EECS 410

Final Project Proposal

Epileptic Seizure Detection using EEG Signals

Group 13
Fei Guo (fxg104)
Xuhui Chen (xxc296)

Rubric

1. Data preprocessing: segment all the records into segments and preprocess the segments (de-noising, normalization or other process if needed). (5%)
2. Implement an appropriate active learning scheme for Epileptic Seizure detection. (10%)
3. Write code implementing the active learning scheme using TensorFlow or Torch. (10%)
4. Write code for the comparison scheme (randomly selecting samples for querying, the baseline scheme), including classifier training and testing. (5%)
5. Compare and evaluate the results utilizing confusion metrics. (5%)
6. Develop an Android app that implements the trained network to test any given segments of EEG signal (randomly select from database). (5%)

*Red bullet means finished.

Motivation

Authors	Features	Classification Method	Accuracy
Polat K *et al.*[4]	Spectral analysis of EEG signals and Welch (FFT) method	Artificial Immune Recognition System (AIRS)	100%
U. Rajendra Acharya *et al.* [5]	Recurrence quantification analysis	SVM	95.6%
Oran et al. [6]	DWT	ANN	96.67%
Ghosh-Dastidar *et al.* [7]	Mixed-band features space	Back Propagation Neural Network(BPNN)	96.7%
V. Srinivasan et al. [8]	Time and frequency domain features	Recurrent Neural Network	99.6%
K. Polat et al. [9]	Fast Fourier Transform	Decision Tree	98.7%

Dataset Description

Department of Epileptology, University of Bonn.[1]

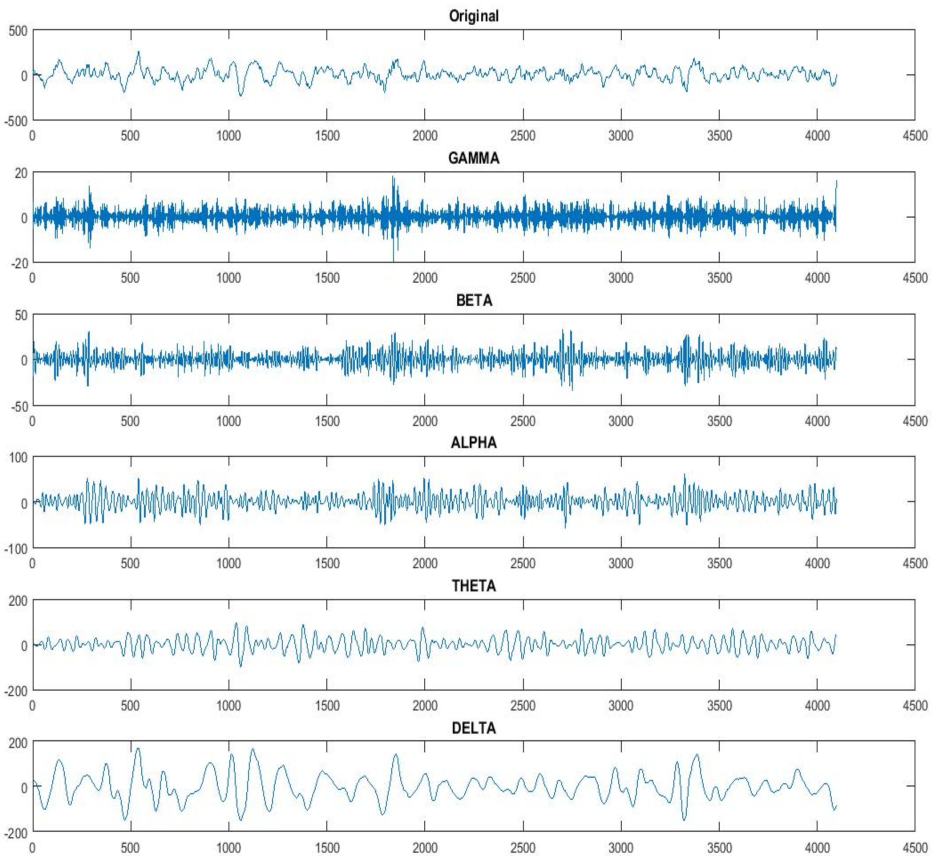
Dataset	Testers	Settings	State
A	5 healthy volunteers	Standardized international 10-20 electrode placement scheme	Eye open
B	5 healthy volunteers	Standardized international 10-20 electrode placement scheme.	Eye closed
C	5 epilepsy patients	Depth Intracranial electrodes implanted (non epileptogenic zone)	Seizure free interval
D	5 epilepsy patients	Depth Intracranial electrodes implanted (epileptogenic zone)	Seizure free interval
E	5 epilepsy patients	Depth Intracranial electrodes implanted	Only seizure activity

Each dataset has 100 records, and each record contains 23.6-second single channel EEG signals.

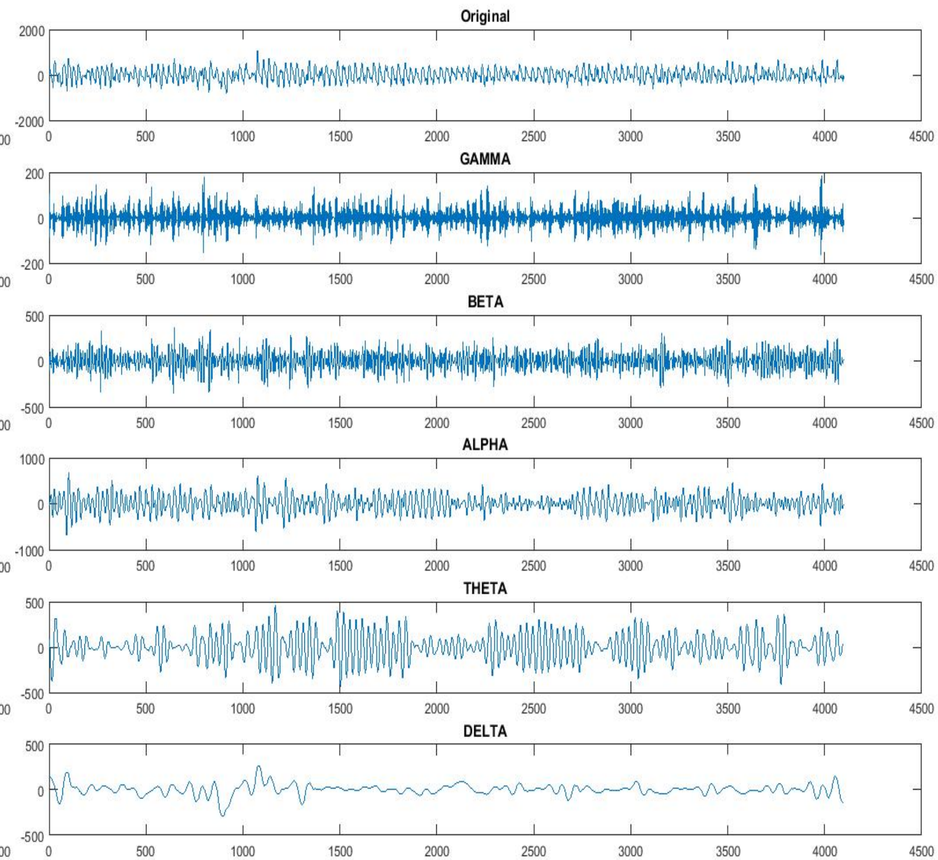
Data Preprocessing

1

C: Seizure Free Interval

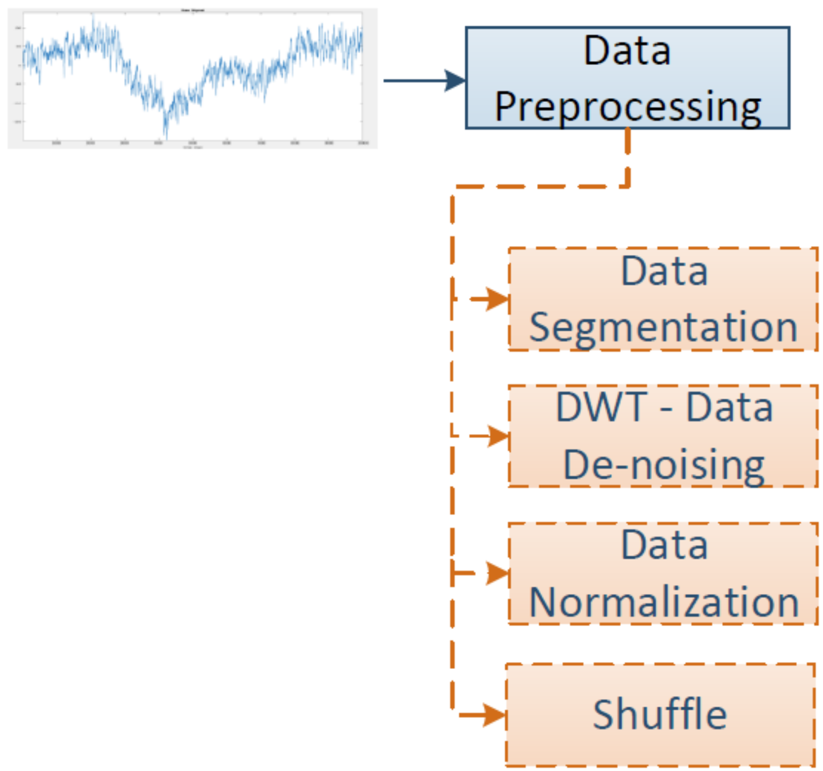


E: Seizure Activity Only



Data Preprocessing

1



Method 2

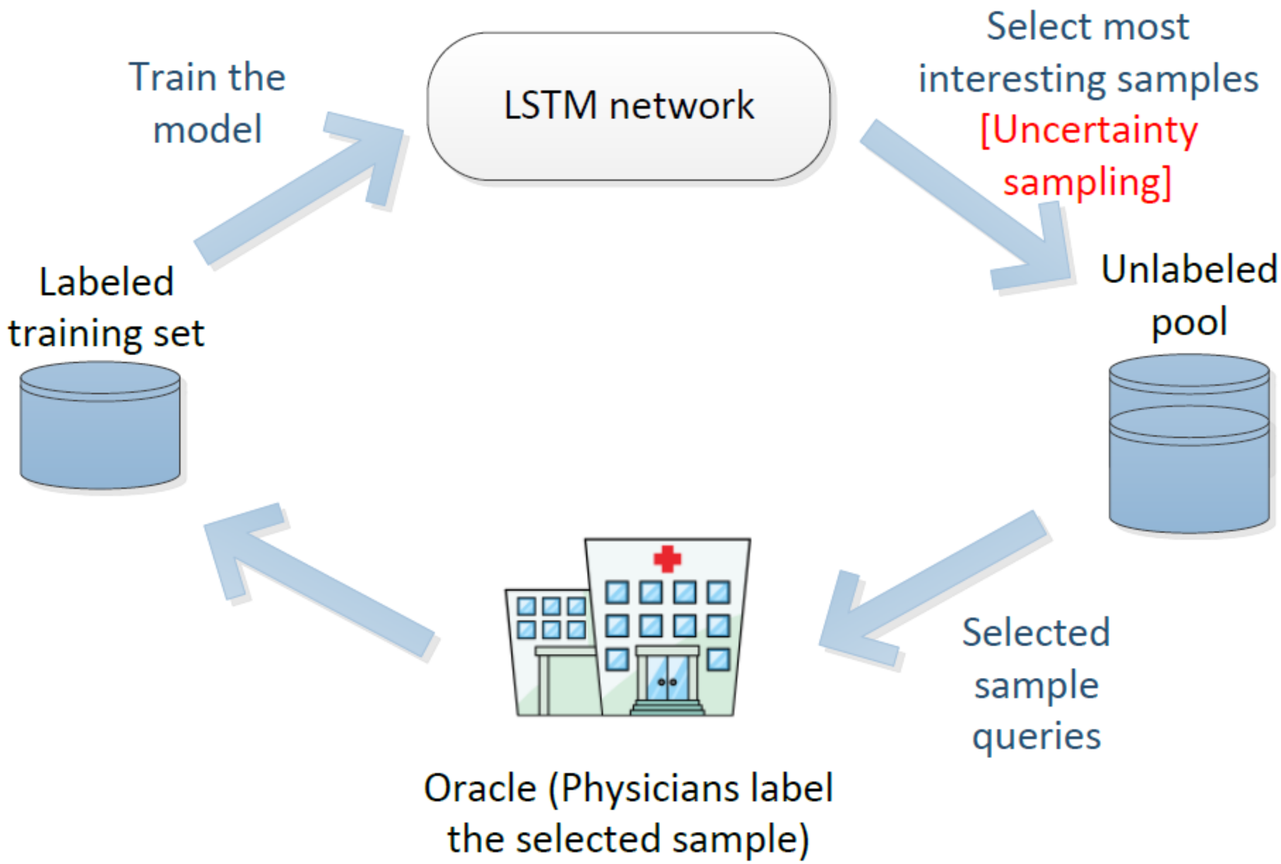


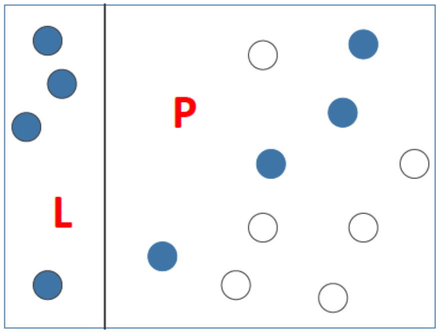
Figure 1. Active learning framework

Method

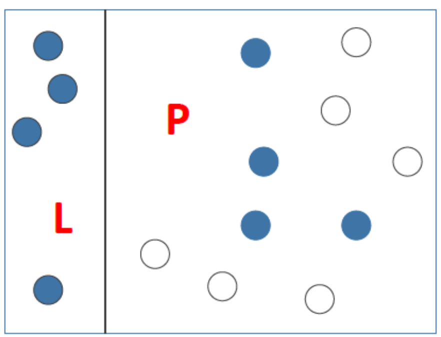
2

The training process: we divide the training dataset into two parts, L and P, and treat them as “labeled training set” and “Unlabeled pool”, respectively.

- Initialization training set L (minority)
- Sample pool P (majority)



Selected active learning scheme: Uncertainty sampling



Comparison scheme: Randomly sampling

Algorithm 1 Active learning framework

Given:

L: Set of labeled examples

P: Set of unlabeled pool

B: number of examples to be selected in each iteration

U_M: utility function

Algorithm:

loop until stopping criterion is met

1. train the LSTM model M using dataset L
2. for all $p_i \in P : u_{p_i} \leftarrow U_M(p_i)$
3. Active learning scheme: select B examples $p_i \in P$ with highest utility u_{p_i}.
(Comparison scheme: randomly select B samples u_{p_i}.)
4. retrieve the selected samples with their labels (p_i, y_{p_i})
5. move the selected examples from P to L

Return: Trained LSTM model M

Code for Uncertainty Sampling

3

```python
    for _ in range(int(quota/query_sample_num)): # the total number of iterations
        unlabeled_entry_ids, x_pool = zip(*trn_ds11.get_unlabeled_entries())
        #print(unlabeled_entry_ids)
        X_pool = np.array(x_pool)
        #print(len(X_pool))
        dvalue = model1.predict(X_pool, batch_size=1)
        score_uncer = np.zeros((len(dvalue),2))
        for j in range(len(dvalue)):
            score_uncer[j][0] = unlabeled_entry_ids[j]
            score_uncer[j][1] = abs(dvalue[j]-0.5) # calculate the uncertainty for each unlabeled sample

        score_uncer_arg = np.argsort(score_uncer[:,1])# sort the score on second column
        score_uncer = score_uncer[score_uncer_arg]

        for i in range(query_sample_num): # "query_sample_num" number of samples in each iteration
            ask_id = int(score_uncer[i][0])
            X, _ = zip(*trn_ds11.data)
            lb = lbr.label(X[ask_id]) # Oracl: returns the label of the queried sample
            trn_ds11.update(ask_id, lb) #updates the unlabeled sample with queried label.

        trn_ds_feature, trn_ds_label = zip(*trn_ds11.get_labeled_entries())
        trn_ds_feature_np = np.array(trn_ds_feature)
        trn_ds_label_np = np.array(trn_ds_label)

        print('Train a new LSTM1...')
        print('number of training samples: ', len(trn_ds_label_np))

        model1 = Sequential()
        model1.add(LSTM(256, input_shape=(1, 347)))
        model1.add(Dense(1, activation='sigmoid'))
        model1.compile(loss='binary_crossentropy',
                    optimizer='adam',
                    metrics=['accuracy'])
        model1.fit(trn_ds_feature_np, trn_ds_label_np, batch_size=batch_size, epochs=10) # train a new LSTM
```

Code for Uncertainty Sampling

3

```python
        # Test the current LSTM network
        predict_prob2 =  model1.predict(X_test, batch_size=1)# the predicted probability of testing data
        true_positive_rate, true_negative_rate, precision, F1, acc = metric_test(predict_prob2,y_test)

        TPR1 = np.append(TPR1, true_positive_rate)
        TNR1 = np.append(TNR1, true_negative_rate)
        Preci1 = np.append(Preci1, precision)
        F1_Total1 = np.append(F1_Total1, F1)
        ACC_Total1 = np.append(ACC_Total1, acc)
        print('TPR:', TPR1)
        print('TNR:', TNR1)
        print('Precision:', Preci1)
        print('F1:', F1_Total1)
        print('Acc:', ACC_Total1)
```

Code for Random Sampling

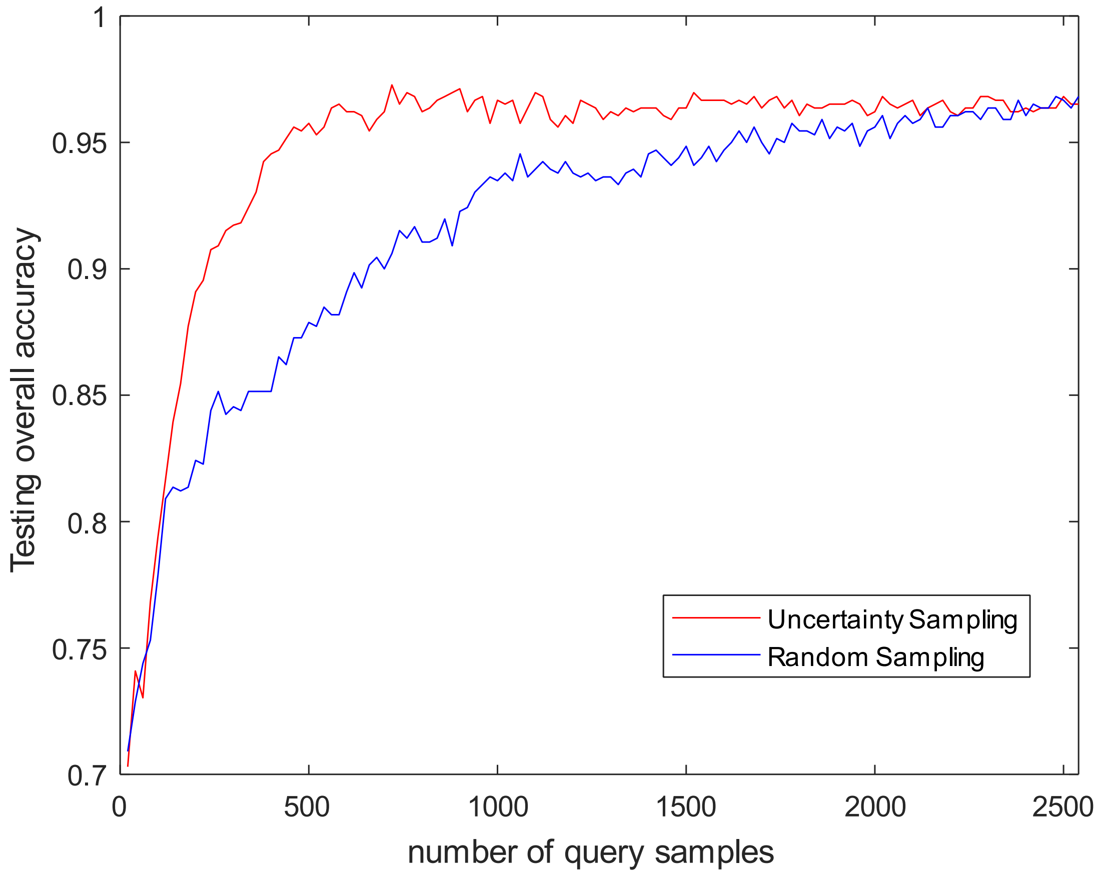
4

```python
for _ in range(int(quota/query_sample_num)): # the total number of iterations
    for i in range(query_sample_num):
        ask_id = qs.make_query() #returns the index of the sample that the active learning algorithm wants to query
        X, _ = zip(*trn_ds2.data)
        lb = lbr.label(X[ask_id]) #returns the label of the given sample answered by oracle.
        trn_ds2.update(ask_id, lb) #updates the unlabeled sample with queried label.

    trn_ds_feature, trn_ds_label = zip(*trn_ds2.get_labeled_entries())
    trn_ds_feature_np = np.array(trn_ds_feature)
    trn_ds_label_np = np.array(trn_ds_label)

    #model.train(trn_ds)
    print('Train a new LSTM1...')

    model = Sequential()
    model.add(LSTM(256, input_shape=(1,347)))
    model.add(Dense(1, activation = 'sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    print('Train...')
    model.fit(trn_ds_feature_np, trn_ds_label_np, batch_size=batch_size, epochs=10)
```

```python
    # Test the current LSTM network
    predict_prob2 =  model.predict(X_test, batch_size=1) # the predicted probability of testing data
    true_positive_rate, true_negative_rate, precision, F1, acc = metric_test(predict_prob2,y_test)

    TPR2 = np.append(TPR2, true_positive_rate)
    TNR2 = np.append(TNR2, true_negative_rate)
    Preci2 = np.append(Preci2, precision)
    F1_Total2 = np.append(F1_Total2, F1)
    ACC_Total2 = np.append(ACC_Total2, acc)
    print('TPR:', TPR2)
    print('TNR:', TNR2)
    print('Precision:', Preci2)
    print('F1:', F1_Total2)
    print('Acc:', ACC_Total2)
```

Evaluation – Active learning

5



Evaluation – LSTM performance

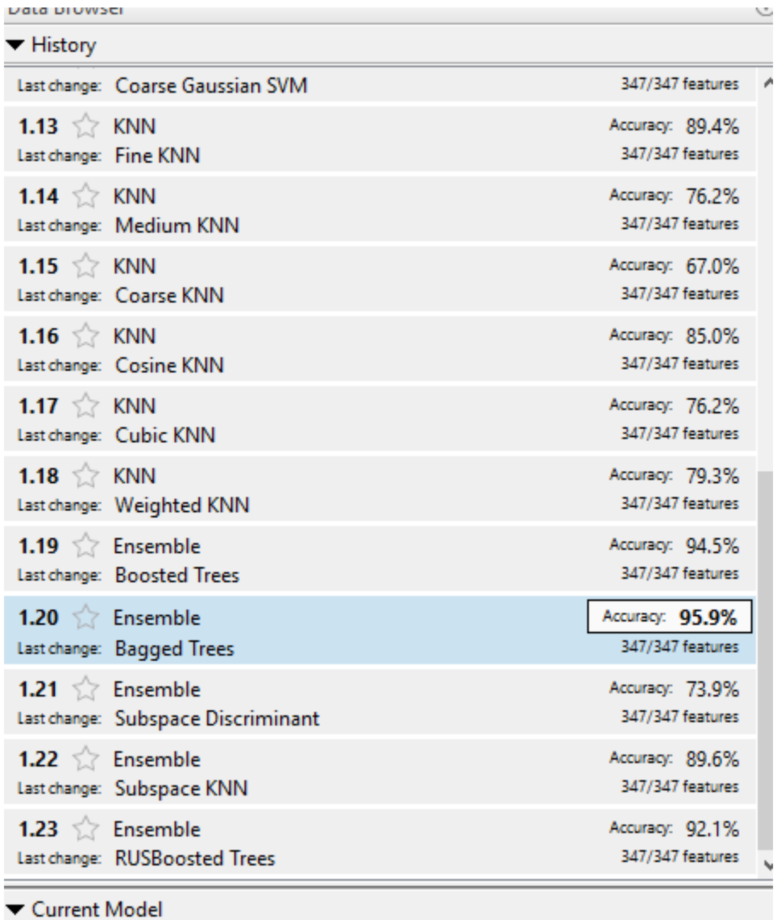
5

```python
# 5-fold cross validation
kf = KFold(n_splits = 5, shuffle=False)
acc_total = 0

for train_index,test_index in kf.split(X):
    x_train, x_test = X[train_index], X[test_index]
    y_train, y_test = Y[train_index], Y[test_index]
    print('train samples: ', len(x_train))
    print( 'test samples: ', len(x_test))

```

```python
    print('Build model...')
    model = Sequential()
    model.add(LSTM(256, input_shape=(1, 347)))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    model.fit(X_train, y_train,
              batch_size=batch_size,
              epochs=10)

    predict_probability = model.predict(X_test)
    predict_result = copy.deepcopy(predict_probability)
    predict_result[predict_probability >= 0.5] = 1
    predict_result[predict_probability < 0.5] = 0

    print (confusion_matrix(y_test, predict_result))
    score, acc = model.evaluate(X_test, y_test,
                                batch_size=batch_size)
    acc_total = acc_total + acc
    print('Test accuracy:', acc)

print('Averaged testing accuracy: ', acc_total/5.0)
```

Evaluation

MATLAB classifiers



Ensemble Bagged Trees: 95.9%

		True class	
		1	0
Predicted class	1	1029	63
	0	71	2137

Precision = 94.23%
Recall = 93.55%

LSTM: 96.3%

		True class	
		1	0
Predicted class	1	1001	23
	0	99	2177

Precision = 97.75%
Recall = 91%

Android app

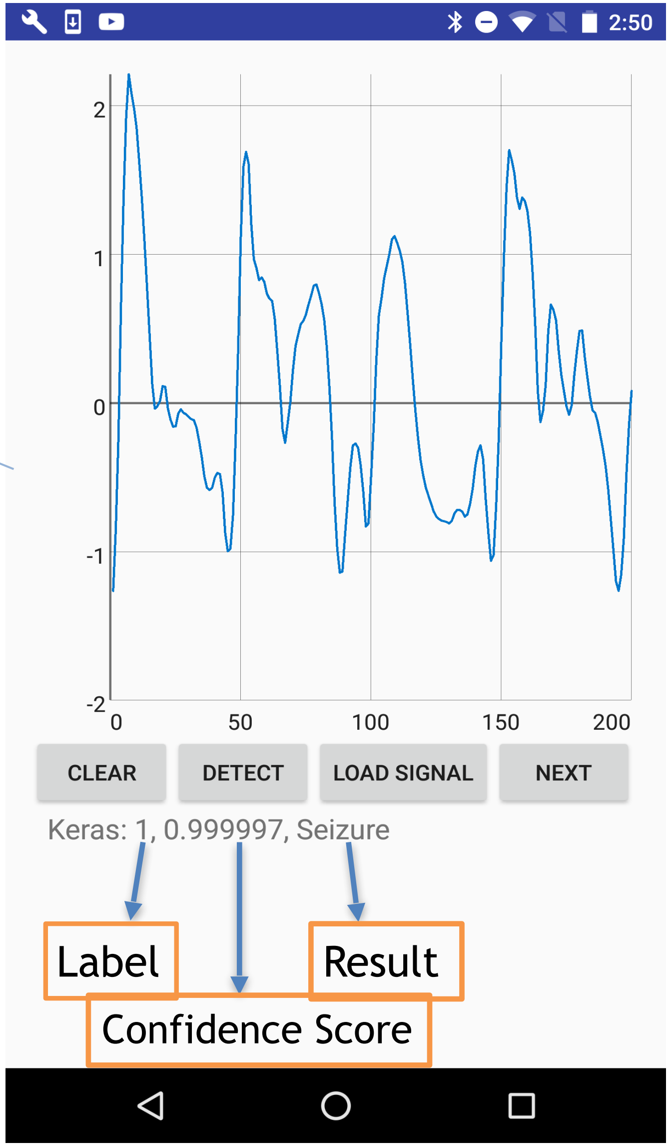
6

Implement TensorFlow on Android.

GraphView – Plot/Display Signals

Buttons - Load/Detect/Clear/Next

TextView – Classification Results



Extra

Except the tasks described in the rubrics, we are also trying to implement the training process on the smart phone, but we are still debugging the code. Since we could not find an appropriate library to directly be used for LSTM training on Android, we write the training code by ourselves. We haven't got any results, so we didn't put the performance in the slides.

reference

[1] Andrzejak RG, Lehnertz K, Rieke C, Mormann F, David P, Elger CE (2001) Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, Phys. Rev. E, 64, 061907

[4] Polat, K. and Güneş, S., 2008. Artificial immune recognition system with fuzzy resource allocation mechanism classifier, principal component analysis and FFT method based new hybrid automated identification system for classification of EEG signals. *Expert Systems with Applications*, *34*(3), pp.2039-2048.

[5] U.R. Acharya, S.V. Sree, S. Chattopadhyay, W. Yu, A.P.C. Alvin, Application ofrecurrence quantification analysis for the automated identification ofepileptic EEG signals, Int. J. Neural Syst. 21 (3) (2011) 199–211.

[6] U. Orhan, M. Hekim, M. Ozer, EEG signals classification using the K-meansclustering and a multilayer perceptron neural network model, Expert Syst.Appl. 38 (10) (2011) 13475–13481.

[7] S. Ghosh-Dastidar, H. Adeli, N. Dadmehr, Mixed-band wavelet-chaos-neuralnetwork methodology for epilepsy and epileptic seizure detection, IEEE Trans.Biomed. Eng. 54 (9) (2007) 1545–1551.

[8] V. Srinivasan, C. Eswaran, N. Sriraam, Artificial neural network based epileptic detection using time-domain and frequency domain features, Journal of Medical Systems 29 (6) (2005) 647–660.

[9] K. Polat, S. Guenes, Classification of epileptiform EEG using a hybrid systems based on decision tree classifier and fast Fourier transform, Applied Mathematics and Computation 32 (2) (2007) 625–631.

THANK YOU!

Q & A