**Assignment 2**

**(Neural Network Course)**

1. Implement non-linear perceptron algorithm for the classification using Hebbian Learning rule. The dataset (data55.mat) contains 4 features and the last column is the output (class label). You can use hold-out cross-validation (70, 10, and 20%) for the selection of training, validation and test instances. Evaluate accuracy, sensitivity and specificity measures for the evaluation of test instances. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).

2. Implement kernel perceptron algorithm for the classification. The dataset (data55.mat) contains 4 features and the last column is the output (class label). You can use hold-out cross-validation (70, 10, and 20%) for the selection of training, validation and test instances. Evaluate accuracy, sensitivity and specificity measures for the evaluation of test instances. (You can use RBF, and polynomial kernels). (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

3. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). Design a multilayer perceptron based neural network (two hidden layers) for the classification. You can use both holdout (70, 10, and 20%) and 5-fold cross-validation approaches for evaluating the performance of the classifier (individual accuracy and overall accuracy). You can select the number of hidden neurons of each hidden layer and other MLP parameters using grid-search method. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).

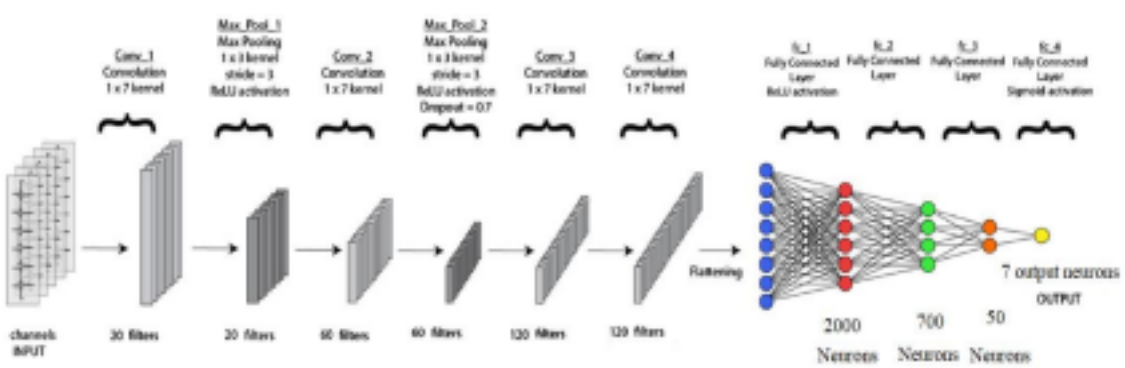
4. Implement the radial basis function neural network (RBFNN) for the classification problem. You can use Gaussian, multiquadric and linear kernel functions for the implementation. You can use both holdout (70, 10, and 20%) and 5-fold cross-validation approaches for evaluating the performance of the classifier (individual accuracy and overall accuracy). The dataset (data5.mat) contains 72 features and the last column is the output (class labels). (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).

5. Implement the stacked autoencoder based deep neural network for the classification problem. The deep neural network must contain 3 hidden layers from three autoencoders. You can use holdout (70, 10, and 20%) cross-validation technique for selecting, training and test instances for the classifier. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). For autoencoder implementation, please use back propagation algorithm which has been already taught in the class. Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

6. Implement an extreme learning machine (ELM) classifier for the classification. You can use Gaussian and tanh activation functions. Please select the training and test instances using 5-fold cross validation technique Evaluate individual accuracy and overall accuracy. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

7. Implement a deep neural network, which contains two hidden layers (the hidden layers are obtained from the ELM-autoencoders). The last layer will be the ELM layer which means the second hidden layer feature vector is used as input to the ELM classifier. The network can be called as deep layer stacked autoencoder based extreme learning machine. You can use holdout approach (70, 10, 20%) for evaluating the performance of the classifier. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

8. Implement a multi-channel 1D deep CNN architecture for the seven-class classification task. The input and the class labels are given in .mat file format. There is a total of 17160 number of instances present in both input and class-label data files. The input data for each instance is a multichannel time series (12-channel) with size as (12 ×800). The class label for each multichannel time series instance is given in the class_label.mat file. You can select the training and test instances using hold out cross-validation (70% training, 10% validation, and 20% testing). The architecture of the multi channel deep CNN is given as follows. The number of filters, length of each filter, and number of neurons in the fully connected layers are shown in the following figure. Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are allowed)



9. The two-dimensional time-frequency images of class1, class2, class3 are given in the folders as class1.zip, class2.zip, class3.zip, respectively. Design a 2D deep CNN classifier for the three-class classification. Evaluate the classification performance using hold-out cross-validation (70% training, 10% validation, 20% testing), and 10-fold cross-validation methods. Evaluate individual accuracy and overall accuracy for the multiclass CNN classifier. You can consider 4 convolutional layer, three pooling layer, and 5 fully connected layers. You can select the number of filters, stride for convolution and pooling layers, and number of neurons for fully connected layers as per your own choice. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are allowed). You can apply dropout, batch normalization, and regularization to improve the classification performance.