Xiaotong Sun XSEDE Username: xs018 Nov 11, 2021

HW 5
Machine Learning
RNN

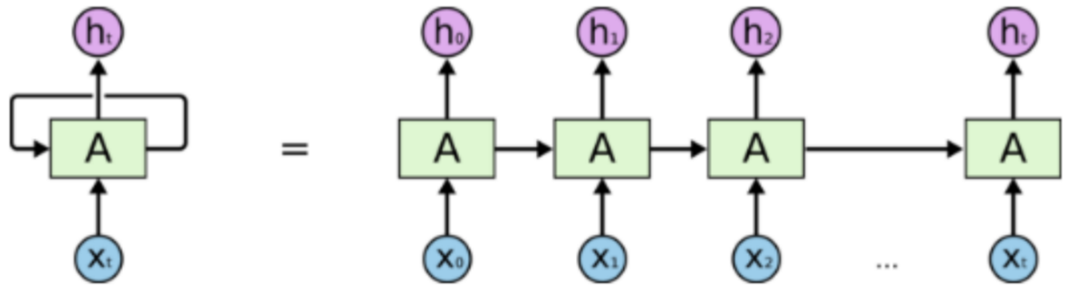
HW 5 Problem Statement (provided by Dr. Han Hu, MEEG 491V/591V-028, Fall 2021)
Problem 5-1 Time Series Prediction:
Re-do Problem 1-2 using PCA-MLP. Run SVD or PCA to obtain the PCs of the images. Feed the PCs to an MLP neural network to classify the regime of the boiling imagesThe data file "DS - 1_36W_vapor_fraction.txt" under /ocean/projects/mch210006p/shared/HW5 includes the vapor fraction (second column, dimensionless) vs. time (first column, unit: ms) of the boiling image sequences. The data are sampled with a frequency of 3,000 Hz (namely, a time step of 0.33 ms). Develop a recurrent neural network (RNN) model to forecast vapor fraction of future frames based on the past frames, e.g., predicting the vapor fraction profile of t = 33.33 ms – 66 ms using the vapor fraction history of t = 0.33 – 33 ms. Options include regular RNN, bidirectional RNN, gated recurrent unit (GRU), bidirectional GRU, long short-term memory (LSTM), bidirectional LSTM.

(a) Develop a baseline model with an input sequence length of 16.33 ms (50 data points) and an output
sequence length of 16.33 ms (50 data points). Plot the model-predicted signal vs. the true signal.

(b) Vary the input and output sequence lengths to evaluate their effect on the error of the model predictions.

Code Running Instructions
The environment is default.

Recurrent Neural Network (RNN)
The different with other neural networks is that RNN connected internal states, the input of next sequence uses the output of the former sequence.



An unrolled recurrent neural network.

Unlike standard RNNs whose repeating nodes have a simple structure such as a single tanh layer, Long Short Term Memory networks (LSTMs) has a more complex structure in the module which can involve long term memory among nodes.

Coding explanation
First step: Loading dataset.

The dataset is loaded by pandas.read_csv. Since we only need the second column of the dataset, "useclos = [1]".
Second step: Splitting the dataset into train and test datasets.
The training and testing data are divided by 9:1 by "train_test_split" function from sklearn. Then, the most important step is to set up the sequence of the data.
Third step: split the sequence.
I didn't use "for loop" to find the sequence for each time, I use the matrix calculation so that the running time is much shorter than the "for loop". Since the sequences should make as "1, 2, …, 50. 2, 3, 4, …, 51. … 4901, 4902, …, 4950." as the input, the second, third, …, last sequence only need to add 1, 2, 3, …, 4900 for each time to the first sequence. Then use the broadcasting to make a sequence matrix and flatten it.
Fourth steps: Get the X, Y sequence.
X use the first 4900 sequences without the last sequence, Y use the last 4900 sequences without the first sequence.
Last step: create the RNN model and add LSTM layer.
Finally, train, validate, and test the LSTM model.

Evaluation: the validation loss and mean square error is used as the evaluation.

Results
a). The graph of the predicted signal and true signal is shown below. Because it looks not that good when only shown in one sequence, the whole sequence in testing is shown.

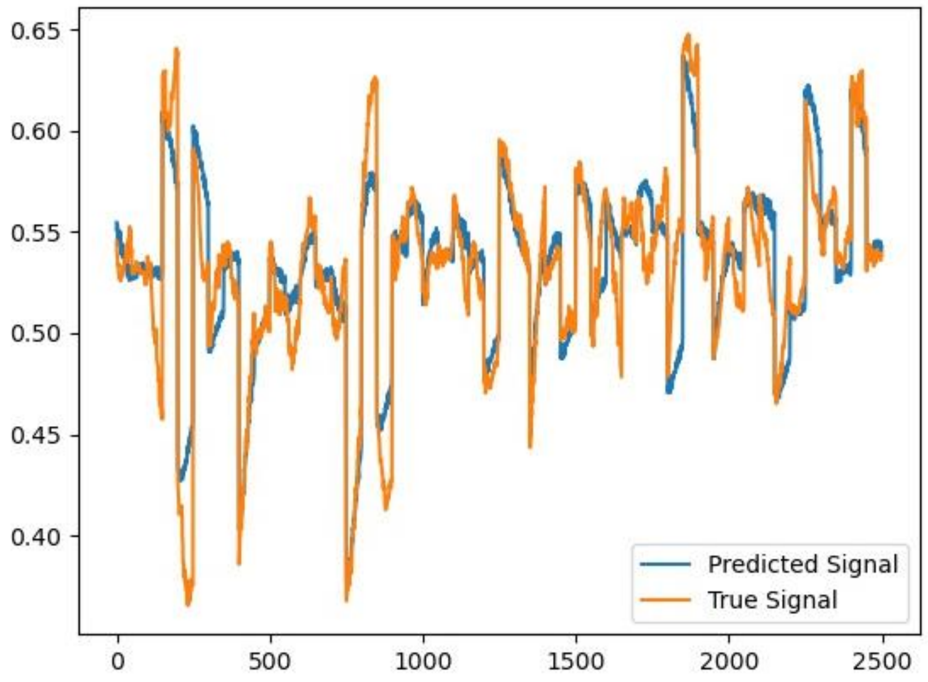


Fig 1. Predicted signal and true signal (length 50).

The result of running time, epochs, model structure, and test mean square error is shown below:

Running epochs, time, and validation loss:

```
Epoch 1/100
3528/3528 - 5s - loss: 0.0987 - val_loss: 0.0029
Epoch 2/100
3528/3528 - 3s - loss: 0.0020 - val_loss: 0.0017
Epoch 3/100
3528/3528 - 3s - loss: 0.0015 - val_loss: 0.0014
Epoch 4/100
3528/3528 - 3s - loss: 0.0013 - val_loss: 0.0016
Epoch 5/100
3528/3528 - 3s - loss: 0.0013 - val_loss: 0.0013
Epoch 6/100
3528/3528 - 3s - loss: 0.0012 - val_loss: 0.0014
Epoch 7/100
3528/3528 - 3s - loss: 0.0012 - val_loss: 0.0012
Epoch 8/100
3528/3528 - 3s - loss: 0.0013 - val_loss: 0.0012
Epoch 9/100
3528/3528 - 3s - loss: 0.0012 - val_loss: 0.0012
Epoch 10/100
3528/3528 - 3s - loss: 0.0011 - val_loss: 0.0012
Epoch 11/100
3528/3528 - 3s - loss: 0.0012 - val_loss: 0.0012
Epoch 12/100
3528/3528 - 3s - loss: 0.0011 - val_loss: 0.0011
Epoch 13/100
3528/3528 - 3s - loss: 0.0011 - val_loss: 0.0011
Epoch 14/100
3528/3528 - 3s - loss: 0.0011 - val_loss: 0.0011
Epoch 15/100
3528/3528 - 3s - loss: 0.0011 - val_loss: 0.0011
```

```
Epoch 16/100
3528/3528 - 3s - loss: 0.0011 - val_loss: 0.0011
Epoch 17/100
3528/3528 - 3s - loss: 0.0011 - val_loss: 0.0011
Epoch 18/100
3528/3528 - 3s - loss: 0.0010 - val_loss: 0.0011
Epoch 19/100
3528/3528 - 3s - loss: 0.0010 - val_loss: 0.0010
Epoch 20/100
3528/3528 - 3s - loss: 9.9036e-04 - val_loss: 9.9271e-04
Epoch 21/100
3528/3528 - 3s - loss: 0.0010 - val_loss: 0.0010
Epoch 22/100
3528/3528 - 3s - loss: 9.5182e-04 - val_loss: 9.6199e-04
Epoch 23/100
3528/3528 - 3s - loss: 9.4207e-04 - val_loss: 9.5360e-04
Epoch 24/100
3528/3528 - 3s - loss: 9.5055e-04 - val_loss: 9.2635e-04
Epoch 25/100
3528/3528 - 3s - loss: 9.0706e-04 - val_loss: 9.9548e-04
Epoch 26/100
3528/3528 - 3s - loss: 8.9674e-04 - val_loss: 9.4400e-04
Epoch 27/100
3528/3528 - 3s - loss: 8.8141e-04 - val_loss: 8.6345e-04
Epoch 28/100
3528/3528 - 3s - loss: 8.4708e-04 - val_loss: 8.8592e-04
Epoch 29/100
3528/3528 - 3s - loss: 8.6212e-04 - val_loss: 8.3101e-04
Epoch 30/100
3528/3528 - 3s - loss: 8.1563e-04 - val_loss: 8.5486e-04
```

```
Epoch 31/100
3528/3528 - 3s - loss: 8.1116e-04 - val_loss: 8.1703e-04
Epoch 32/100
3528/3528 - 3s - loss: 8.1647e-04 - val_loss: 8.9086e-04
Epoch 33/100
3528/3528 - 3s - loss: 8.0094e-04 - val_loss: 8.3046e-04
Epoch 34/100
3528/3528 - 3s - loss: 8.0899e-04 - val_loss: 7.7525e-04
Epoch 35/100
3528/3528 - 3s - loss: 7.8347e-04 - val_loss: 7.7360e-04
Epoch 36/100
3528/3528 - 3s - loss: 7.6657e-04 - val_loss: 8.0623e-04
Epoch 37/100
3528/3528 - 3s - loss: 7.6848e-04 - val_loss: 7.9061e-04
Epoch 38/100
3528/3528 - 3s - loss: 7.8638e-04 - val_loss: 7.6048e-04
Epoch 39/100
3528/3528 - 3s - loss: 7.6827e-04 - val_loss: 8.1253e-04
Epoch 40/100
3528/3528 - 3s - loss: 7.6865e-04 - val_loss: 7.9322e-04
Epoch 41/100
3528/3528 - 3s - loss: 7.7311e-04 - val_loss: 7.6635e-04
Epoch 42/100
3528/3528 - 3s - loss: 7.5013e-04 - val_loss: 8.0829e-04
Epoch 43/100
Restoring model weights from the end of the best epoch.
3528/3528 - 3s - loss: 7.4828e-04 - val_loss: 7.8856e-04
Epoch 00043: early stopping
```

The model structure:

```
_________________________________________________________________
Layer (type)            Output Shape          Param #
=================================================================
lstm (LSTM)              multiple              66560
_________________________________________________________________
dense (Dense)             multiple              6450
=================================================================
Total params: 73,010
Trainable params: 73,010
Non-trainable params: 0
_________________________________________________________________
Test mean square error: 0.0007
```

b). I used the length of sequence with 10, 20, 100 as the comparison.

The result of length with 10 is shown below:

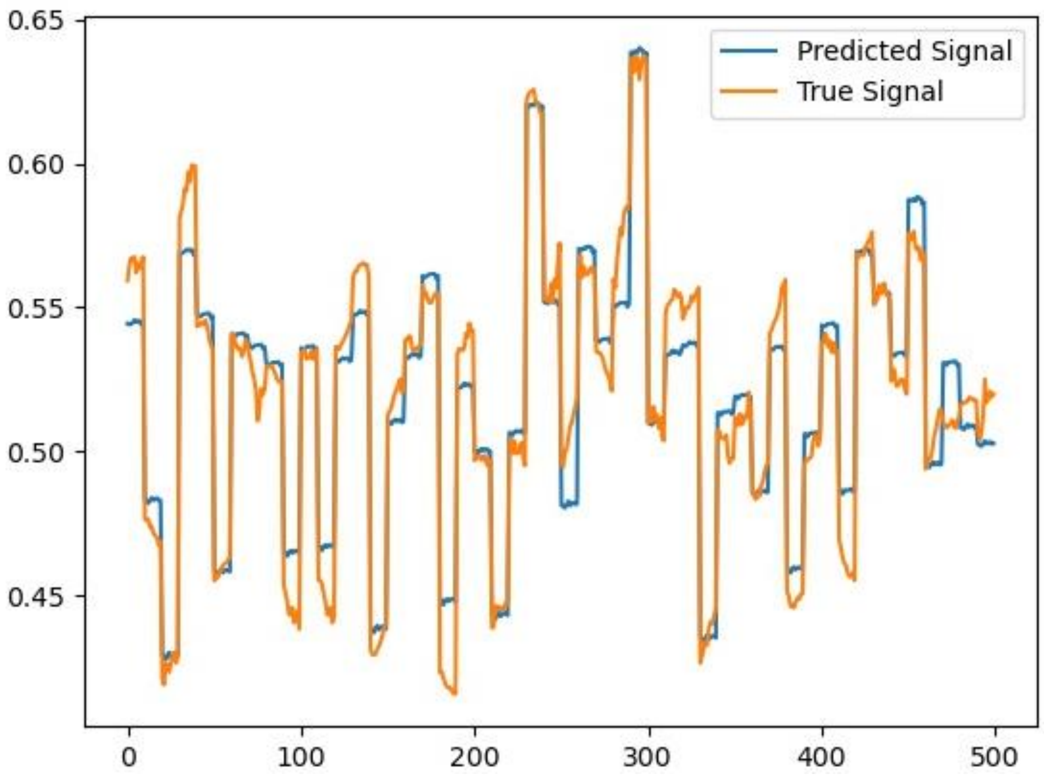


Fig 2. Predicted signal and true signal (length 10)

The result of each epoch and model structure is shown as follows:

```
Train on 3585 samples, validate on 897 samples

Epoch 1/100

3585/3585 - 3s - loss: 0.0505 - val_loss: 2.1846e-04

Epoch 2/100

3585/3585 - 1s - loss: 2.2070e-04 - val_loss: 1.9925e-04

Epoch 3/100

3585/3585 - 1s - loss: 2.1727e-04 - val_loss: 2.4452e-04

Epoch 4/100

3585/3585 - 1s - loss: 2.2035e-04 - val_loss: 2.4875e-04

Epoch 5/100

3585/3585 - 1s - loss: 2.1588e-04 - val_loss: 1.9930e-04

Epoch 6/100

3585/3585 - 1s - loss: 2.1397e-04 - val_loss: 1.9412e-04

Epoch 7/100

3585/3585 - 1s - loss: 2.0681e-04 - val_loss: 1.9101e-04

Epoch 8/100

3585/3585 - 1s - loss: 2.1156e-04 - val_loss: 2.2776e-04

Epoch 9/100

3585/3585 - 1s - loss: 2.1184e-04 - val_loss: 2.7747e-04

Epoch 10/100

3585/3585 - 1s - loss: 2.0959e-04 - val_loss: 3.5826e-04

Epoch 11/100

3585/3585 - 1s - loss: 2.3109e-04 - val_loss: 1.8814e-04

Epoch 12/100

3585/3585 - 1s - loss: 2.1630e-04 - val_loss: 3.9831e-04

Epoch 13/100

3585/3585 - 1s - loss: 2.3323e-04 - val_loss: 4.9775e-04

Epoch 14/100

3585/3585 - 1s - loss: 2.2194e-04 - val_loss: 2.6277e-04
```

```
Epoch 15/100

3585/3585 - 1s - loss: 2.2087e-04 - val_loss: 4.4799e-04

Epoch 16/100

Restoring model weights from the end of the best epoch.

3585/3585 - 1s - loss: 2.2739e-04 - val_loss: 1.8955e-04

Epoch 00016: early stopping

Model: "sequential"

_________________________________________________________________

Layer (type)              Output Shape           Param #

=================================================================

lstm (LSTM)               multiple               66560

_________________________________________________________________

dense (Dense)             multiple               1290

=================================================================

Total params: 67,850

Trainable params: 67,850

Non-trainable params: 0

_________________________________________________________________.
```

The test mean square error is 0.0002, which is better than the test mean square with length 50.

The result of length with 20 is shown below:

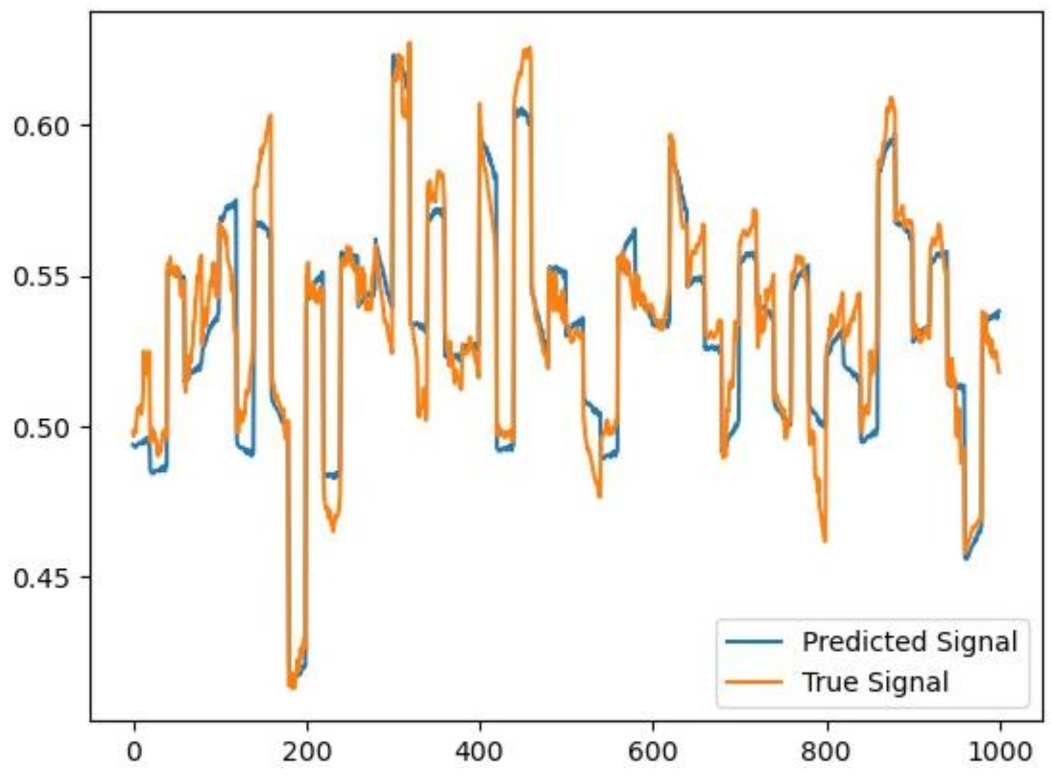


Fig 3. Predicted signal and true signal (length 20)

The result of each epoch and model structure is shown as follows:

Train on 3571 samples, validate on 893 samples

Epoch 1/100

3571/3571 - 3s - loss: 0.0390 - val_loss: 5.7034e-04

Epoch 2/100

3571/3571 - 2s - loss: 5.6644e-04 - val_loss: 5.8744e-04

Epoch 3/100

3571/3571 - 2s - loss: 5.6130e-04 - val_loss: 6.3311e-04

Epoch 4/100

3571/3571 - 2s - loss: 5.7601e-04 - val_loss: 5.6122e-04

Epoch 5/100

3571/3571 - 2s - loss: 5.7746e-04 - val_loss: 5.4298e-04

Epoch 6/100

3571/3571 - 2s - loss: 5.4760e-04 - val_loss: 5.3675e-04

Epoch 7/100

```
3571/3571 - 2s - loss: 5.3320e-04 - val_loss: 5.2835e-04
Epoch 8/100
3571/3571 - 2s - loss: 5.2708e-04 - val_loss: 5.3707e-04
Epoch 9/100
3571/3571 - 2s - loss: 6.0942e-04 - val_loss: 5.1551e-04
Epoch 10/100
3571/3571 - 2s - loss: 5.5244e-04 - val_loss: 5.6220e-04
Epoch 11/100
3571/3571 - 2s - loss: 5.4216e-04 - val_loss: 6.7949e-04
Epoch 12/100
3571/3571 - 2s - loss: 5.2225e-04 - val_loss: 5.5727e-04
Epoch 13/100
3571/3571 - 2s - loss: 5.9309e-04 - val_loss: 4.9416e-04
Epoch 14/100
3571/3571 - 2s - loss: 5.4860e-04 - val_loss: 4.7075e-04
Epoch 15/100
3571/3571 - 2s - loss: 4.8796e-04 - val_loss: 5.6662e-04
Epoch 16/100
3571/3571 - 2s - loss: 4.6362e-04 - val_loss: 4.8987e-04
Epoch 17/100
3571/3571 - 2s - loss: 4.7232e-04 - val_loss: 4.3585e-04
Epoch 18/100
3571/3571 - 2s - loss: 4.1122e-04 - val_loss: 3.7871e-04
Epoch 19/100
3571/3571 - 2s - loss: 4.4803e-04 - val_loss: 3.7253e-04
Epoch 20/100
3571/3571 - 2s - loss: 4.1300e-04 - val_loss: 5.5753e-04
Epoch 21/100
3571/3571 - 2s - loss: 3.5557e-04 - val_loss: 3.2238e-04
Epoch 22/100
```

```
3571/3571 - 2s - loss: 3.7127e-04 - val_loss: 3.0521e-04
Epoch 23/100
3571/3571 - 2s - loss: 3.3309e-04 - val_loss: 5.4921e-04
Epoch 24/100
3571/3571 - 2s - loss: 3.2290e-04 - val_loss: 3.3036e-04
Epoch 25/100
3571/3571 - 2s - loss: 3.6466e-04 - val_loss: 2.7921e-04
Epoch 26/100
3571/3571 - 2s - loss: 2.7293e-04 - val_loss: 2.8344e-04
Epoch 27/100
3571/3571 - 2s - loss: 2.9696e-04 - val_loss: 3.0453e-04
Epoch 28/100
3571/3571 - 2s - loss: 2.6617e-04 - val_loss: 2.2968e-04
Epoch 29/100
3571/3571 - 2s - loss: 2.5204e-04 - val_loss: 4.0734e-04
Epoch 30/100
3571/3571 - 2s - loss: 2.4657e-04 - val_loss: 2.2545e-04
Epoch 31/100
3571/3571 - 2s - loss: 2.4203e-04 - val_loss: 2.1678e-04
Epoch 32/100
3571/3571 - 2s - loss: 2.5790e-04 - val_loss: 3.4489e-04
Epoch 33/100
3571/3571 - 2s - loss: 2.4878e-04 - val_loss: 4.6594e-04
Epoch 34/100
3571/3571 - 2s - loss: 2.5791e-04 - val_loss: 2.2930e-04
Epoch 35/100
3571/3571 - 2s - loss: 2.3231e-04 - val_loss: 2.9053e-04
Epoch 36/100
3571/3571 - 2s - loss: 2.5767e-04 - val_loss: 2.1286e-04
Epoch 37/100
```

```
3571/3571 - 2s - loss: 2.4964e-04 - val_loss: 2.0259e-04
Epoch 38/100
3571/3571 - 2s - loss: 2.3425e-04 - val_loss: 5.3951e-04
Epoch 39/100
3571/3571 - 2s - loss: 2.7098e-04 - val_loss: 2.3335e-04
Epoch 40/100
3571/3571 - 2s - loss: 2.5953e-04 - val_loss: 2.5728e-04
Epoch 41/100
3571/3571 - 2s - loss: 2.5799e-04 - val_loss: 2.8511e-04
Epoch 42/100
Restoring model weights from the end of the best epoch.
3571/3571 - 2s - loss: 2.4147e-04 - val_loss: 2.3443e-04
Epoch 00042: early stopping
```

The test mean square error is 0.0002, which is same with length 10 and better than length 50.

The result of length with 100 is shown below:

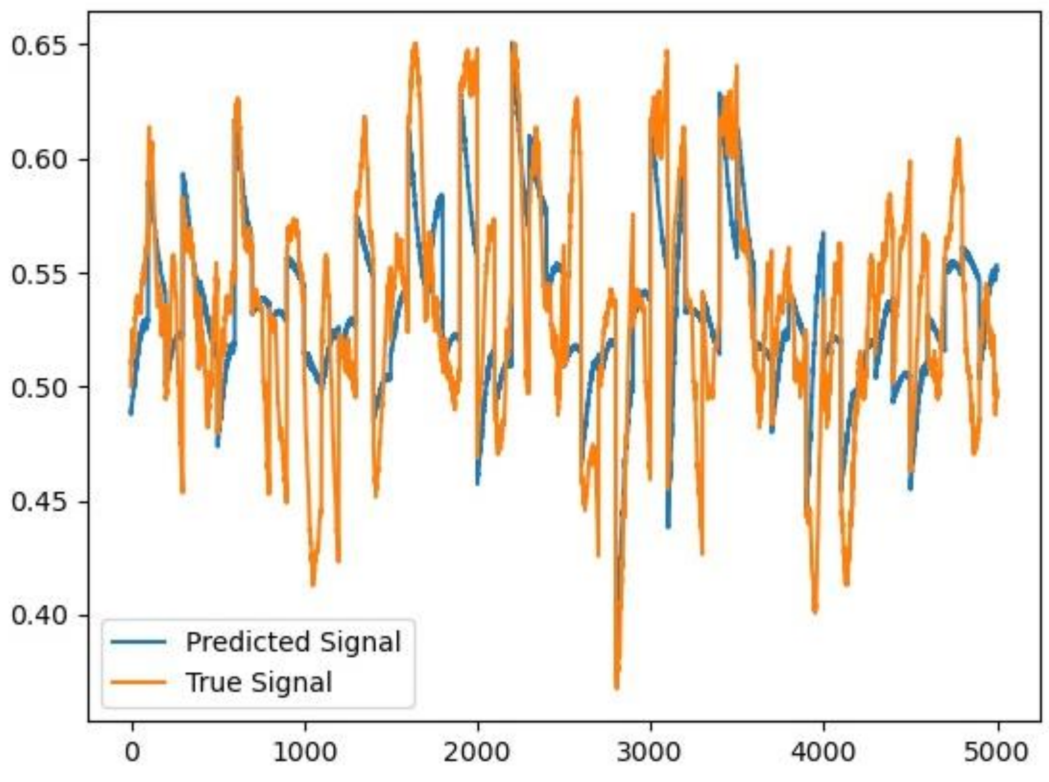


Fig 4. Predicted signal and true signal (length 100)

The result of each epoch and model structure is shown as follows:

```
Train on 3456 samples, validate on 864 samples

Epoch 1/100

3456/3456 - 11s - loss: 1515590.2725 - val_loss: 0.1652

Epoch 2/100

3456/3456 - 9s - loss: 0.1694 - val_loss: 0.1100

Epoch 3/100

3456/3456 - 6s - loss: 0.0981 - val_loss: 0.0775

Epoch 4/100

3456/3456 - 8s - loss: 0.0243 - val_loss: 0.0033

Epoch 5/100

3456/3456 - 8s - loss: 0.0029 - val_loss: 0.0030

Epoch 6/100

3456/3456 - 6s - loss: 0.0028 - val_loss: 0.0029

Epoch 7/100

3456/3456 - 8s - loss: 0.0027 - val_loss: 0.0028

Epoch 8/100

3456/3456 - 6s - loss: 0.0026 - val_loss: 0.0027

Epoch 9/100

3456/3456 - 8s - loss: 0.0025 - val_loss: 0.0026

Epoch 10/100

3456/3456 - 8s - loss: 0.0025 - val_loss: 0.0026

Epoch 11/100

3456/3456 - 6s - loss: 0.0024 - val_loss: 0.0025

Epoch 12/100

3456/3456 - 8s - loss: 0.0024 - val_loss: 0.0025

Epoch 13/100

3456/3456 - 6s - loss: 0.0024 - val_loss: 0.0025

Epoch 14/100
```

```
3456/3456 - 9s - loss: 0.0023 - val_loss: 0.0024
Epoch 15/100
3456/3456 - 6s - loss: 0.0023 - val_loss: 0.0024
Epoch 16/100
3456/3456 - 8s - loss: 0.0023 - val_loss: 0.0023
Epoch 17/100
3456/3456 - 8s - loss: 0.0022 - val_loss: 0.0023
Epoch 18/100
3456/3456 - 7s - loss: 0.0022 - val_loss: 0.0023
Epoch 19/100
3456/3456 - 9s - loss: 0.0022 - val_loss: 0.0022
Epoch 20/100
3456/3456 - 6s - loss: 0.0021 - val_loss: 0.0022
Epoch 21/100
3456/3456 - 9s - loss: 0.0021 - val_loss: 0.0022
Epoch 22/100
3456/3456 - 9s - loss: 0.0021 - val_loss: 0.0021
Epoch 23/100
3456/3456 - 6s - loss: 0.0020 - val_loss: 0.0021
Epoch 24/100
3456/3456 - 7s - loss: 0.0020 - val_loss: 0.0020
Epoch 25/100
3456/3456 - 5s - loss: 0.0020 - val_loss: 0.0020
Epoch 26/100
3456/3456 - 6s - loss: 0.0019 - val_loss: 0.0019
Epoch 27/100
3456/3456 - 5s - loss: 0.0019 - val_loss: 0.0019
Epoch 28/100
3456/3456 - 6s - loss: 0.0019 - val_loss: 0.0019
Epoch 29/100
```

```
3456/3456 - 5s - loss: 0.0018 - val_loss: 0.0019
Epoch 30/100
3456/3456 - 8s - loss: 0.0018 - val_loss: 0.0019
Epoch 31/100
3456/3456 - 6s - loss: 0.0018 - val_loss: 0.0018
Epoch 32/100
3456/3456 - 7s - loss: 0.0018 - val_loss: 0.0018
Epoch 33/100
3456/3456 - 7s - loss: 0.0018 - val_loss: 0.0019
Epoch 34/100
3456/3456 - 7s - loss: 0.0018 - val_loss: 0.0018
Epoch 35/100
3456/3456 - 6s - loss: 0.0018 - val_loss: 0.0017
Epoch 36/100
3456/3456 - 6s - loss: 0.0017 - val_loss: 0.0019
Epoch 37/100
3456/3456 - 6s - loss: 0.0017 - val_loss: 0.0019
Epoch 38/100
3456/3456 - 6s - loss: 0.0017 - val_loss: 0.0017
Epoch 39/100
3456/3456 - 7s - loss: 0.0017 - val_loss: 0.0016
Epoch 40/100
3456/3456 - 7s - loss: 0.0016 - val_loss: 0.0016
Epoch 41/100
3456/3456 - 6s - loss: 0.0017 - val_loss: 0.0016
Epoch 42/100
3456/3456 - 6s - loss: 0.0016 - val_loss: 0.0017
Epoch 43/100
3456/3456 - 6s - loss: 0.0016 - val_loss: 0.0016
Epoch 44/100
```

```
3456/3456 - 7s - loss: 0.0016 - val_loss: 0.0017

Epoch 45/100

3456/3456 - 6s - loss: 0.0016 - val_loss: 0.0015

Epoch 46/100

3456/3456 - 7s - loss: 0.0015 - val_loss: 0.0015

Epoch 47/100

3456/3456 - 7s - loss: 0.0015 - val_loss: 0.0017

Epoch 48/100

3456/3456 - 6s - loss: 0.0016 - val_loss: 0.0015

Epoch 49/100

3456/3456 - 6s - loss: 0.0015 - val_loss: 0.0015

Epoch 50/100

3456/3456 - 7s - loss: 0.0015 - val_loss: 0.0015

Epoch 51/100

3456/3456 - 6s - loss: 0.0015 - val_loss: 0.0015

Epoch 52/100

3456/3456 - 7s - loss: 0.0015 - val_loss: 0.0016

Epoch 53/100

3456/3456 - 6s - loss: 0.0015 - val_loss: 0.0015

Epoch 54/100

3456/3456 - 7s - loss: 0.0016 - val_loss: 0.0016

Epoch 55/100

3456/3456 - 6s - loss: 0.0015 - val_loss: 0.0015

Epoch 56/100

Restoring model weights from the end of the best epoch.

3456/3456 - 7s - loss: 0.0015 - val_loss: 0.0015

Epoch 00056: early stopping

Model: "sequential"

_________________________________________________________________

Layer (type)                 Output Shape              Param #
```

```
=================================================================
lstm (LSTM)              multiple              66560
_________________________________________________________________
dense (Dense)            multiple              12900
=================================================================
Total params: 79,460
Trainable params: 79,460
Non-trainable params: 0
_________________________________________________________________
```

The test mean square error is 0.0015 which is worse that length 10, 20, 50. In addition, it takes the longest running time among those tests, about 6 minutes.

Above all, when the length of sequence is short, we will get good predictions.

Challenges

First challenge in this assignment is load the dataset. The dataset is Nan when I use “df = pd.read_csv(url, usecols=[1], engine='python'”, until the “sep = "\t"” is added. I tried “sep = " ")”, because blank always be the separator, but for this dataset, “/t” is used.

Second challenge is splitting the sequence and match it with the predicted data X, and tested data Y.

References

[1] https://machinelearningmastery.com/understanding-simple-recurrent-neural-networks-in-keras/.

[2] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e