```
%{
This function builds a patternet ANN model using given training data.
It then tests it using the given test data and produces a confusion
 matrix
with the results. The patternet created has a single hidden layer.

Arguments:
- `trainInputs`    -> samples to use for training (inc. validation
 data)
- `testInputs`     -> samples to use for testing the model
- `trainTargets`   -> class labels for the training set
- `testTargets`    -> class labels for the test set
- `hiddenLayerSize` -> number of neurons in the single hidden layer
- `trainingAlgo`    -> algorithm to use for training the NN

Returns:
- `accuracy`        -> the classification accuracy of the built model
 when
                        tested with the given unseen test set.
%}

function [accuracy]=nn(trainInputs, testInputs, trainTargets,
 testTargets, hiddenLayerSize, trainingAlgo)

    % Create a Pattern Recognition Network with the defined number of
 hidden layers.
    % `patternnet` is specific for pattern-recognition NNs
    net = patternnet(hiddenLayerSize, trainingAlgo);
%     net.trainParam.showWindow = 0;   % hide the training window
    %{
    patternnet() is specialized for pattern recognition problems.
    - Default training algo: Scaled conjugate gradient backpropagation
 (trainscg).
        * trainscg's goal: minimize a cost function.
    - Default loss cost function: Cross-entropy.
        * This function measures the performance of a classification
 model whose
        output varies between 0 and 1.
        * Cross-entropy loss increases as the prediction probability
 diverges
        from the output value.
        * Therefore, small values -> good performance, large values ->
 bad performance.
    %}

    % Set up Division of Data for Training and Validation.
    % The test subset has already been extracted.
    net.divideParam.trainRatio = 50/100;
    net.divideParam.valRatio = 50/100;

    % Standardise and normalise the input data.
```

```
   % standardisation shifts the data such that the center is 0 and
the
   % standard deviation is 1. Function normalises each column by
default.
   % 'range' makes all the values be between 0 and 1 (normalisation)
   trainInputs = normalize(trainInputs, 'range');
   testInputs = normalize(testInputs, 'range');

   % Train the Network
   [net, tr] = train(net, trainInputs, trainTargets);

   % ------------------------
   % Test the Network with the test subset from the current dataset
   % ------------------------
   actualTstOutputs = net(testInputs);

   %  compare the NN's predictions against the training set
   idealTstOutputs = testTargets;
   tstPerform = perform(net, idealTstOutputs, actualTstOutputs);

   sets_for_labels = [{'LGW'} {'RA'} {'RD'} {'SiS'} {'StS'}];

   % we need to convert the targets from Nx5 boolean values into a
single
   % string row/column to be able to run confusionchart
   for yy=1 : size(idealTstOutputs,2)
       % get the 5 1/0 values representing the class label
       current_ideal_class = idealTstOutputs(:,yy);
       current_actual_class = actualTstOutputs(:,yy);
       % find where the '1' is
       [~,I_ideal] = max(current_ideal_class);
       [~,I_actual] = max(current_actual_class);
       % get the corresponding string value of the class label
       idealTstOutputsSimplified(:,yy) = sets_for_labels(I_ideal);
       actualTstOutputsSimplified(:,yy) = sets_for_labels(I_actual);
   end
   % create the confusion matrix object to show and retrieve
   % classification accuracy from
   plotTitle = sprintf('ANN Confusion Matrix for %i
features',size(trainInputs,1));
   cm =
confusionchart(idealTstOutputsSimplified,actualTstOutputsSimplified,...
       'Title', plotTitle,...
       'RowSummary', 'absolute',...
       'ColumnSummary', 'absolute');

   % Calculate the classification accuracy from the confusion matrix
   % Need to first obtain the number of correct classifications, this
will
   % be equal to the sum of the values in the diagonal of the CM
   confusionMatrixResults = cm.NormalizedValues;
   correct_predictions = 0;
   for ii=1 : length(confusionMatrixResults)
```

```
        correct_predictions = correct_predictions +
 confusionMatrixResults(ii,ii);
    end
    accuracy = (correct_predictions/length(testTargets))*100;

    fprintf("\n-------------\nSummary:\n    Hidden layer neurons: %i
\n", hiddenLayerSize)
    fprintf("    Number of features: %i \n", size(trainInputs,1))
    fprintf("    ANN classification accuracy %f\n", accuracy)
    fprintf('    Patternnet performance: %f \n', tstPerform);
    fprintf('    num_epochs: %d, stop: %s\n-------------\n\n',
 tr.num_epochs, tr.stop);

end
```

Published with MATLAB® R2020a