High Frequency Resistance Prediction Model User Manual

Tong Lin, Levent Burak Kara

January 30, 2020

Contents

1 Data Format Organization

1. Create a folder in *data* folder and name it *excel_data*.

2. Separate the excel data so that each excel file only contains a single set of experiment data. Save them into the *excel_data* folder.

3. Go to the *utils* folder. Open *organize_HFR_data_2.m* and change the main path to be the path of the *utils* folder.

4. After running the script, you will see *DataSummary.mat* in the *data* folder.

5. Create a folder called *DataSummary_Plot* in the *data* folder.

6. Run *LH_Plot_check_data.m* in the *utils* folder.

7. You will see images of "HFR and current v.s sequences" produced in the *data* folder, which is shown in Figure 1

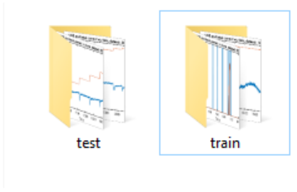


Figure 1: Files in *DataSummary_Plot folder*

8. For each experiment data, manually check if the HFR value or current value is good or not. For example, if the HFR or current remain constant, the experiment time is too short, or the HFR readings are completely noisy, the images should be deleted.

9. Some images have partial noise. For these images, revisit the excel file to delete the noisy part and re-run step 1 through step 8.

10. Create a *train* folder and a *test* folder under the *DataSummary_Plot*

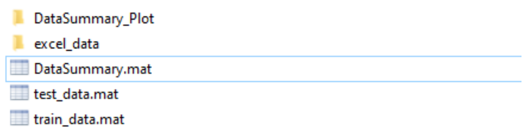
folder.

11. Manually split the images into train and test data. Make sure that the train data and the test data are not similar. A train/test ratio of 4/1 is recommended.

12. Run *train_test_split.m* in the *utils* folder. It will generate a *train_data.mat* file and a *test_data.mat* file in the *data* folder.

13. On occasion, MATLAB may leave the last row empty when reading CSV files. Run *remove_nan.m* in the *utils* folder to remove any Nan data.

14. At this point, data organization is complete. Your data structure and files should be the same as those shown in Figure 2



DataSummary_Plot
excel_data
DataSummary.mat
test_data.mat
train_data.mat

Figure 2: Files in *data* folder

2 Data Preprocessing

Three data preprocessing methods are used:

1. HFR difference limits
   code: *filterHFR.m* in the *function* folder.
   We suppress the filter spikes so that for two consecutive HFR values whose difference is greater than $|5|$, their differences are confined to $|0.1|$. An example result is shown in Figure 3.

2. Signal smoothing
   code: line 48 to line 63 in *create_LSTM_data.m* in the *functions* folder.

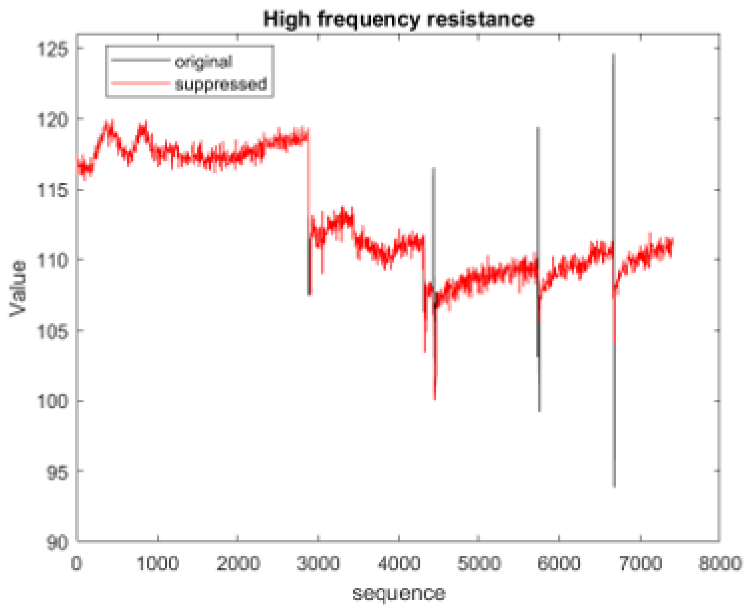


Figure 3: An example of limiting HFR value

For signals that have outlet and inlet values, the inlet value and the difference between the outlet and inlet values are used. The signals are filtered using a Savitzky–Golay filter, except for voltage and current, which behave like step functions. Our recommended value for the filter setting is $order = 1, window\_size = 101$. An example result is shown in Figure 4.

3. Signal resampling
   code: line 64 to line 65 in *create_LSTM_data.m* in the *functions* folder.
   The signal is resampled every $skip + 1$ point since the original sensor frequency is too high. Our recommended value for $skip$ is 19. An example result is shown in Figure 5.

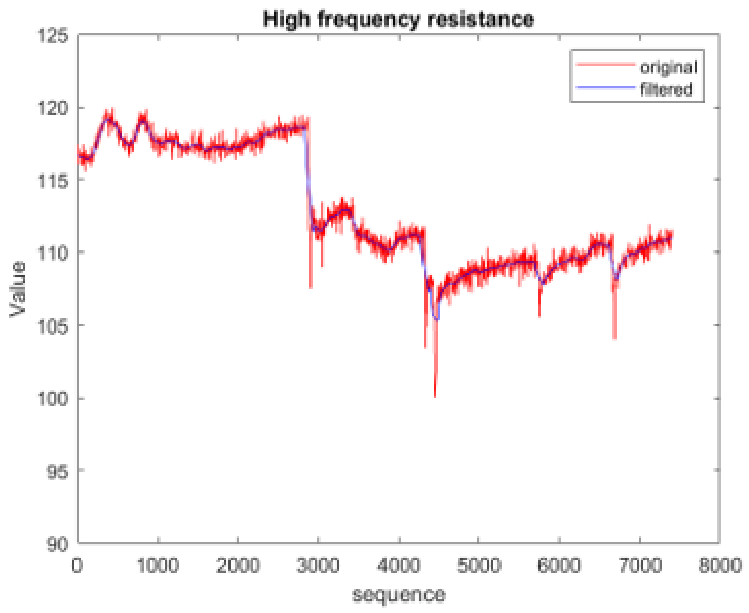


Figure 4: An example of using Sgolay_filter

3 Model Training

1. Open *train.m* and set the *filePath* to save the model.

2. Run the script to being training. The script will automatically get data, pre-process data, train the model and save the model at every epoch. An example of the training process is shown in Figure 6.

3. The explanation of the parameters in the code and the recommended values are shown in Figure 7.

4 Model Testing and Selection

During the training process, many models are produced. We need to select the best model based on the test data sets. Thus, we need to test every model on the test data sets in order to select the best one. In addition, it is also a good practice to check the model performances on the training data sets to make sure that our model overfits the data. Thus, we evaluate each

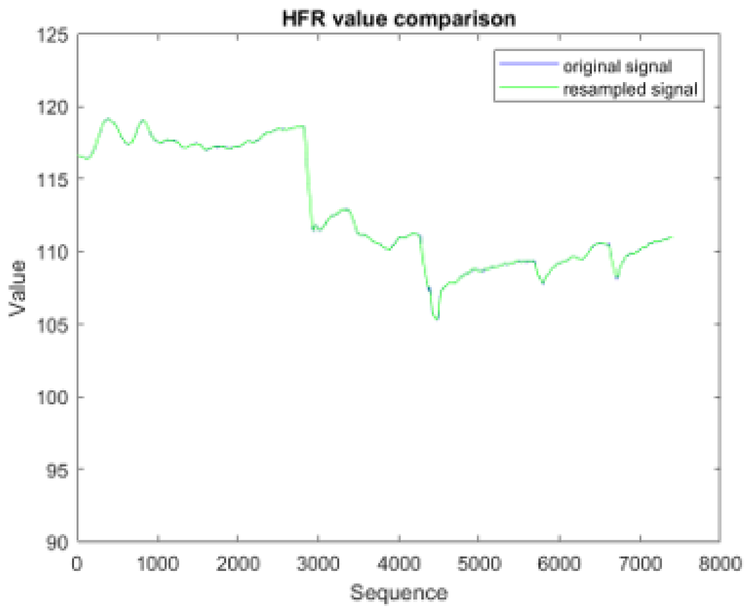


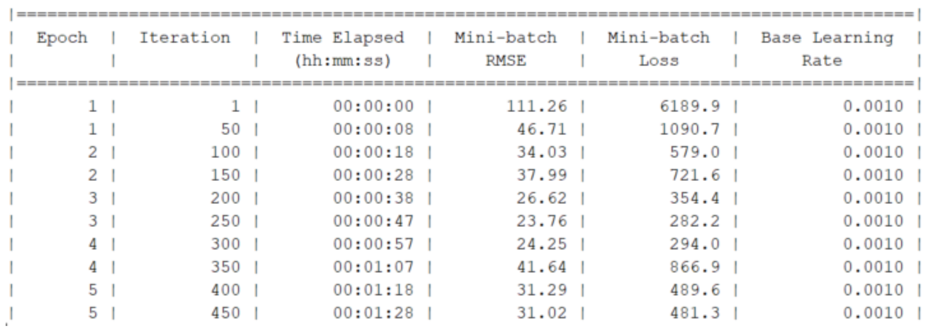
Figure 5: An example of re-sampling

model in the folder on the test datasets and the train datasets. The root mean square value (RMSE) for the train data sets and the test data sets are then saved in the folder.

1. Open *evaluation.m*. Set the *folder_path* to the folder where you save your model.

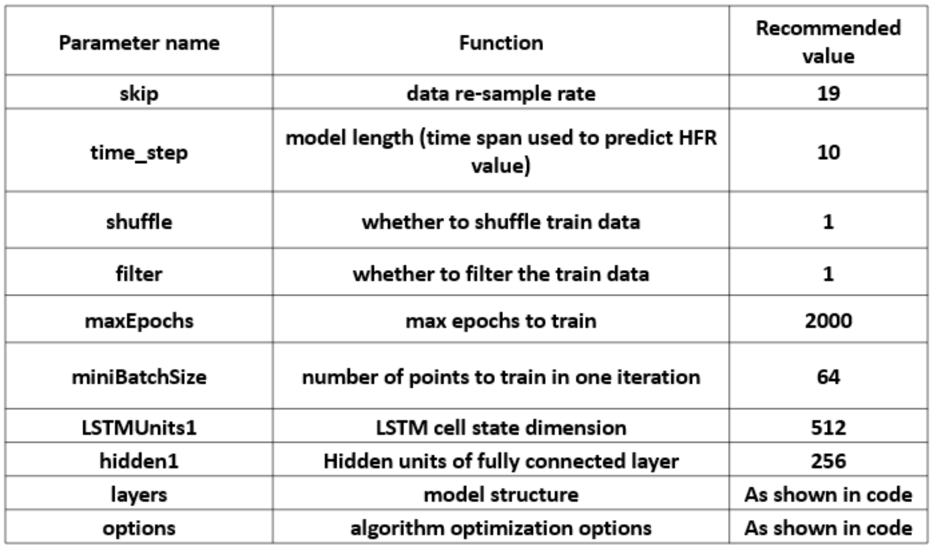
2. Run *evaluation.m* to obtain the RMSE value of each model for the train data sets and the test data sets.

3. Make sure the training process overfits the data. Select the model with the lowest RMSE value in the test data set. An example result is shown in Figure 8.



```
|==========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     Loss     |      Rate       |
|==========================================================================================|
|       1 |           1 |       00:00:00 |       111.26 |       6189.9 |          0.0010 |
|       1 |          50 |       00:00:08 |        46.71 |       1090.7 |          0.0010 |
|       2 |         100 |       00:00:18 |        34.03 |        579.0 |          0.0010 |
|       2 |         150 |       00:00:28 |        37.99 |        721.6 |          0.0010 |
|       3 |         200 |       00:00:38 |        26.62 |        354.4 |          0.0010 |
|       3 |         250 |       00:00:47 |        23.76 |        282.2 |          0.0010 |
|       4 |         300 |       00:00:57 |        24.25 |        294.0 |          0.0010 |
|       4 |         350 |       00:01:07 |        41.64 |        866.9 |          0.0010 |
|       5 |         400 |       00:01:18 |        31.29 |        489.6 |          0.0010 |
|       5 |         450 |       00:01:28 |        31.02 |        481.3 |          0.0010 |
```

Figure 6: An example of re-sampling



Parameter name	Function	Recommended value
skip	data re-sample rate	19
time_step	model length (time span used to predict HFR value)	10
shuffle	whether to shuffle train data	1
filter	whether to filter the train data	1
maxEpochs	max epochs to train	2000
miniBatchSize	number of points to train in one iteration	64
LSTMUnits1	LSTM cell state dimension	512
hidden1	Hidden units of fully connected layer	256
layers	model structure	As shown in code
options	algorithm optimization options	As shown in code

Figure 7: An example of re-sampling

5 Model Implementation

All the implementation codes are in the *simulink* folder. To implement MATLAB code in Simulink, the prediction model needs to be rewritten into basic math operations. The following explains how to use the model in Simulink:

1. In *simulink/utilities*, open *save_weights.m*.

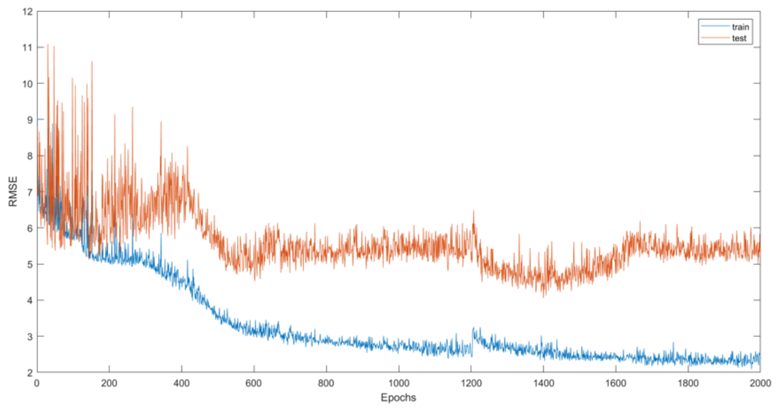


Figure 8: Training curve and test curve example

2. Set the model path to be the same path as your selected model file. Change T to *time_step* in the training setup.

3. Run the script and you will obtain the model specifications for simulink in the *model_specs* folder.

(Optional) The below step is for the simulated input signal on a computer

To test on a computer (not a board), you can also obtain simulated input signal for simulink using *HFR_data_4_sim.m*.

4. Open *simulnk/utilities/HFR_data_4_sim.m*.

   - Set the *data_set_path* to the path of your test data set is.

   - Set the *set_num*. For example, there are 8 data sets in *test_data.mat*. You can choose one set to test on the *simulink*.

   - Make sure the structure field name is correct (line 15 in the code).

   - Run the script and you will obtain the simulated experiment signals for simulink in the *simulink/data* folder.

5. Open the signal data. It should be 13 × # of points. In *simulink*, set the simulation stop time to (# of points -1).

---

4. Open *simulink/test_model_ver1.slx*.
   * Simulink is static code, thus, many setting needs to be manually input into the model. Below is the detailed explanation. Note that all operations below are in the simulink.

5. Go to *Prediction_model/LSTMmodel/*.

6. Open *For Iterator*. Set *Iteration limit* equal to *time step* in the training setup.

7. Open *compute_gates*. Set *n_LSTM_units* equal to *LSTMUnits*1 in the training setup.

8. Open *c_state* (the one that is not connected). Go to "Signal Attributes" and set "Initial value" to zeros(*LSTMUnits*1, 1)

9. Repeat step 8 for *h_state*.

10. Open *Reset c_state and h_state*. Reset the two constants to zeros (*LSTMUnits*1, 1).

11. Open *organize_data*. Set T = "the value of time step in the training setup".
12. Open *if*. Set u1 = "*time step* in the training setup".

13. Go to *Prediction_model/MLPmodel/*

14. Open *MLP_forward*

    - For *relu*1, set the shape to (1,*hidden*1)

    - For *output_*1, set the shape to (*hidden*1,1)

- For $weight2$, reshape it to (1,$hidden1$)

15. Now you can run $test\_model\_ver1.slx$.