Laboratory exercise: multi-layer perceptrons

In this lab exercise you are going to study the performance of different MLP configurations when used to classify images. The data set proposed is the *CalTech 101 Silhouettes Data Set*, which is described and can be downloaded in the following link: https://people.cs.umass.edu/~marlin/data.shtml. The data set is named "caltech101_silhouettes_28.mat".

The set of images available represents 101 different silhouettes (e.g. bonsai, chair, elephant, etc.). Therefore, you have 101 possible output classes. On the other hand, each image is represented by 28x28 pixels, which corresponds to 784 classification features (input variables). The number of instances, i.e. number of images, available for this problem is 8671. Note that you will have to encode the labels with a one-hot encoding scheme.

The idea is to study the performance of different parameter configurations for a MLP neural network. In order to compute and report the performance of each configuration it is required to use the mean accuracy measure. Each configuration should be executed at least 3 times in order to get the mean accuracy value.

You should, at least, study the following parameters:

1) The next two configurations of the hidden and output layers transfer functions and cost function:
   1.1) logsig for the hidden layer, logsig for the output layer and mean squared error.
   1.2) logsig for the hidden layer, softmax for the output layer and cross-entropy.
2) Different number of hidden units: 50, 200 and 500.
3) Different percentage of training, validation and test data sets: 80/10/10, 40/20/40 and 10/10/80.

The rest of the parameters (learning rate, momentum, number of epochs, L2 regularization parameter - if used, etc) can be set a priori, but you have to do a previous search to find reasonable values. Additionally (and optionally), you can test other architectures or perform any modification to the base architecture described.

Write a brief document (four sheets maximum) that includes:

1) Description of the runs with the different configurations that you have performed.
2) Explain how you have selected the rest of parameters.
3) Those tables that you consider necessary to describe the results obtained for the different network configurations. Explain and reason the results presented in the tables.
4) Your own conclusions with respect the results obtained.
5) Include the *.m* files with your code.
6) **If you use ChatGPT (or another similar tool) in the document, indicate it every time it is used. We want to evaluate your work, not someone else's.**

IMPORTANT: It is always a good idea to test the hardware+software environment and make a good estimation of the execution times with enough advance.

NOTE: The exercise is designed to be done in Matlab, but you can implement it in another environment.

<u>Reminder of how to modify the basic parameters of a MLP in Matlab</u>

```
net.divideFcn = 'dividerand'; % divideFCN allows to change the way the data
                              % is divided into training, validation and test
                              % data sets.
net.divideParam.trainRatio = 0.1; % Ratio of data used as training set.
net.divideParam.valRatio = 0.1;   % Ratio of data used as validation set.
net.divideParam.testRatio = 0.8;  % Ratio of data used as test set.

net.trainParam.max_fail = 6;     % validation check parameter.
net.trainParam.epochs = 2000;    % number of epochs parameter.
net.trainParam.min_grad = 1e-5; % minimum gradient before stopping.

% You can define different transfer functions for each layer (layer{1} and
% layer{2}). Take a look to this parameter in Matlab to see all available
% functions.
net.layers{1}.transferFcn = 'logsig';
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'softmax';
net.layers{2}.transferFcn = 'logsig';

% Probably you will not need any additional processing in your network.
net.outputs{:}.processFcns = {};

% You can define different cost/performance functions. Take a look to this
% parameter in Matlab to see all available functions.
net.performFcn = 'crossentropy';
net.performFcn = 'mse';

% You can use different Training functions. Take a look to this parameter
% in Matlab to see all available functions.
% Notice that trainlm is often the fastest backpropagation algorithm in the
% toolbox, and is highly recommended as a first choice supervised algorithm
% for regression, although it does require more memory than other
% algorithms, as for example traingdm or traingdx.

net.trainFcn = 'trainlm';   % Levenberg-Marquardt
net.trainFcn = 'traingdm';  % Gradient Descent with momentum
net.trainFcn = 'traingdx';  % Gradient descent with momentum and adaptive
                            % learning rate backpropagation

% If you chose training functions that use momentum and learning rate, you
% need to set these parameters too.
net.trainParam.mc = 0.8;    % momentum parameter
net.trainParam.lr = 0.01;   % learning rate parameter
```