

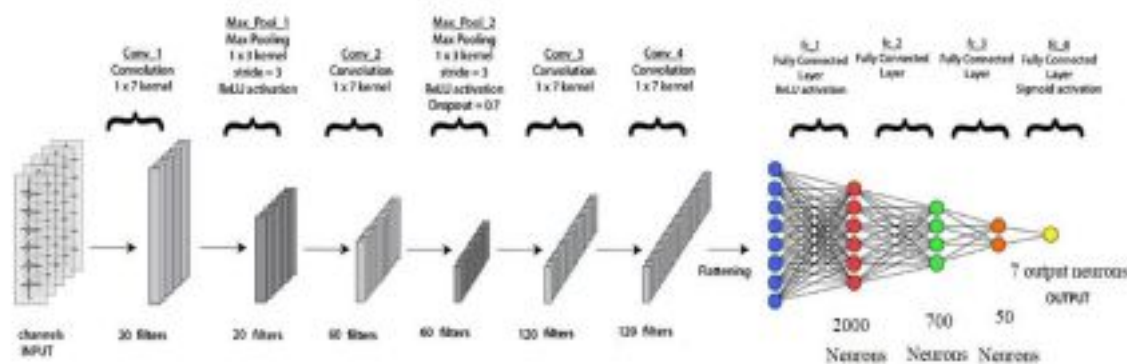
## Assignment 2

### (Neural Network and Fuzzy Logic Course)

1. Implement non-linear 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. (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 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 support vector machine (SVM) classifier for the multi-class classification task. You can use one vs one and one vs all multiclass coding methods to create binary SVM models. Implement the SMO algorithm for the evaluation of the training parameters of SVM such as Lagrange multipliers. 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)
9. 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)



10. Implement the hybrid fuzzy deep neural network (HFDNN) for the six-class classification task. The

input and output instances for the HFDNN are given in input\_a2.mat and label.mat files, respectively. For a single instance, the input size is 72. There is a total of 720 instances given in the input and label datasets. You can select the training and test instances using hold-out cross validation (70%training, 10% validation, and 20% testing). The HFDNN architecture shown in Fig. 1 has neural network hidden layers, fuzzy membership and rule layers, and a fusion layer. The descriptions of the HFDNN architecture are given in reference. Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

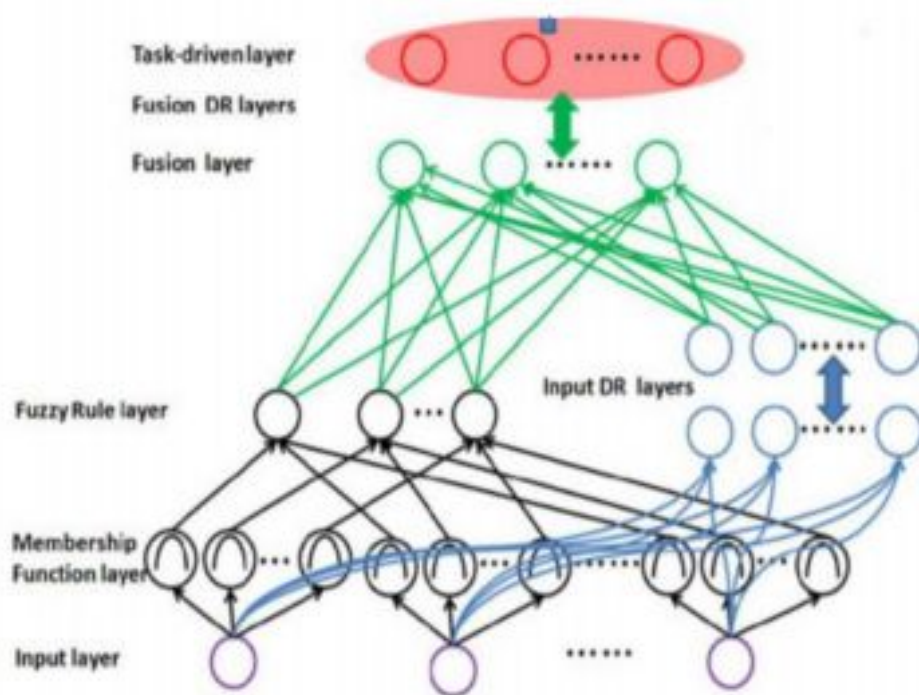


Fig.1 Architecture of hybrid fuzzy deep neural network [1].

For understanding hybrid fuzzy deep neural network, please follow the following article.

Deng, Yue, Zhiquan Ren, Youyong Kong, Feng Bao, and Qionghai Dai. "A hierarchical fused fuzzy deep neural network for data classification." *IEEE Transactions on Fuzzy Systems* 25, no. 4 (2016): 1006-1012.

11. Please write in your own words that what you have learned by solving the Assignment 2.