**Technical Report**

**Technical Report**

**Document Deformation**

|  |  |
| --- | --- |
| Prepared by: | Eng. Mohammad Sakka |
|  |  |
| Date: | 20/3/2021 |
|  |  |

**REPORT SENSITIVITY**

Does the report have any of the following sensitivities?

Intended for journal publication YES

Results are incomplete NO

Commercial/IP concerns NO

**pseudocodes**

1. Q-Learning with KNN
2. Input**:**
3. discountFactor
4. learningRate
5. oldQTableWeight
6. k
7. dataset % sequential dataset
8. chunkLength % all chunk records have the same texture type
9. Output**:**
10. bestFeature % as timeseries for each chunk
11. accTimeSer % accuracy time series
12. start**:**
13. **[**Features,Labels**]** **=** decomposeDataset**(**dataset**)**
14. numberOfStates **=** numberOfFeaturesTypes
15. numberOfActions **=** numberOfFeaturesTypes
16. Q **=** zeros**(**numberOfActions**)** % Q-Table
17. numberOfChunks **=** length**(**dataset**)/**chunkLength
18. **for** i **=** 1**:**numberOfChunks
19. chunk **=** getChunk**(**dataset**,**i**,**chunkLength**)**
20. **for** s **=** 1**:**numberOfStates
21. % apply Knn predection algorithm on this chunk based on Features{s}
22. stateAcc **=** findTestingAcc**(**chunkFeatures**{**s**},**chunkLabels**,**k**)**
23. **for** a **=** 1**:**numberOfActions
24. % apply Knn predection algorithm on this chunk based on Features{a}
25. actionAcc **=** findTestingAcc**(**chunkFeatures**{**a**},**chunkLabels**,**k**)**
26. reward **=** actionAcc **-** stateAcc
27. Q**(**s**,** a**)** **=** oldQTableWeight **\*** Q**(**s**,** a**)** **+** **(**1**-**oldQTableWeight**)** **\*** ...
28. **(**learningRate **\*** **(**reward **+** discountFactor **\*** max**(**Q**(**s**,**otherStates**))** **-** Q**(**s**,** a**)))**
29. **end**
30. **end**
31. bestFeaturesTimeSer{i} = max**(**max**(**Q**)) %** bestFeature in this chunk
32. accTimeSer**(**i**)** **=** accuracy of bestFeatures
33. **end**
34. **end**
35. %%%%%%%%%%
36. KNN pseudocode**:**
37. Input**:**
38. chunkFeatures
39. chunkLabels
40. k
41. Output**:**
42. predictedLabels
43. start**:**
44. **for** r **=** 1**:**numberOfRecords
45. compute the equlidean distance between this record and all other records
46. nearestRecs **=** take the nearest k records to the current record
47. prevailing record **=** find the prevailing record of nearestRecs
48. predictedLabels**(**r**)** **=** prevailing record
49. **end**
50. **end**

2- Q-Learning with KNN (second version)

**-** 1 Input**:**

**-** 2 discountFactor

**-** 3 learningRate

**-** 4 oldQTableWeight

**-** 5 k

**-** 6 dataset % sequential dataset

**-** 7 chunkLength % all chunk records have the same texture type

**-** 8 Output**:**

**-** 9 bestFeature % as timeseries for each chunk

**-** 10accTimeSer % accuracy time series

**-** 11start**:**

**-** 12**[**Features Labels**]** **=** decomposeDataset**(**dataset**)**

**-** 13numberOfStates **=** numberOfFeaturesTypes

**-** 14numberOfActions **=** numberOfFeaturesTypes

**-** 15Q **=** zeros**(**numberOfActions**)** % Q-Table

**-** 16numberOfChunks **=** length**(**dataset**)/**chunkLength

**-** 17**for** i **=** 1**:**numberOfChunks

**-** 18 chunk **=** getChunk**(**dataset**,**i**,**chunkLength**)**

**-** 19 **for** s **=** 1**:**numberOfStates

**-** 20 % apply Knn predection algorithm on this chunk based on Features{s}

**-** 21 % if the chunk is the first chunk then consider it the teraining and testing else

**-** 22 % consider the previous chunk as training and the current as testing

**-** 23 **if** i **==**1

**-** 24 stateAcc **=** findKnnAcc**(**chunkFeatures**{**s**},**chunkLabels**,**k**)**

**-** 25 **else**

**-** 26 stateAcc **=** findKnnAcc2**(**chunkFeatures**{**s**},**chunkLabels**,**previousChunk**,**k**)**

**-** 27 **end**

**-** 28 **for** a **=** 1**:**numberOfActions

**-** 29 % apply Knn predection algorithm on this chunk based on Features{a}

**-** 30 % if the chunk is the first chunk then consider it the teraining and testing else

**-** 31 % consider the previous chunk as training and the current as testing

**-** 32 **if** i **==**1

**-** 33 actionAcc **=** findKnnAcc**(**chunkFeatures**{**a**},**chunkLabels**,**k**)**

**-** 34 **else**

**-** 35 actionAcc **=** findKnnAcc2**(**chunkFeatures**{**a**},**chunkLabels**,**previousChunk**,**k**)**

**-** 36 **end**

**-** 37 reward **=** actionAcc **-** stateAcc

**-** 38 Q**(**s**,** a**)** **=** oldQTableWeight **\*** Q**(**s**,** a**)** **+** **(**1**-**oldQTableWeight**)** **\*** ...

**-** 39 **(**learningRate **\*** **(**reward **+** discountFactor **\*** max**(**Q**(**s**,**otherStates**))** **-** Q**(**s**,** a**)))**

**-** 40 **end**

**-** 41 **end**

**-** 42 bestFeature in this chunk **=** max**(**max**(**Q**))**

**-** 43 accTimeSer**(**i**)** **=** accuracy of bestFeatures

**-** 44 previousChunk **=** currentChunk

**-** 45 **end**

**-** 46 **end**

**-** 47 %%%%%%%%%%

**-** 48 KNN second version pseudocode**:**

**-** 49 Input**:**

**-** 50 chunkFeatures

**-** 51 previousChunk

**-** 52 k

**-** 53 Output**:**

**-** 54 predictedLabels

**-** 55 start**:**

**-** 56 **for** r **=** 1**:**numberOfRecords

**-** 57 compute the equlidean distance between this record and all records in the previousChunk

**-** 58 nearestRecs **=** take the nearest k records to the current record

**-** 59 prevailing record **=** find the prevailing record of nearestRecs

**-** 60 predictedLabels**(**r**)** **=** prevailing record

**-** 61 **end**

**-** 62 **end**

3- QL-ELM

**-** 1 Input**:**

**-** 2 discountFactor

**-** 3 learningRate

**-** 4 oldQTableWeight

**-** 5 dataset % sequential dataset

**-** 6 chunkLength % all chunk records have the same texture type

**-** 7 numberOfFeatures

**-** 8 Output**:**

**-** 9 bestFeature % as timeseries for each chunk

**-** 10accTimeSer % accuracy time series

**-** 11start**:**

**-** 12% building ELM Learner using the N,R,and T data for each type of features

**-** 13trainingData **=** N **,**R and T data

**-** 14**for** f **=** 1**:**numberOfFeatures

**-** 15 Learner**{**f**}** **=** buildElmLearner**(**trainingData**)**

**-** 16end

**-** 17% performing Q-Learning

**-** 18testingData **=** the rest dataset

**-** 19**[**Features Labels**]** **=** decomposeDataset**(**testingData**)**

**-** 20numberOfStates **=** numberOfFeaturesTypes

**-** 21numberOfActions **=** numberOfFeaturesTypes

**-** 22Q **=** zeros**(**numberOfActions**)** % Q-Table

**-** 23numberOfChunks **=** length**(**dataset**)/**chunkLength

**-** 24

**-** 25**for** i **=** 1**:**numberOfChunks

**-** 26 chunk **=** getChunk**(**dataset**,**i**,**chunkLength**)**

**-** 27 **for** s **=** 1**:**numberOfStates

**-** 28 % apply Knn predection algorithm on this chunk based on Features{s}

**-** 29 stateAcc **=** findTestingAcc**(**Learner**{**s**},**chunkFeatures**{**s**},**chunkLabels**)**

**-** 30 **for** a **=** 1**:**numberOfActions

**-** 31 % apply Knn predection algorithm on this chunk based on Features{a}

**-** 32 actionAcc **=** findTestingAcc**(**Learner**{**a**},**chunkFeatures**{**a**},**chunkLabels**)**

**-** 33 reward **=** actionAcc **-** stateAcc

**-** 34 Q**(**s**,** a**)** **=** oldQTableWeight **\*** Q**(**s**,** a**)** **+** **(**1**-**oldQTableWeight**)** **\*** ...

**-** 35 **(**learningRate **\*** **(**reward **+** discountFactor **\*** max**(**Q**(**s**,**otherStates**))** **-** Q**(**s**,** a**)))**

**-** 36 **end**

**-** 37 **end**

**-** 38 bestFeature in this chunk **=** max**(**max**(**Q**))**

**-** 39 accTimeSer**(**i**)** **=** accuracy of bestFeatures

**-** 40**end**

**-** 41end

**-** 42%%%%%%%%%%

**-** 43% for findTestingAcc we just apply ELM predection procedure1

4- QL-OSELM

1 **-**Input**:**

2 **-**discountFactor

3 **-**learningRate

4 **-**oldQTableWeight

5 **-**dataset % sequential dataset

6 **-**chunkLength % all chunk records have the same texture type

7 **-**numberOfFeatures

8 **-**testingRatio

9 **-**Output**:**

10**-**bestFeature % as timeseries for each chunk

11**-**accTimeSer % accuracy time series

12**-**start**:**

13**-**% performing Q-Learning

14**-[**Features Labels**]** **=** decomposeDataset**(**testingData**)**

15**-**numberOfStates **=** numberOfFeaturesTypes

16**-**numberOfActions **=** numberOfFeaturesTypes

17**-**Q **=** zeros**(**numberOfActions**)** % Q-Table

18**-**numberOfChunks **=** length**(**dataset**)/**chunkLength

19**-for** i **=** 1**:**numberOfChunks

20**-** chunk **=** getChunk**(**dataset**,**i**,**chunkLength**)**

21**-** numberOfTraining **=** **(**1**-**testingRatio**)** **\*** length**(**chunk**)**

22**-** trainingData **=** chunk**(**1**:**numberOfTraining**)**

23**-** testingData **=** chunk**(**numberOfTraining**+**1**:**end**)**

24**-** **if** i**==**1

25**-** **[**Learner**{**s**}]** **=** buildOselmLearner**(**trainingData**{**s**})**

26**-** **else**

27**-** **[**Learner**{**s**}]** **=** updateOselmLearner**(**trainingData**{**s**})**

28**-** **end**

29**-** **for** s **=** 1**:**numberOfStates

30**-** % apply OSELM predection algorithm on this chunk based on Features{s}

31**-** stateAcc **=** findTestingAcc**(**Learner**{**s**},**testingData**{**s**},**chunkLabels**)**

32**-** **for** a **=** 1**:**numberOfActions

33**-** % apply OSELM predection algorithm on this chunk based on Features{a}

34**-** actionAcc **=** findTestingAcc**(**Learner**{**a**},**testingData**{**a**},**chunkLabels**)**

35**-** reward **=** actionAcc **-** stateAcc

36**-** Q**(**s**,** a**)** **=** oldQTableWeight **\*** Q**(**s**,** a**)** **+** **(**1**-**oldQTableWeight**)** **\*** ...

37**-** **(**learningRate **\*** **(**reward **+** discountFactor **\*** max**(**Q**(**s**,**otherStates**))** **-** Q**(**s**,** a**)))**

38**-** **end**

39**-** **end**

40**-** bestFeature in this chunk **=** max**(**max**(**Q**))**

41**-** accTimeSer**(**i**)** **=** accuracy of bestFeatures

42**-end**

43**-end**

44**-**%%%%%%%%%%

45**-**% for findTestingAcc we just apply OSELM predection procedure1

5- QL-OSELM Second Version

Similar to the first version, but use the next chunk for predection and the previous chunk for training but the first chunk was used as training and testing

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* when comparing with traditional KNN, we perform the same methodology that used in QL-KNN for testing in both first and second version
* when comparing with traditional ELM, we perform the same methodology that used in QL-ELM for testing and used N,R and T for training
* when comparing with traditional OSELM, we perform the same methodology that used in QL-OSELM for testing in both first and second version

**Results**

Parameters for all algorithms

Dataset: A

K = 4 // for knn

numberOfHiddenNeurons = length(feature vector) for each feature type // in case of elm and oselm

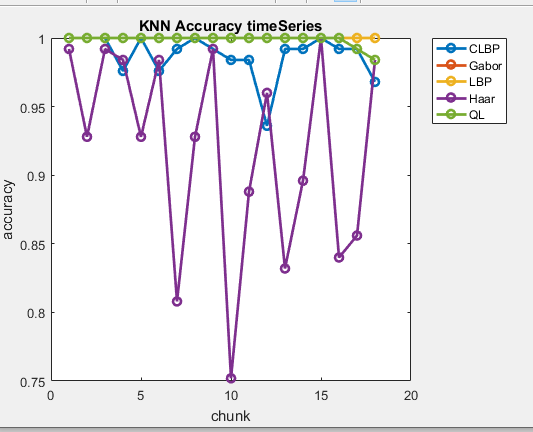
discountFactor = 0.8

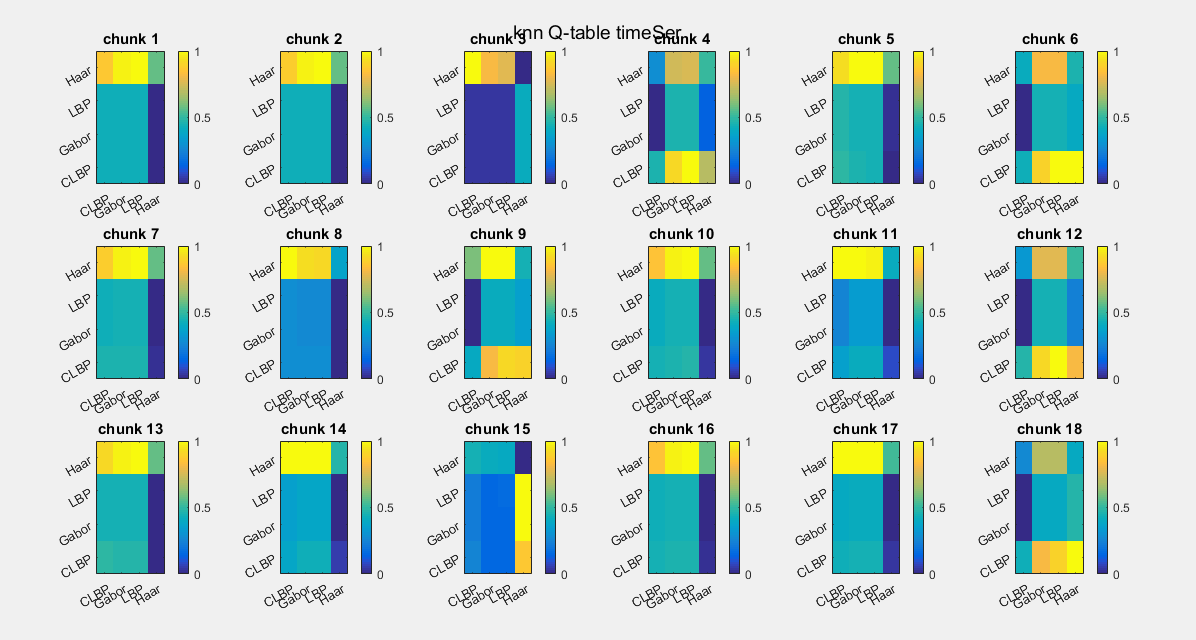
learningRate = 0.33

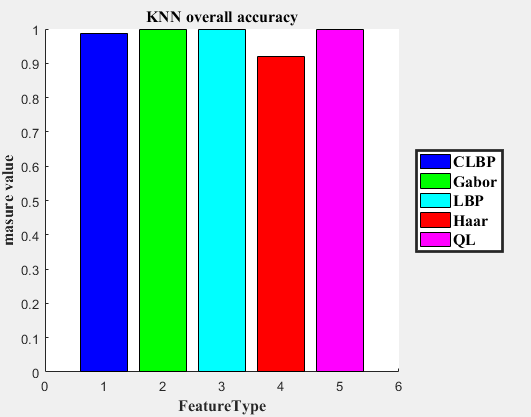
oldQTableWeight = 0.1

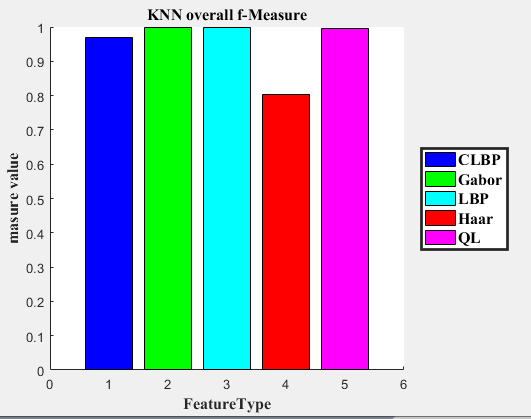
chunkLength = 50 // it is determined based on the dataset

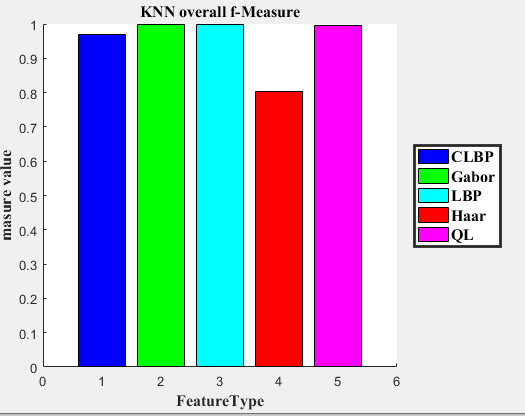
1. KNN and QL-KNN first version

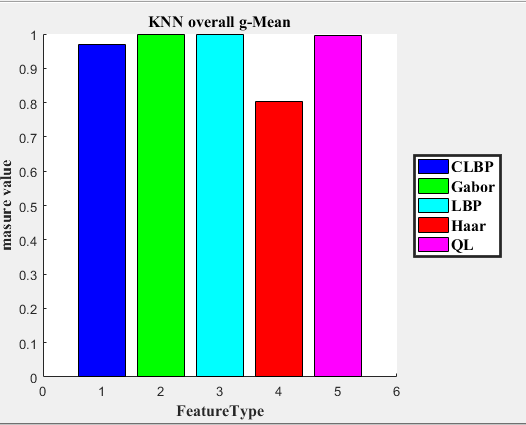


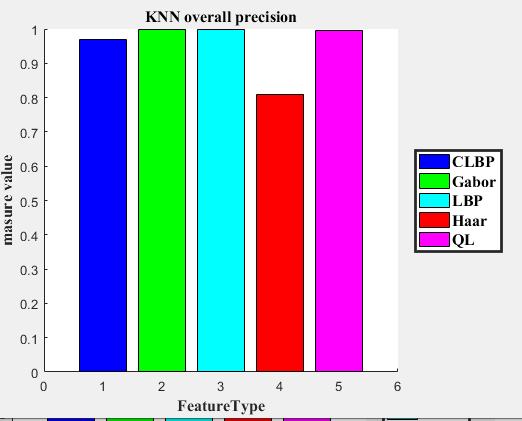


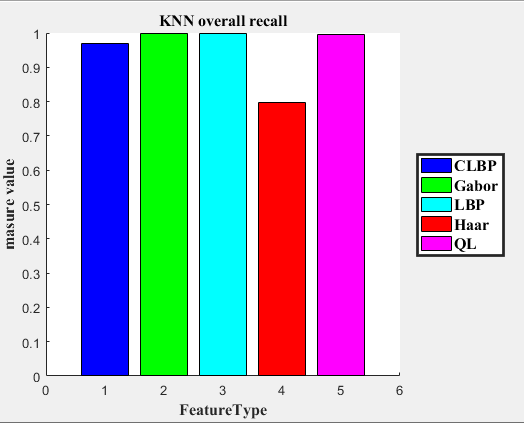


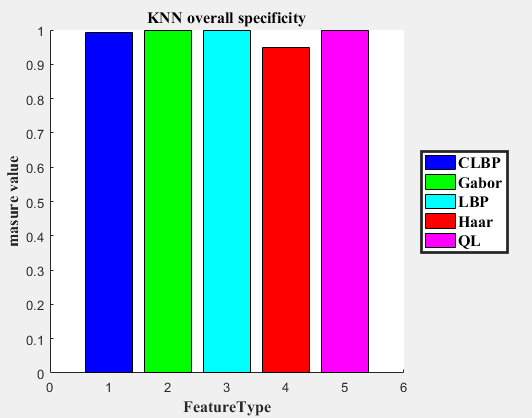


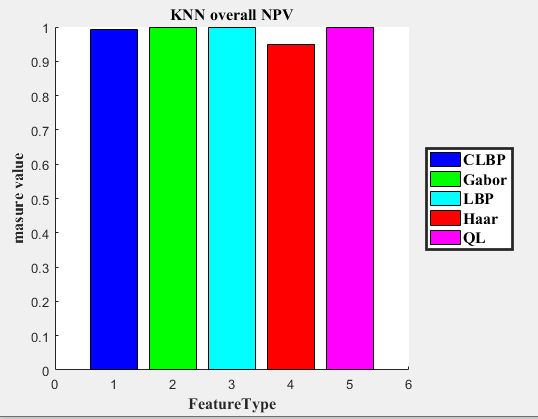




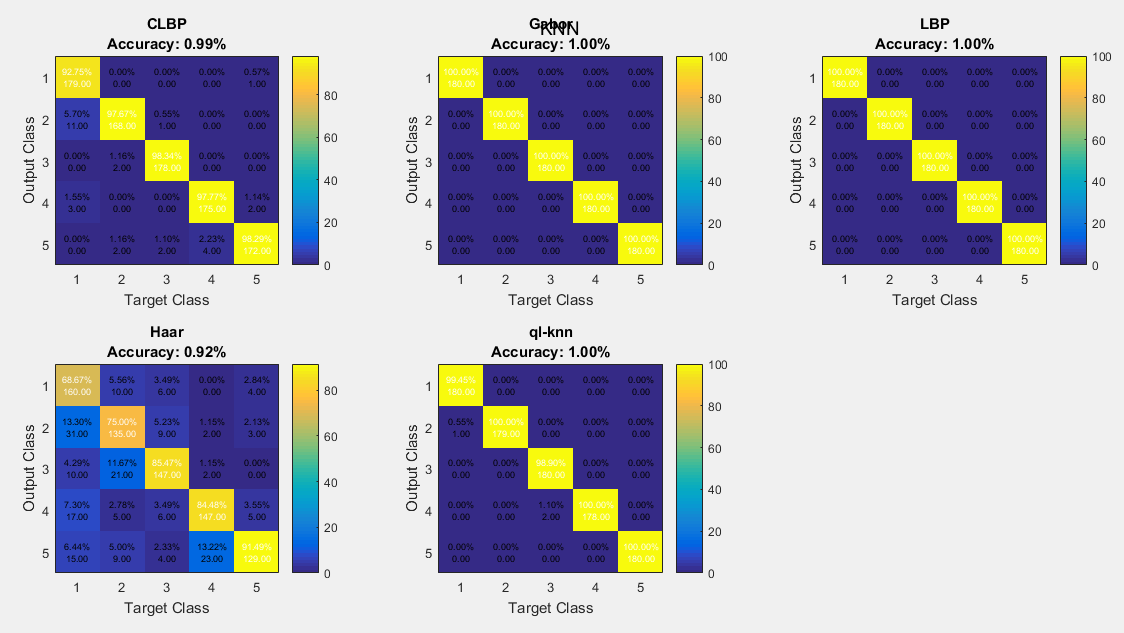




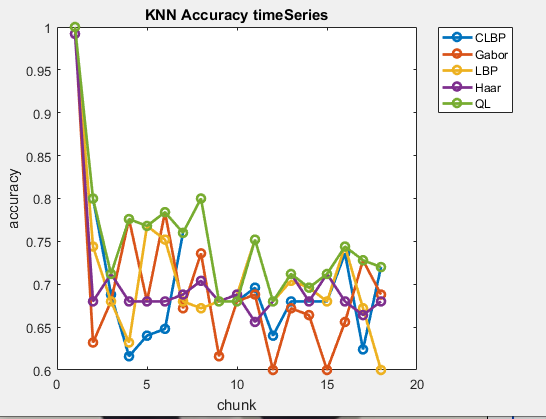


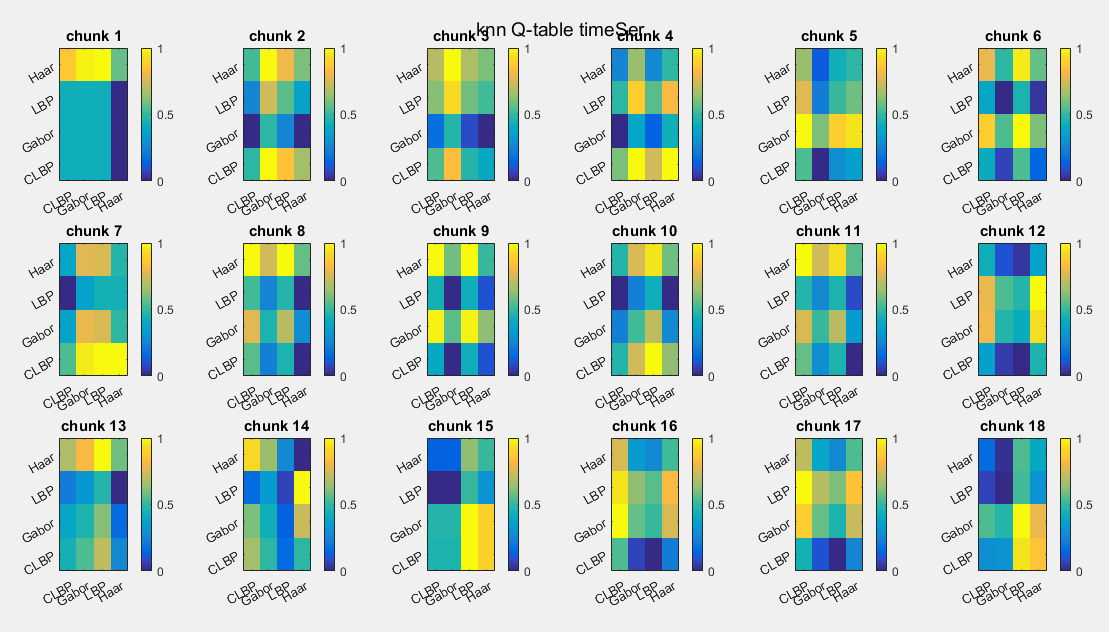


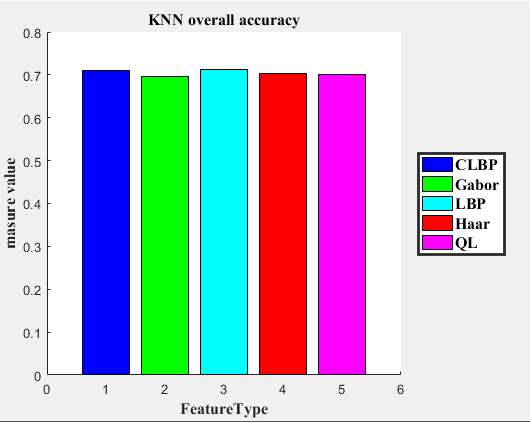
Confusion Matrix

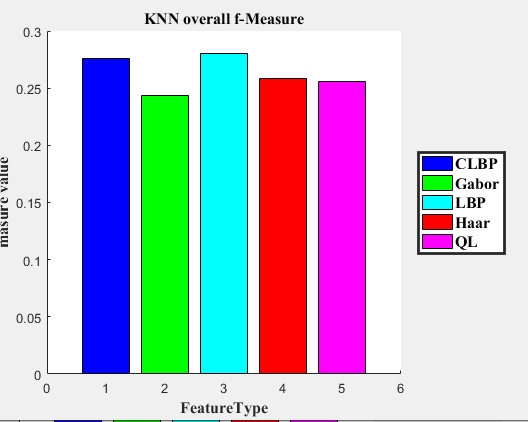


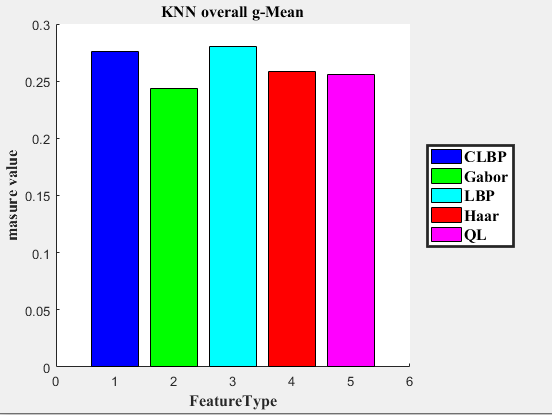
1. KNN and QL-KNN second version

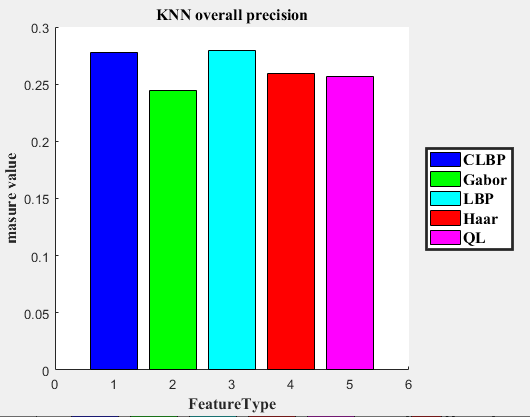


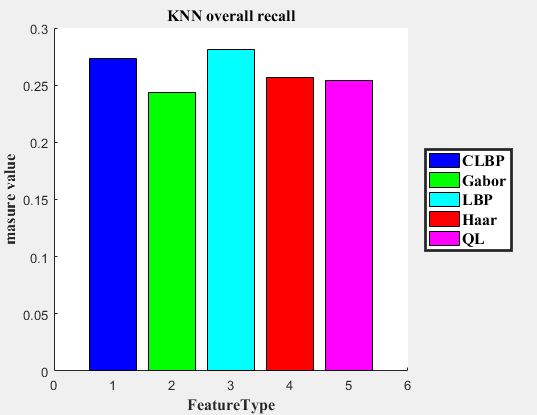


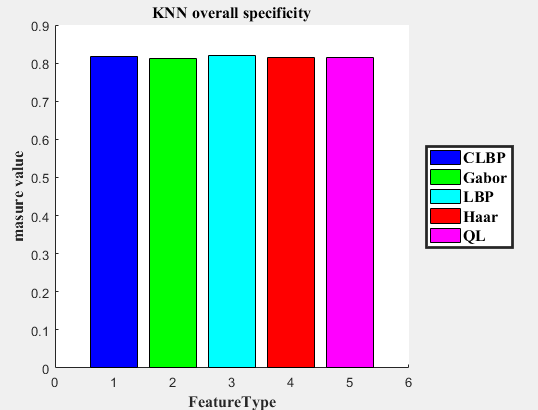


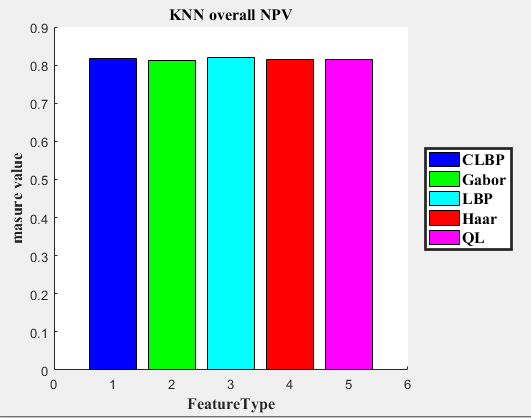




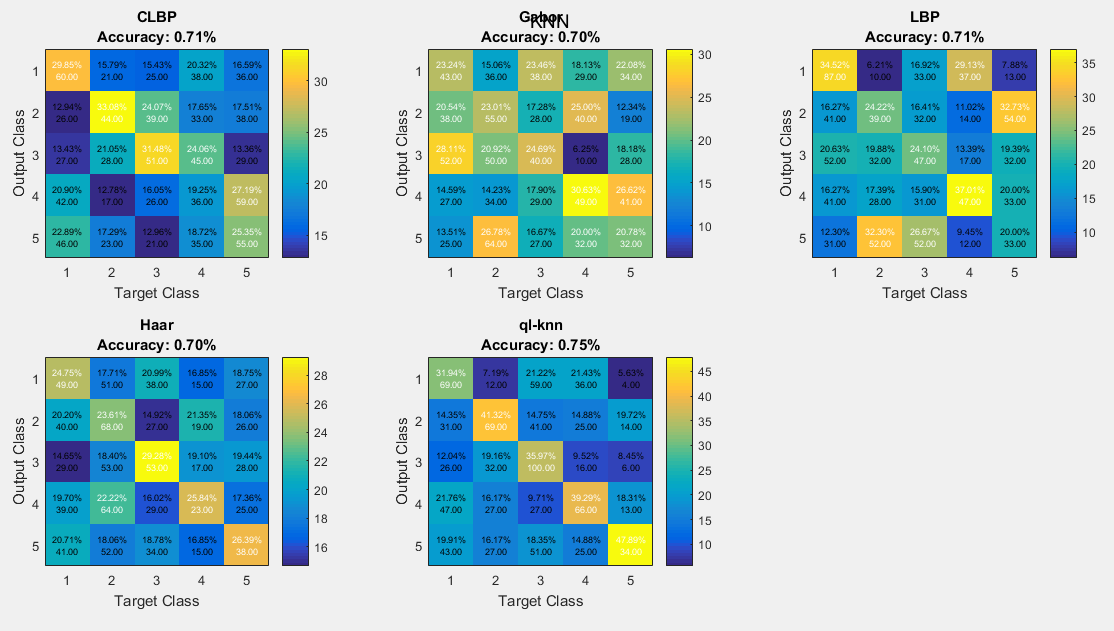




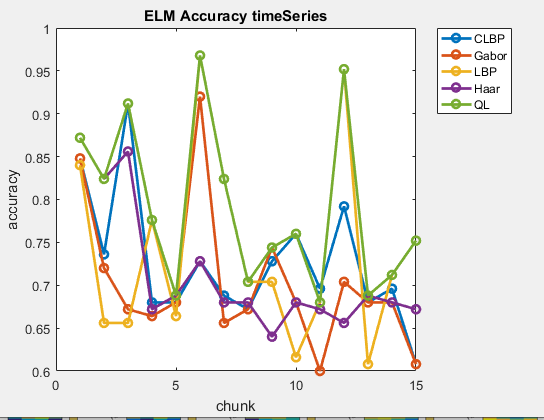


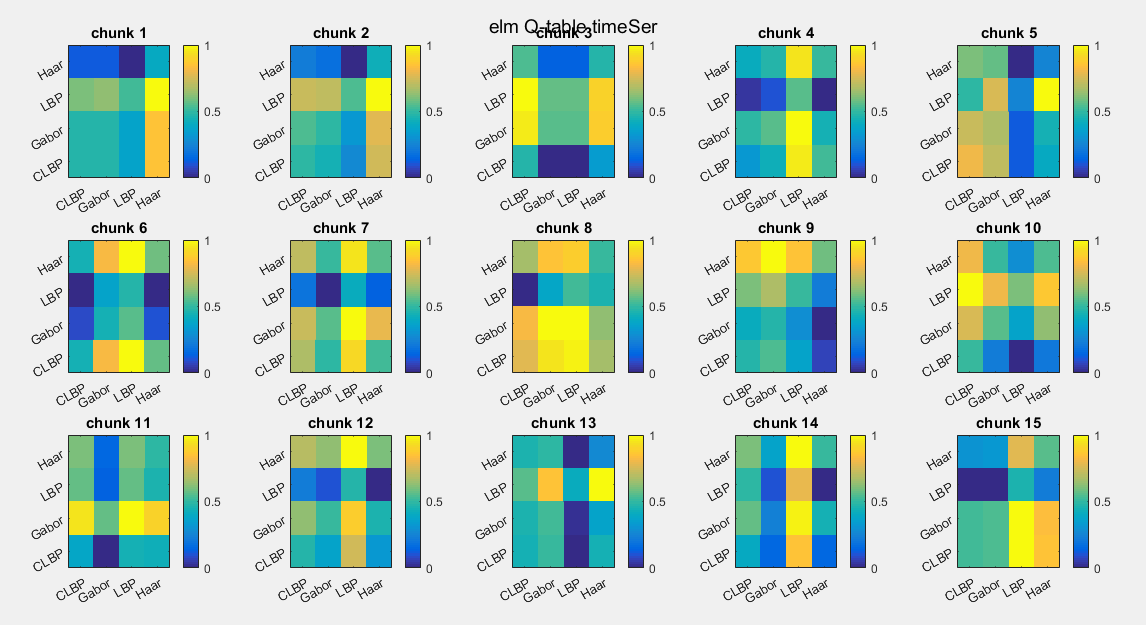


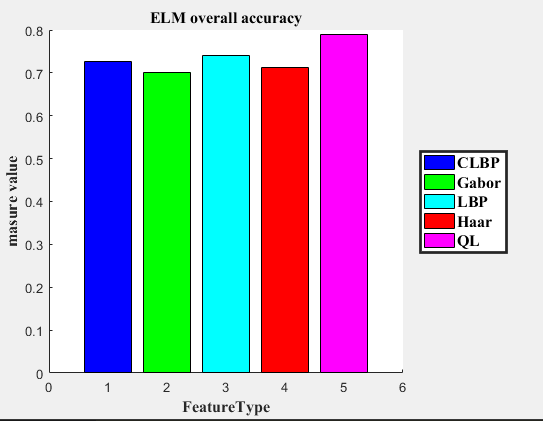
Confusion Matrix

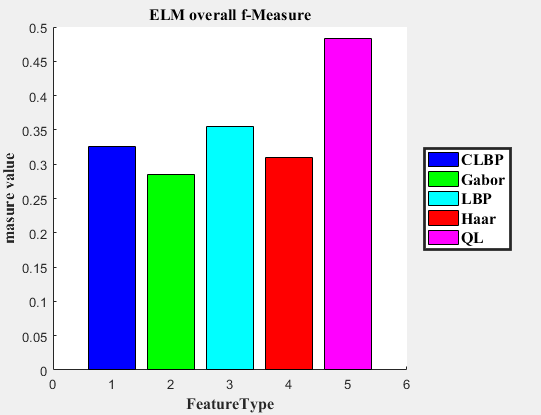


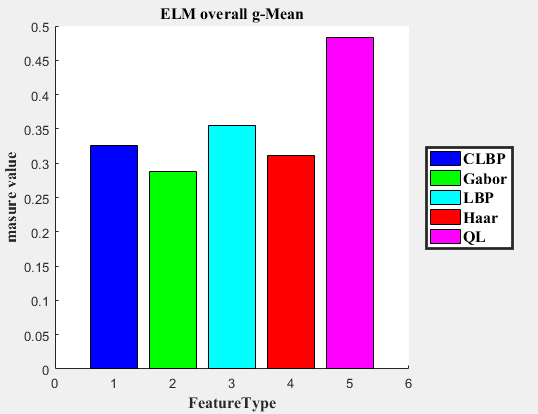
1. ELM and QL-ELM

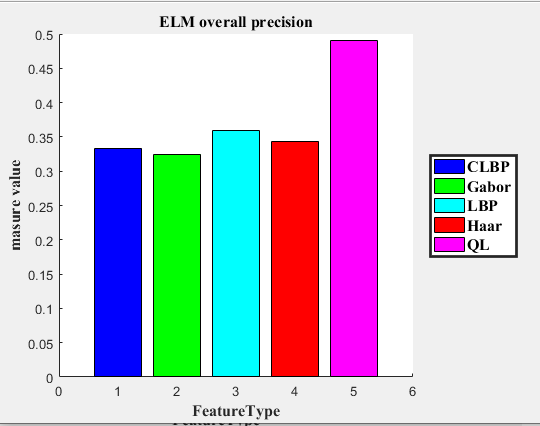


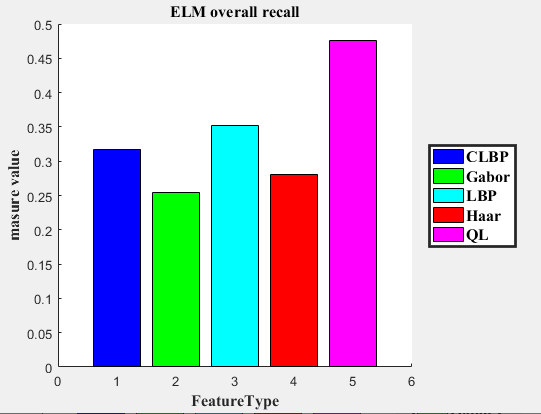


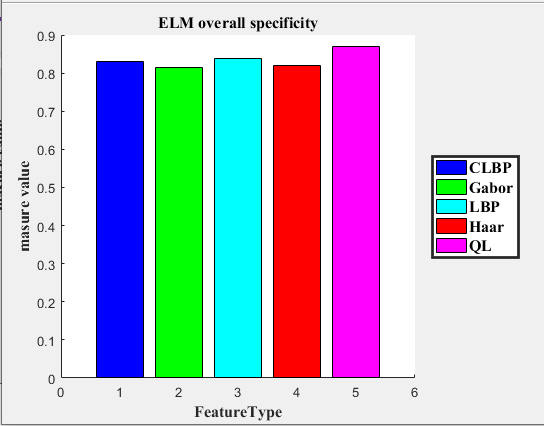


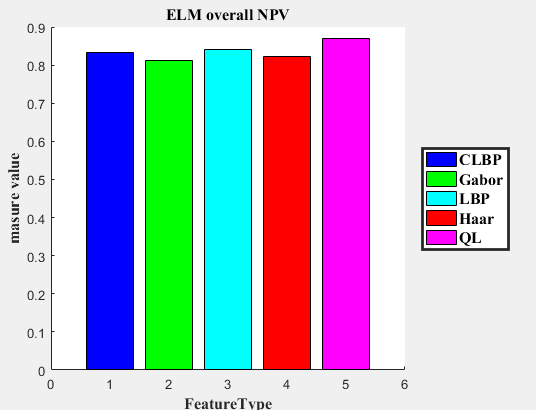




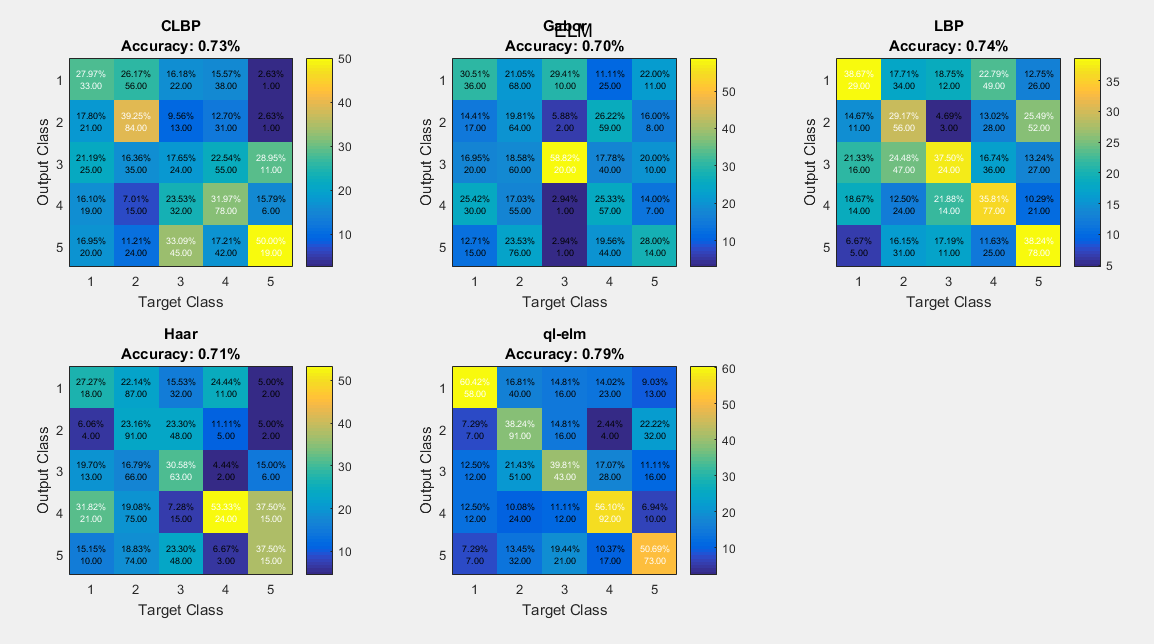




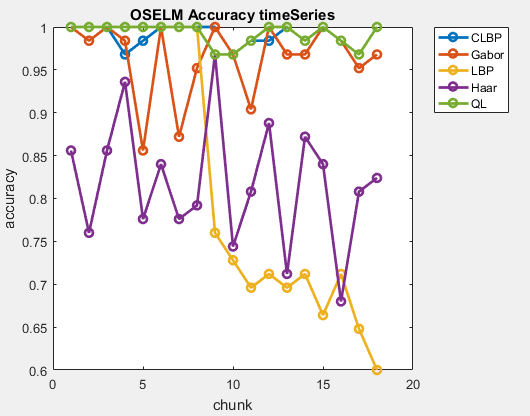


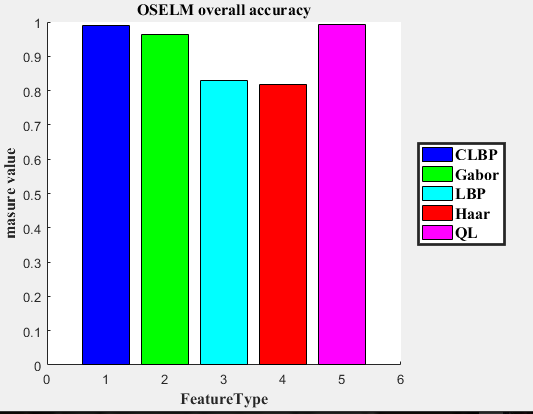


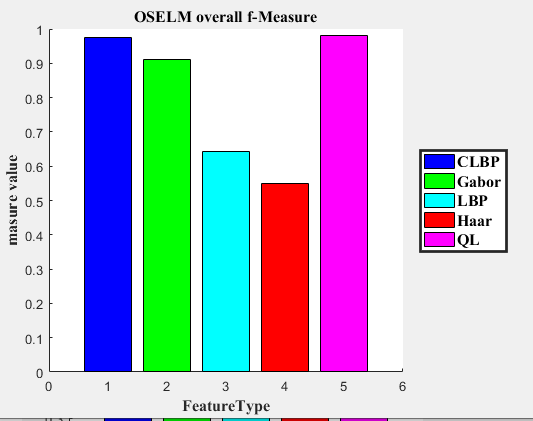
Confusion Matrix

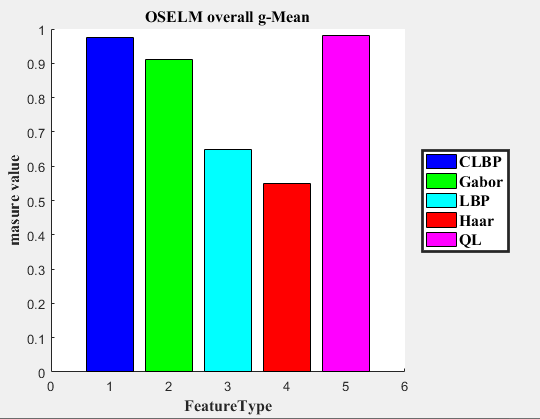


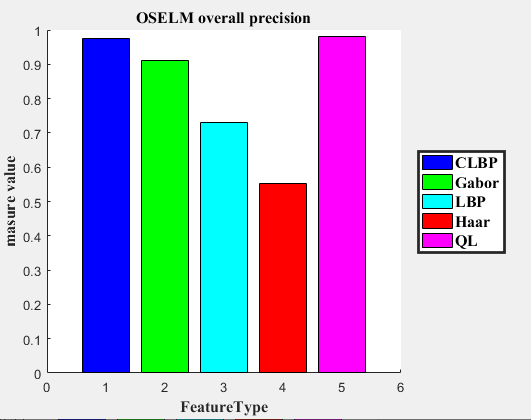
1. OSELM and QL-OSELM first version

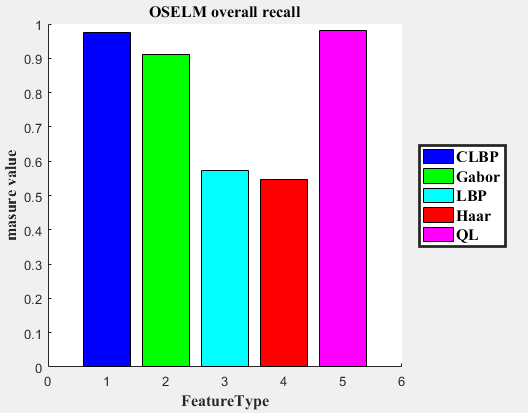


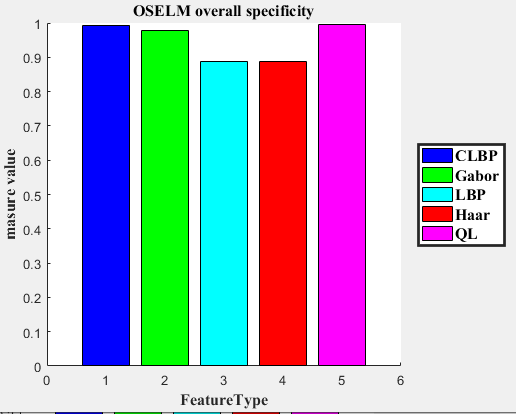


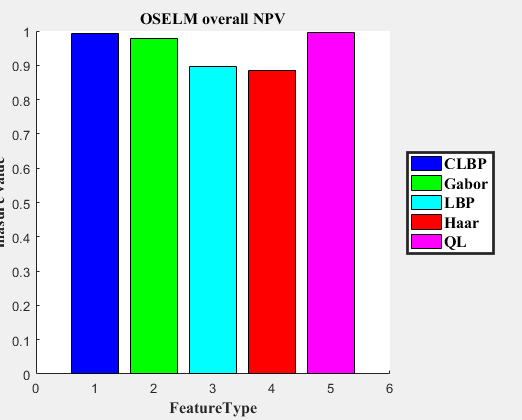




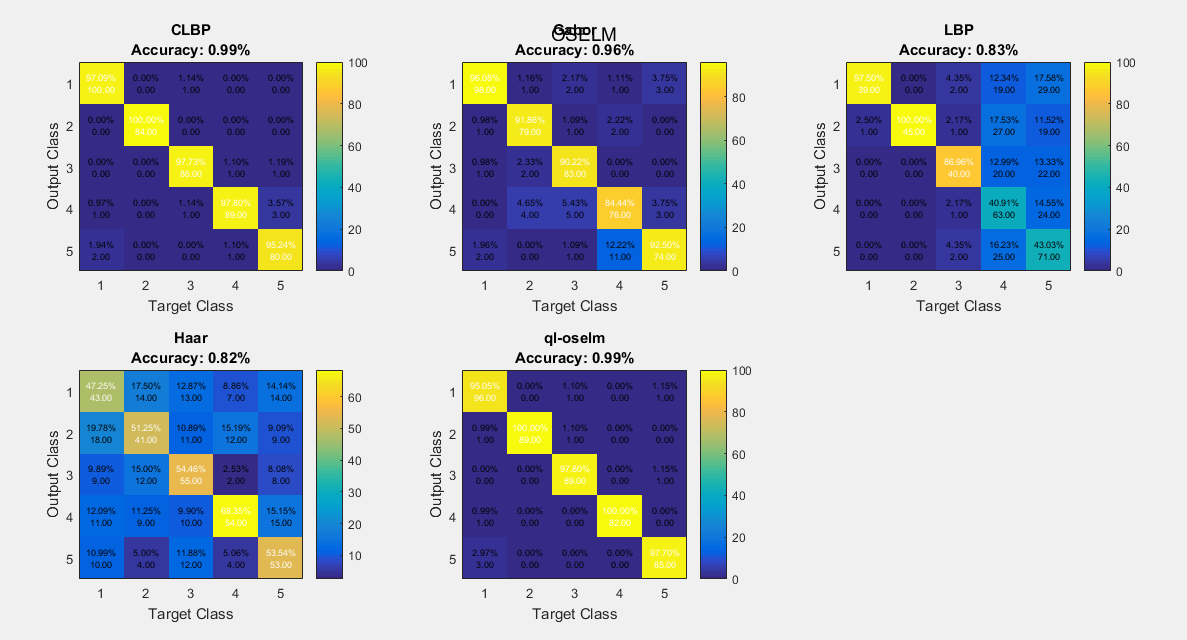




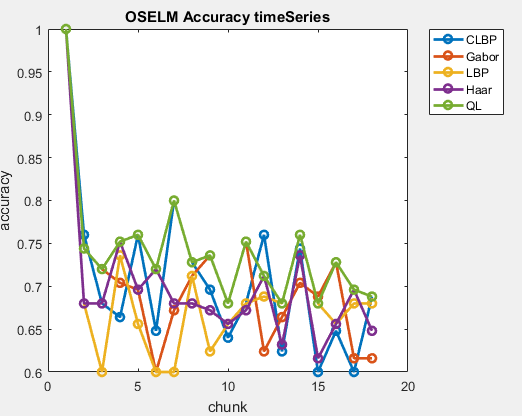


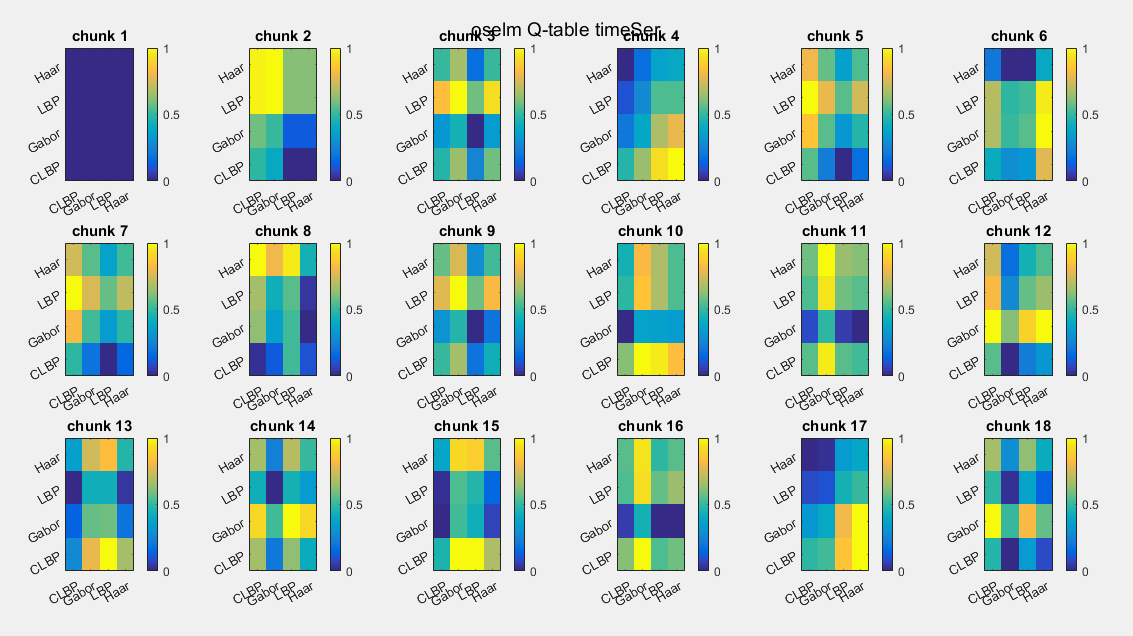


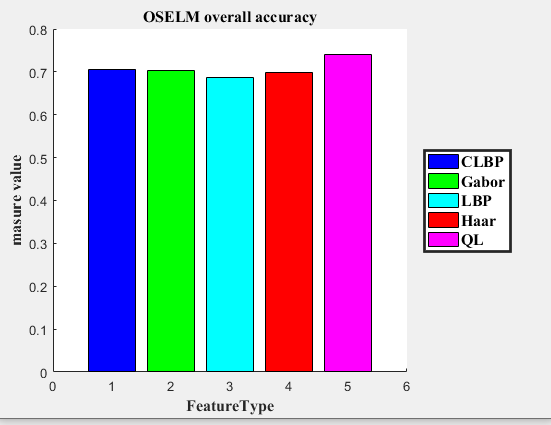
Confusion Matrix

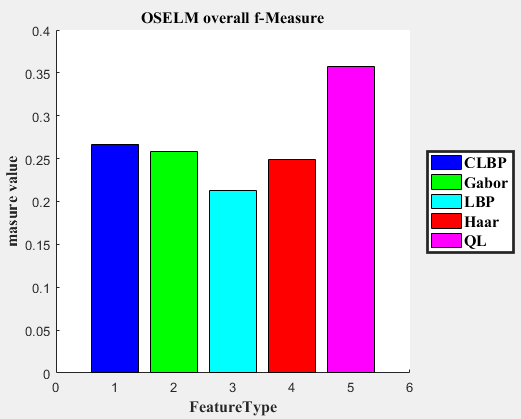


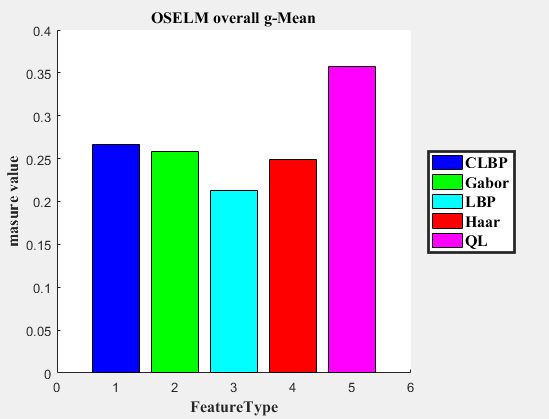
1. OSELM and QL-OSELM second version

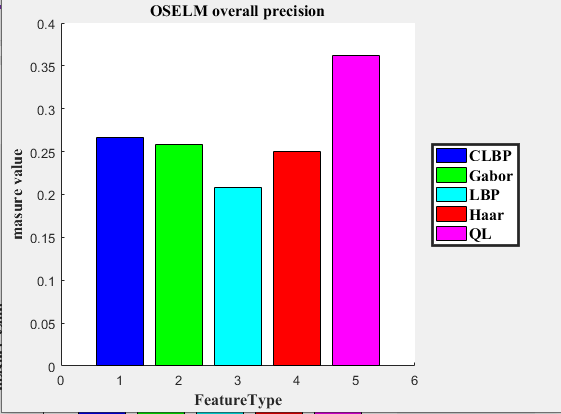


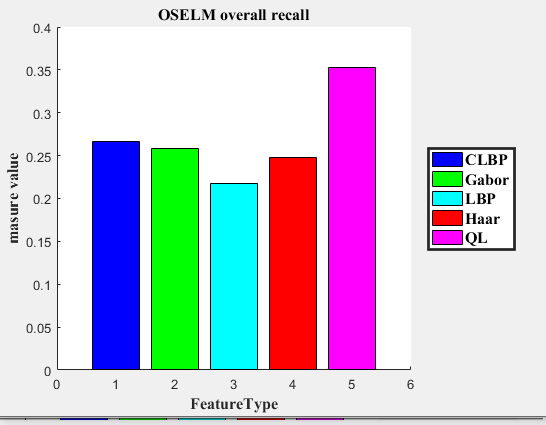


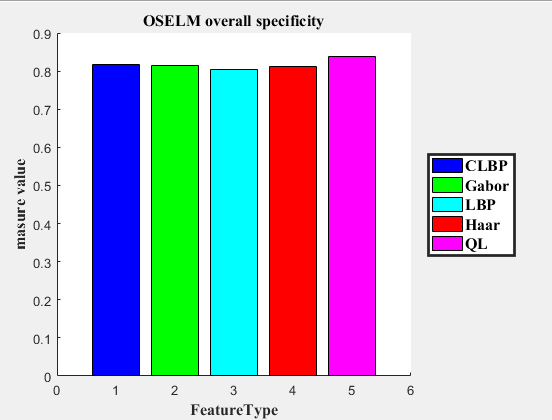


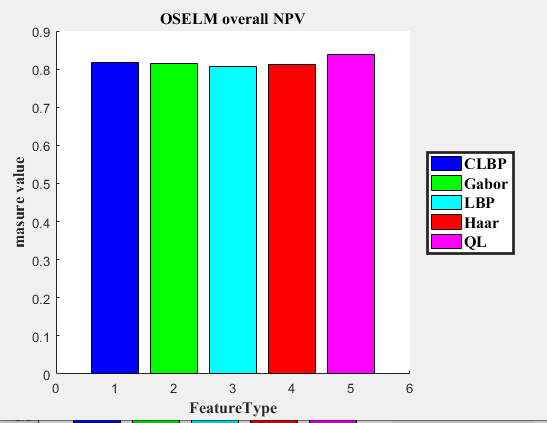




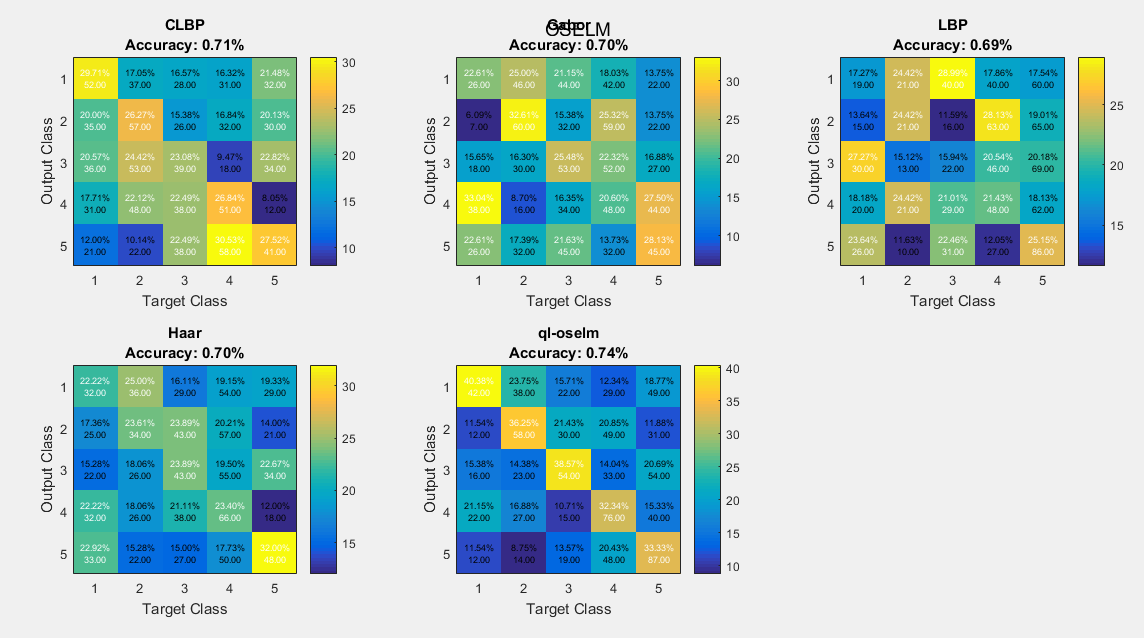




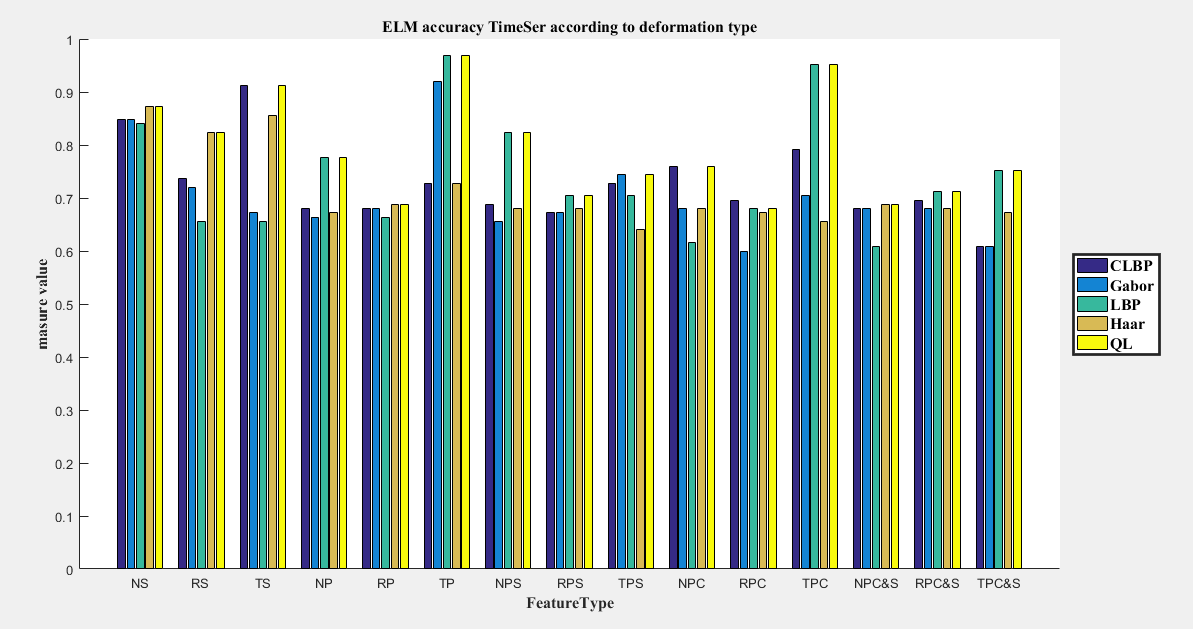


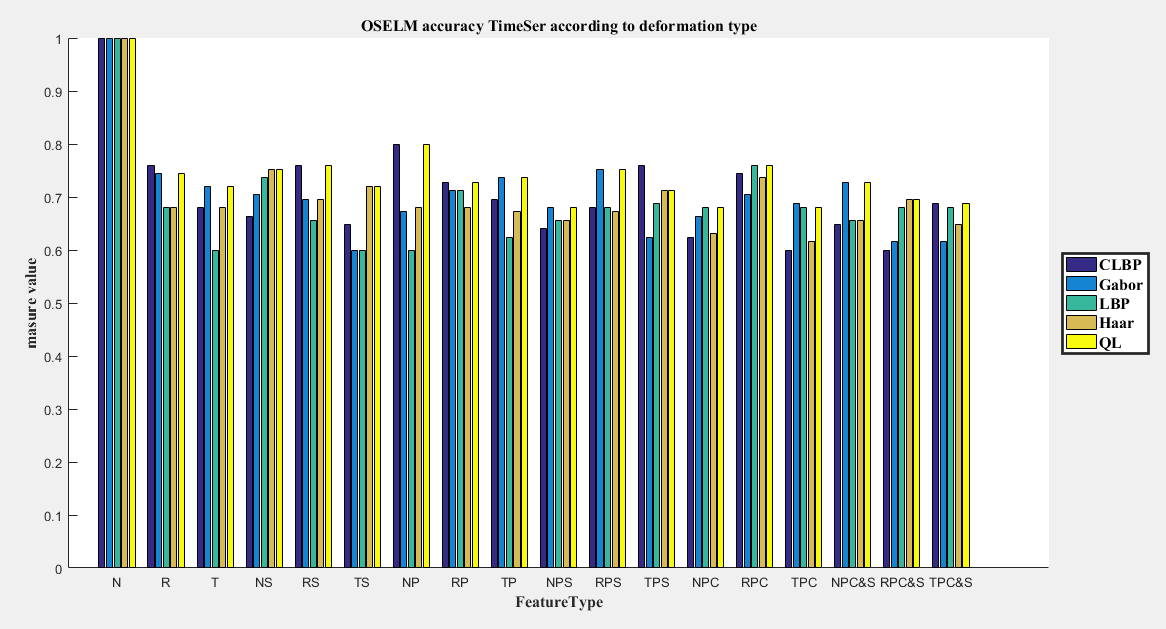


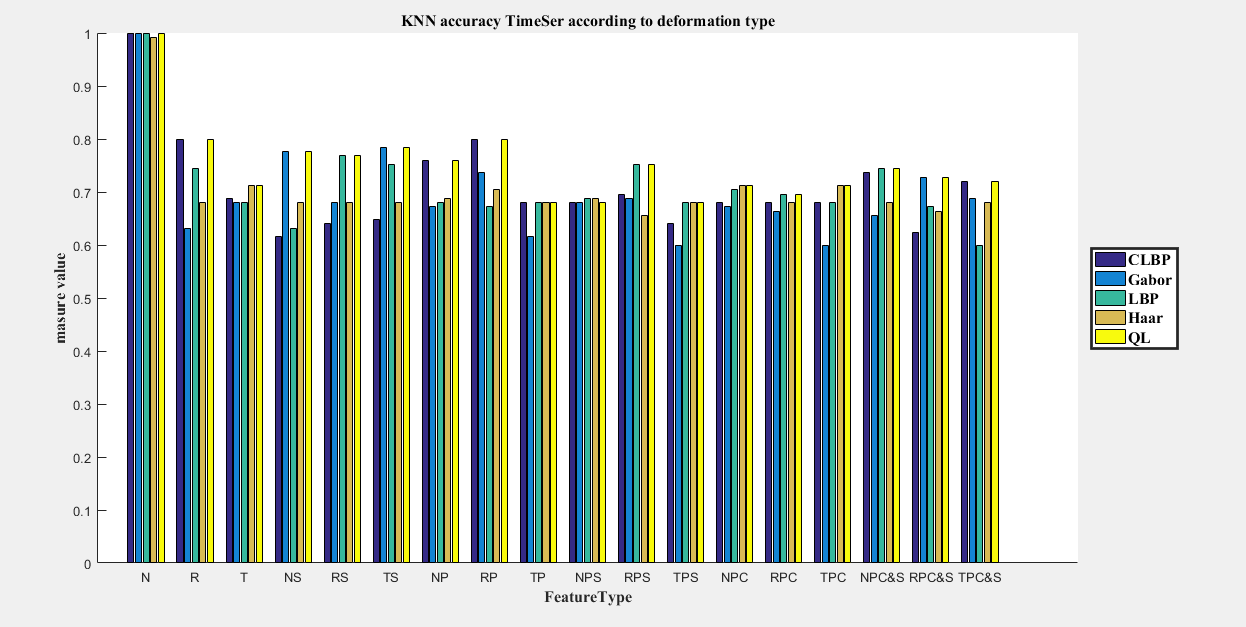
Confusion MAtrix



Accuracy according to deformation type:







If you want to generate the figures by your self

Go to project path/visualization/MainFile