The higher-order extreme learning machine algorithm implementation in MATLAB covers the following seven single-layer neural network types:

- Networks with low-order hidden layer neurons and low-order output layer neurons (classic ELM).
- Networks with higher-order hidden layer neurons and low-order output layer neurons.
- Networks with low-order hidden layer neurons and higher-order output layer neurons.
- Networks with higher-order hidden layer neurons and higher-order output layer neurons.
- Networks with multi-cube hidden layer neurons and low-order output layer neurons.
- Networks with low-order hidden layer neurons and multi-cube output layer neurons.
- Networks with multi-cube hidden layer neurons and multi-cube output layer neurons.

The user places the ".csv" or excel format datasets inside the dataset folder and can use two different conversion functions for converting and normalizing them into MATLAB dataset format. These functions are "convertAndNormalizeExcelToDataset.m" for converting a single input file and "convertAndNormalizeTrainTestExcelFilesToDataset.m" for converting dataset files which are divided into training and test sets. Then, the produced ".mat" files, which contain the normalized versions of the dataset, can be divided into k-folds using one of the following three functions.

- The "createKFoldCrossValidationDataSet.m" function which divides a dataset into training/test sets.
- The "createKFoldCrossValidationDataSet2SetsAnd3Sets.m" function which receives as input a normalized dataset and creates two output files. This function initially creates a separate test set. Then, the first file contains a divided version of the dataset into training/test sets without using k-fold cross-validation. The second file further divides the training set into training/validation sets using k-fold cross-validation.
- The "createKFoldCrossValidationDataSetSeparateSets2SetsAnd3Sets.m" has the same functionality as the previous function, with the only difference it receives as input a normalized dataset that is already divided into training and test sets.

The program supports multi-core systems and is executed by running the "mainMultiCore.m" file. The results are automatically saved in the "results" folder and can be displayed by running the "displayResults.m" function. Finally, the user-defined parameters for each function mentioned above are explained in comment form inside each function.