LABORATORY EXERCISE 2: NEURAL NETWORKS

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 in the following link: https://people.cs.umass.edu/~marlin/data.shtml.

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.

The data set is named "caltech101_silhouettes_28.mat" and can be downloaded from https://people.cs.umass.edu/~marlin/data.shtml. Note that you will have to encode the labels with a one-hot encoding scheme.

You are going to study the performance of different parameter configurations for a MLP neural network. In order to compute and report the performance of each designed network 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.

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.

IMPORTANT: It is a good idea to test the hardware+software environment and make a good estimation of the execution times with sufficient advance. On the other hand, remember that the rest of the parameters (learning rate, momentum and number of epochs) can be set a priori, but you have to do a previous search to find reasonable values.

Write a brief document (four pages 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 a *.m* file with your code.

NOTE: The exercise is designed to be done in Matlab. However, if for some reason it is not possible to do it in Matlab, it will be also accepted if it is done in another programming language.

Reminder of how to modify the basic parameters of the NN in Matlab

```
net.divideFcn = 'dividerand';% divideFCN allow 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 performance gradient

% you can define different transfer functions for each layer (layer{1} and
% layer{2}). You can take a look to this parameter in Matlab to see all
% functions available
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. You can take a look to
% this parameter in Matlab to see all functions available
net.performFcn='crossentropy';
net.performFcn='mse';

% you can use different Training functions. You can take a look to this
% parameter in Matlab to see all functions available.
% 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
```