selected images
plotSampleImages(mouseTrainImgs,mouseTrainImgs.Labels);
```



4. Neural network architecture

Artificial Neural Networks (ANN) are inspired by biological neural networks. It exists several dozens of architecture with pros and cons. The one that has been selected here is called **AlexNet**. It is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 8 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 227-by-227.

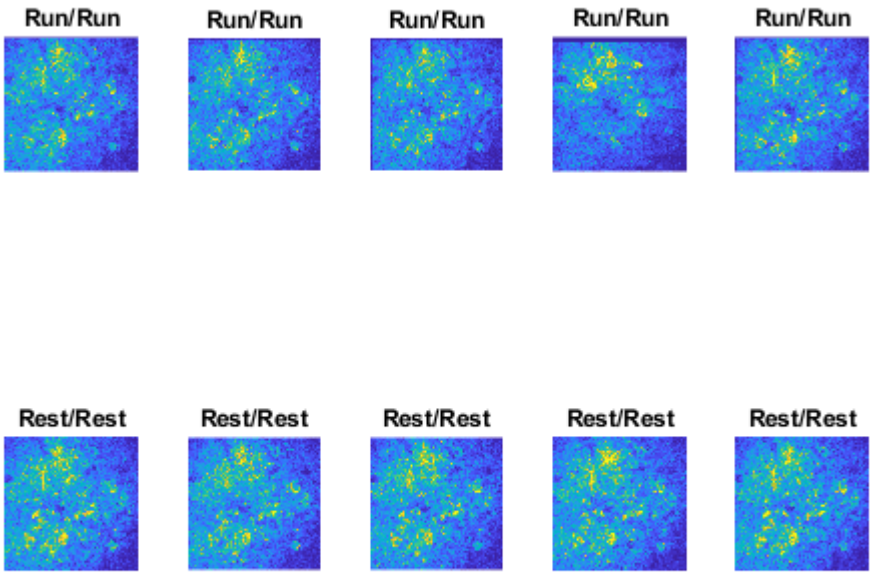
```
% Instantiation of a pre-trained AlexNet convolutional neural network
net = alexnet;
% Setup of the layers
layers = [net.Layers(1:end-3);fullyConnectedLayer(2);softmaxLayer;classificationLayer];
```

5. Training of the neural network

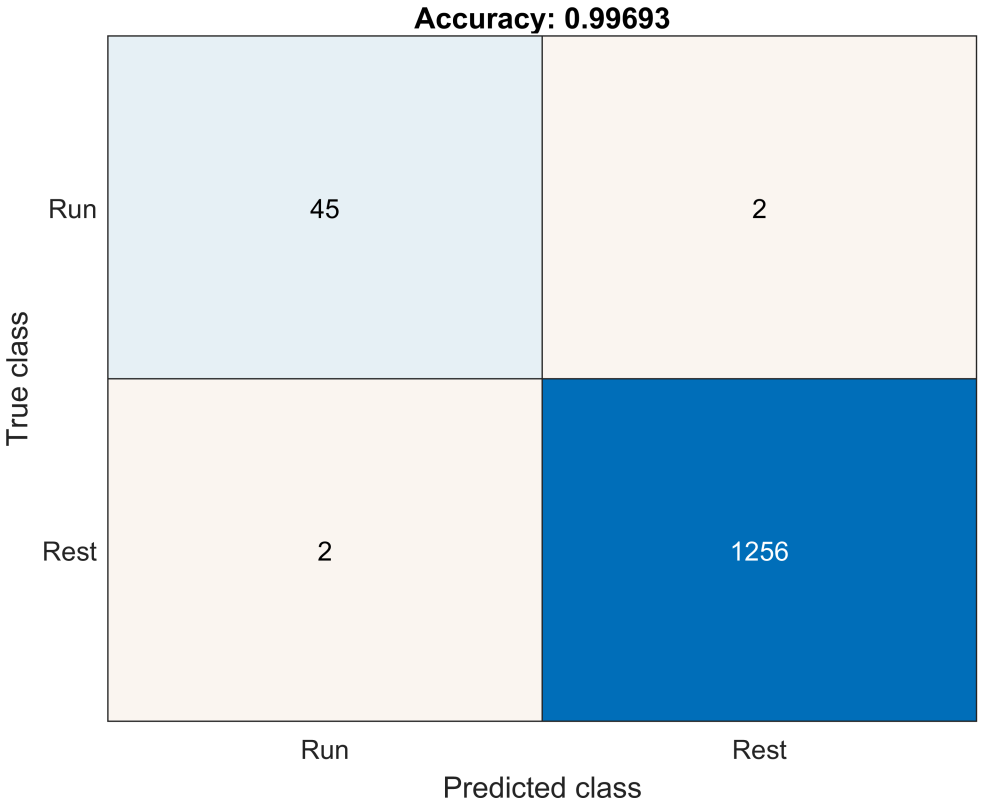
```
% Execute the following code only to train the neural network
% Will take a lot of time on a single CPU...
% tOpts = trainingOptions('sgdm','InitialLearnRate',0.001,'Plots','training-progress');
% mouseNet = trainNetwork(mouseTrainImgs,layers,tOpts);
% save([DATA_FOLDER 'mouseNet.mat'],'mouseNet');
load([DATA_FOLDER 'mouseNet.mat']);
```

6. Validation of the trained network

```
% Classify data using a trained deep learning neural network
modelledActivity = classify(mouseNet,mouseTestImgs);
plotSampleImages(mouseTestImgs,mouseTestImgs.Labels,modelledActivity);
```




```
figure(2);
confusionchart(mouseTestImgs.Labels,modelledActivity);
title("Accuracy: "+mean(mouseTestImgs.Labels == modelledActivity));
```


Accuracy: 0.99693
Run
Rest
True class
45
2
2
1256
Run
Rest
Predicted class