Anomaly Detection at the IoT Edge using Deep Learning

Darmawan Utomo
Prof. Pao-Ann Hsiung

Embedded Systems Laboratory
Department of Computer Science and Information Engineering
National Chung Cheng University
Taiwan

Outline

- Introduction
- Anomaly Detection System Design
  - Data Modeling
  - Anomaly Detection Modeling
- Experiments
- Conclusions and Future Work

Outline

- Introduction
- Anomaly Detection System Design
  - Data Modeling
  - Anomaly Detection Modeling
- Experiments
- Conclusions and Future Work

Introduction

- By the end of 2016, according to [1], number of smart meters deployed in UK, US, and China reached 2.9 million, 70 million, and 96 million, respectively.
- These produce so tremendous amount of data that are hard to store and analyze in the cloud.

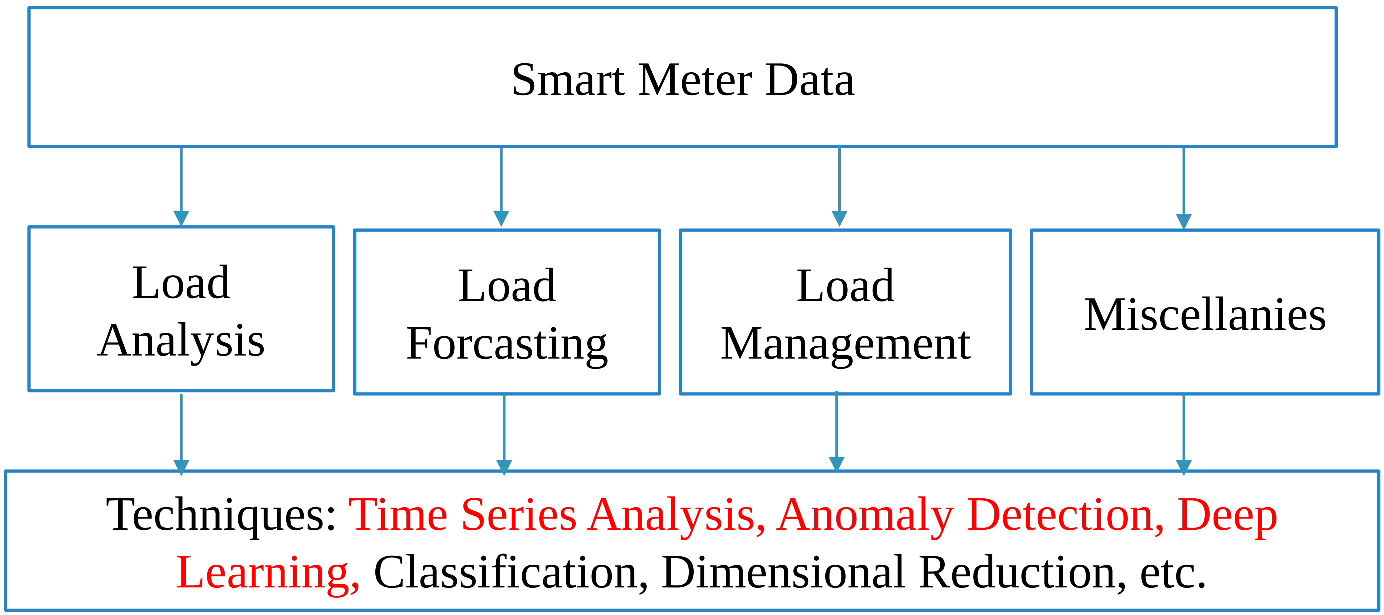
[1] Y. Wang, Q. Chen, T. Hong and C. Kang, "Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges," in *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 3125-3148, May 2019.

Introduction (cont.)

- Thus IoT Edge become very important in this environment.
- Edge offers the ability to collect, analyze, calculate, and store the smart meter data before sending it to the cloud.
- Research on smart meter mainly can be classified into four main categories [1] namely Load Analysis and Load Forcasting, Load Management, and Miscellanies.

Introduction (cont.)

- Common data categories used for Analysis and Techniques [1]:



Introduction (cont.)

- Motivation:
  - The previous review stated that using time series data, anomaly detection, and Deep Learning are growing concerns that should be considered in analysis smart meter data.
  - On the others paper, Andrysiak proposed a statistics-based anomaly detector for predicting smart meter data ourliers [2] and Araya proposed a framework for anomaly detection by implementing an ensemble learning [3].
  - But, [2] and [3] are in a learning contrast. In [2] without learning, and to decide whether there is an anomaly or not, [3] use voting of some learner models. [3] seem generating large latency that may not suitable for Edge computing.

[2] T. Andrysiak, L. Saganowski, and P. Kiedrowski, “Anomaly detection in smart metering infrastructure with the use of time series analysis,” J. Sensors, vol. 2017.
[3] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. Capretz, and G. Bitsuamlak, “An ensemble learning framework for anomaly detection in building energy consumption,” Energy and Buildings, vol. 144, pp. 191 – 206, 2017.

Motivation (cont.)

- Goal
  - Detect anomaly in smart meter using time series data and deep learning.
  - Implement it in an IoT Edge.

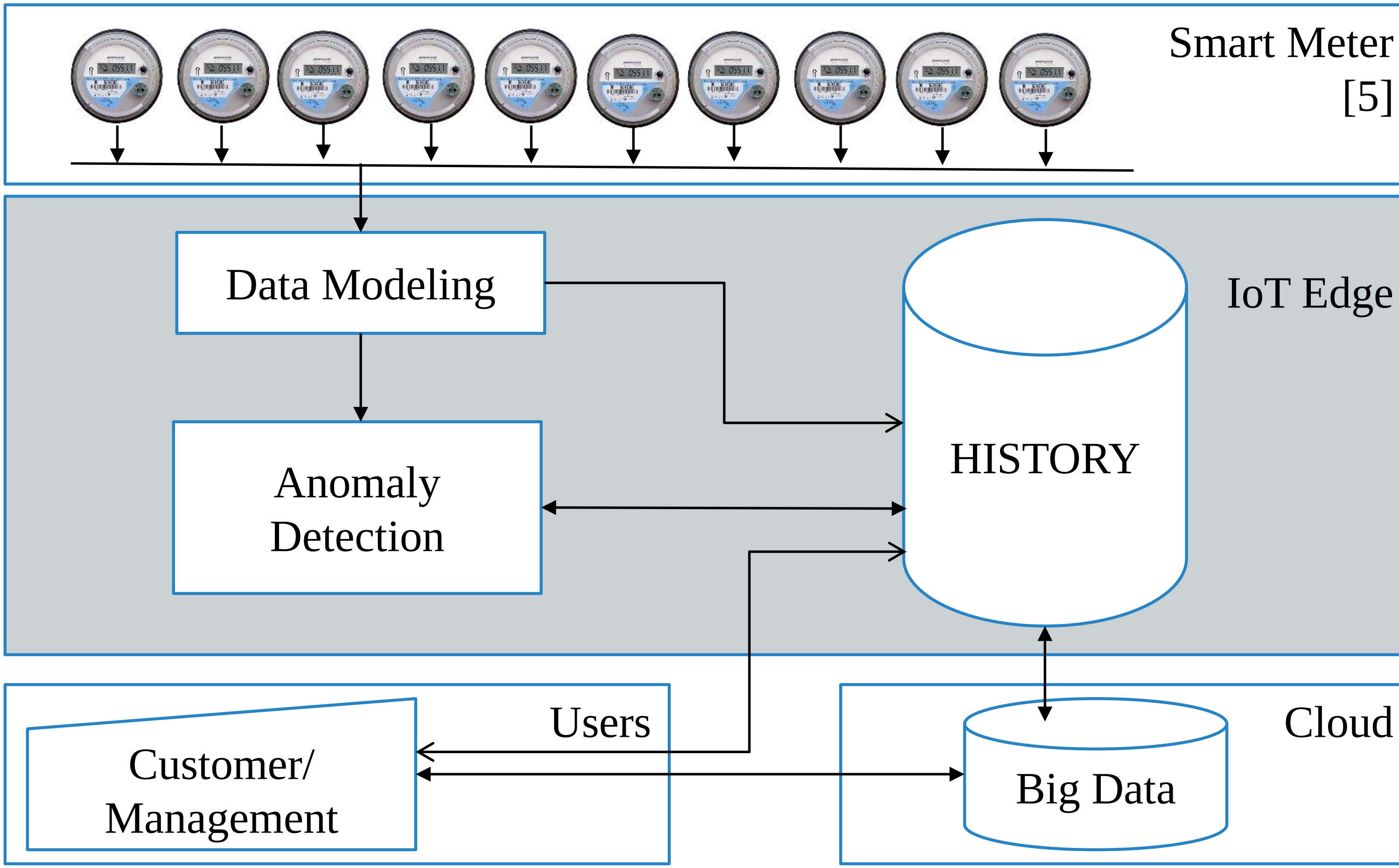
- Proposed method
  - Using time series data and Long short term memory (LSTM [4]) as a DNN model.
  - Implement it in a Raspberry Pi 3B as an IoT Edge example.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735 – 1780, Nov. 1997.

Outline

- Introduction
- Anomaly Detection System Design
  - Data Modeling
  - Anomaly Detection Modeling
- Experiments
- Conclusions and Future Work

Anomaly Detection at the IoT Edge
System Architecture



[5] Smart meter image: http://www.smartmetereducationnetwork.com/how-to-tell-if-I-have-a-ami-dte-smart-advanced-meter.php

Anomaly Detection at the IoT Edge System:

Training, step-by-step:

1. Read the training data
2. Normalized the data value
3. Split the data into training and testing data
4. model. Compile (loss = 'binary_crossentropy', optimizer = 'adam',
   metrics = ['accuracy'])
5. model.fit (train, ytrain, shuffle = True, validation_split = 0.5)
6. Save the model and weights to .json and .h5.
7. score1 = model.evaluate (train, ytrain)

Anomaly Detection at the IoT Edge System:

Testing step-by-step

1. Read the testing data
2. Load model.json and weights.h5.
3. Detection = model.predict (testing, ytest)
4. Evaluate (Detection) → Confusion Matrix

Evaluation Formula:

- Evaluation
  - TP (true positive)
  - FP (false positive)
  - TN (true negative)
  - FN (false negative)
  - Validation accuracy

	T	**F**
P	TP (1,1)	FP (1,0)
N	FN (0,1)	TN (0,0)

- 1: Anomaly
- 0: Normal
- (Predict, Actual)

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

$$Validation\ Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

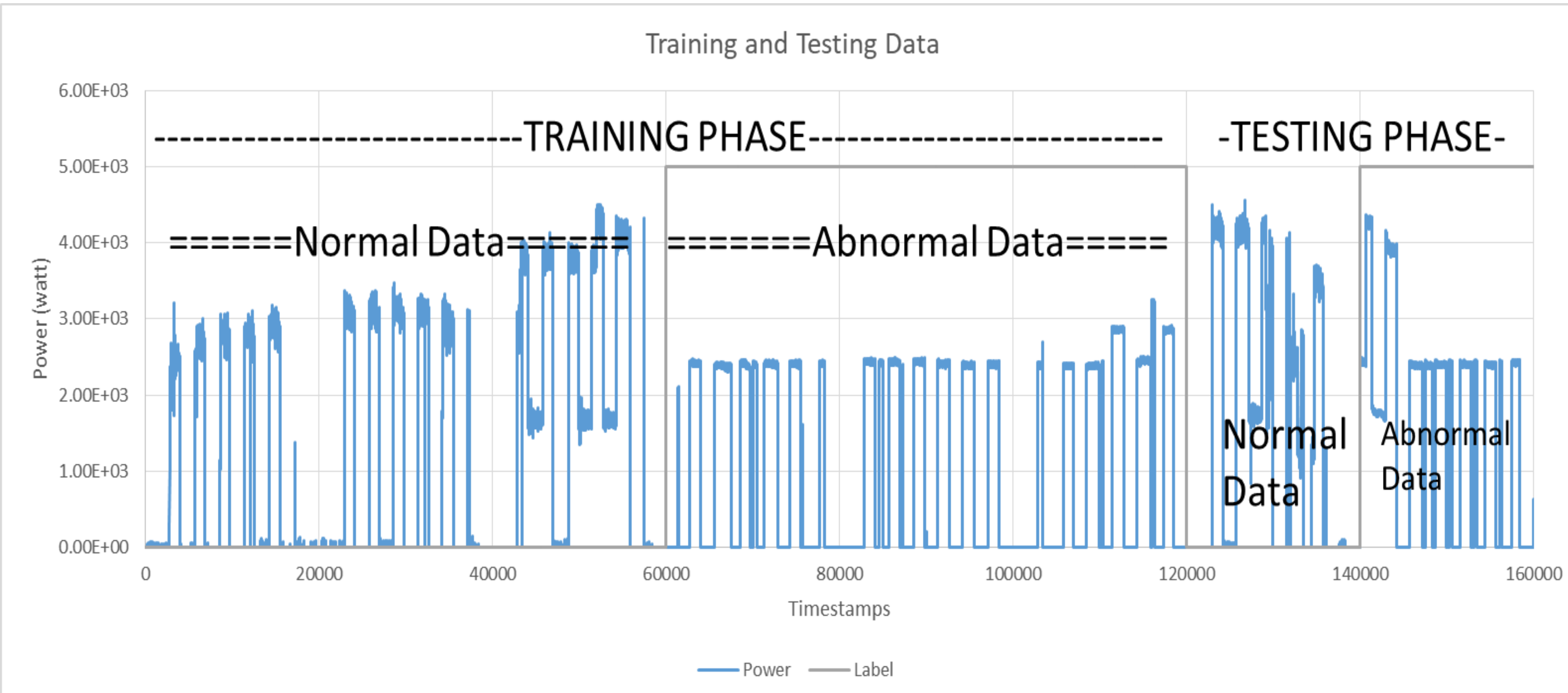
Outline

- Introduction
- Anomaly Detection System Design
  - Data Modeling
  - Anomaly Detection Modeling
- Experiments
- Conclusions and Future Work

Data Modeling

- Two smart meters data are used as normal and anomalous data sources.
- Normal data : 80,000 point data
  - 60,000 data points for training and validation
  - 20,000 data points for testing
- Anomalous data : 80,000 point data
  - 60,000 data points for training and validation
  - 20,000 data points for testing
- Each timestep contains 200 data points (timestamp)
  - Training Normal & Anomalous data: (300, 200, 1) (300,) x 2
  - Testing Normal & Anomalous data: (100, 200, 1) (100,) x 2
- Total for training (600, 200, 1) (600,) and testing (200, 200, 1) (200,)

Data Modeling (cont.)



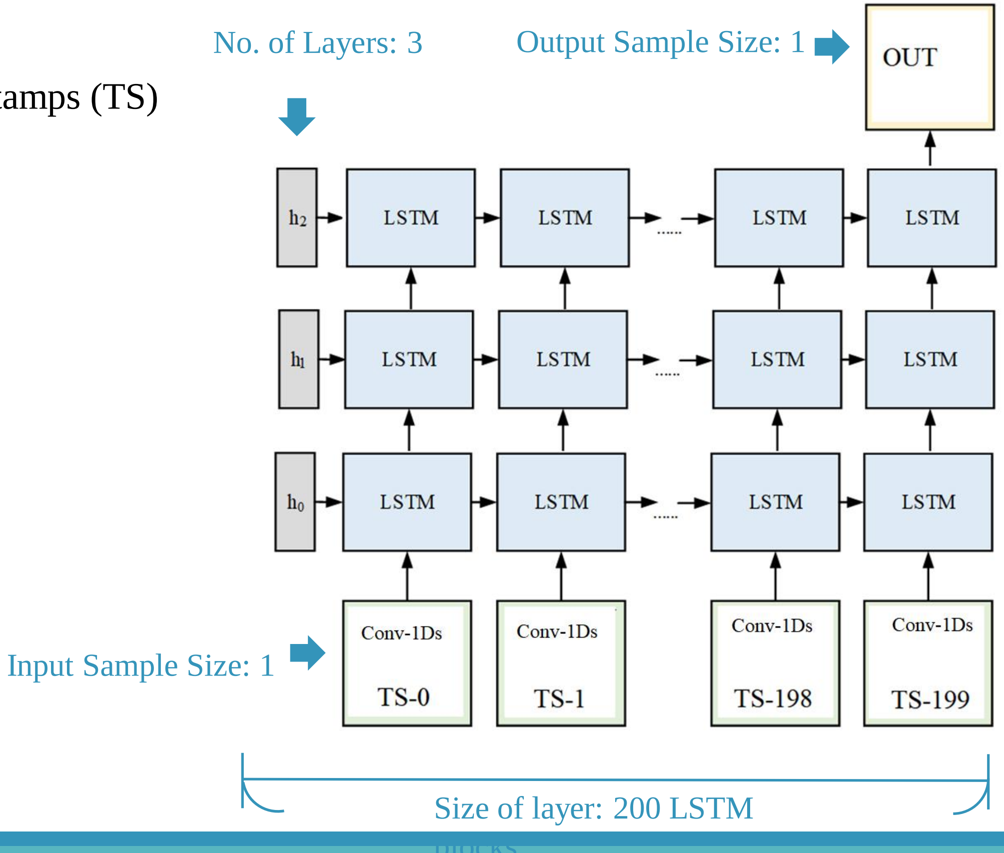
Anomaly Detection Modeling

- Recurrent Neural Network (RNN) is a Deep Neural Network that is suitable for time series data.
  - Long Short Term Memory (LSTM)

- Supervised Learning (data with label)

Anomaly Detection Modeling (cont.): Detection Module

- Every 200 timestamps are used to detect normal or abnormal events.
- Parameter settings
  - Size of Layer: 200 LSTM timestamps (TS)
  - Input Size: 1
  - Output Size: 1
  - No. of Layers: 3
  - Batch Size: 25
  - Epochs: 100



Outline

- Introduction
- Anomaly Detection System Design
  - Data Modeling
  - Anomaly Detection Modeling
- Experiments
- Conclusions and Future Work

Experiments

- Experiment setup

Item	Type
Raspberry Pi 3B+ CPU	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
Memory, MMC	1 GB RAM, 16 GB Flash
Operating System	Raspbian
Communication	BCM43438 wireless LAN and Bluetooth Low Energy (BLE)
Voltage, Idle-max Power	5 Volts, 1.7 W – 5.1 W
Programming Language	Python 3.5
Programming Framework	Keras, Tensorflow

Experiments (cont.)

- Experiment data

Name	Description
Dataset	Two smart meters
Sampling	Every 30 seconds for 1 month
Training data set	• 300 timesteps from Normal data • 300 timesteps from Anomalous data
Testing data set	• 100 timesteps from Normal data • 100 timesteps from Anomalous data

- All the training data and the testing data are obtained from [6].

[6] N. Batra, O. Parson, M. Berges, A. Singh, and A. Rogers, "A comparison of non-intrusive load monitoring methods for commercial and residential buildings," CoRR, vol. abs/1408.6595, 2014.

Experiments (cont.)

- Experiments
  - Training & Validation
  - The Performance of the Anomaly Detection
  - User Time Allocations

Experiments (cont.): 1. Training and Validation the dataset

- Experiment settings

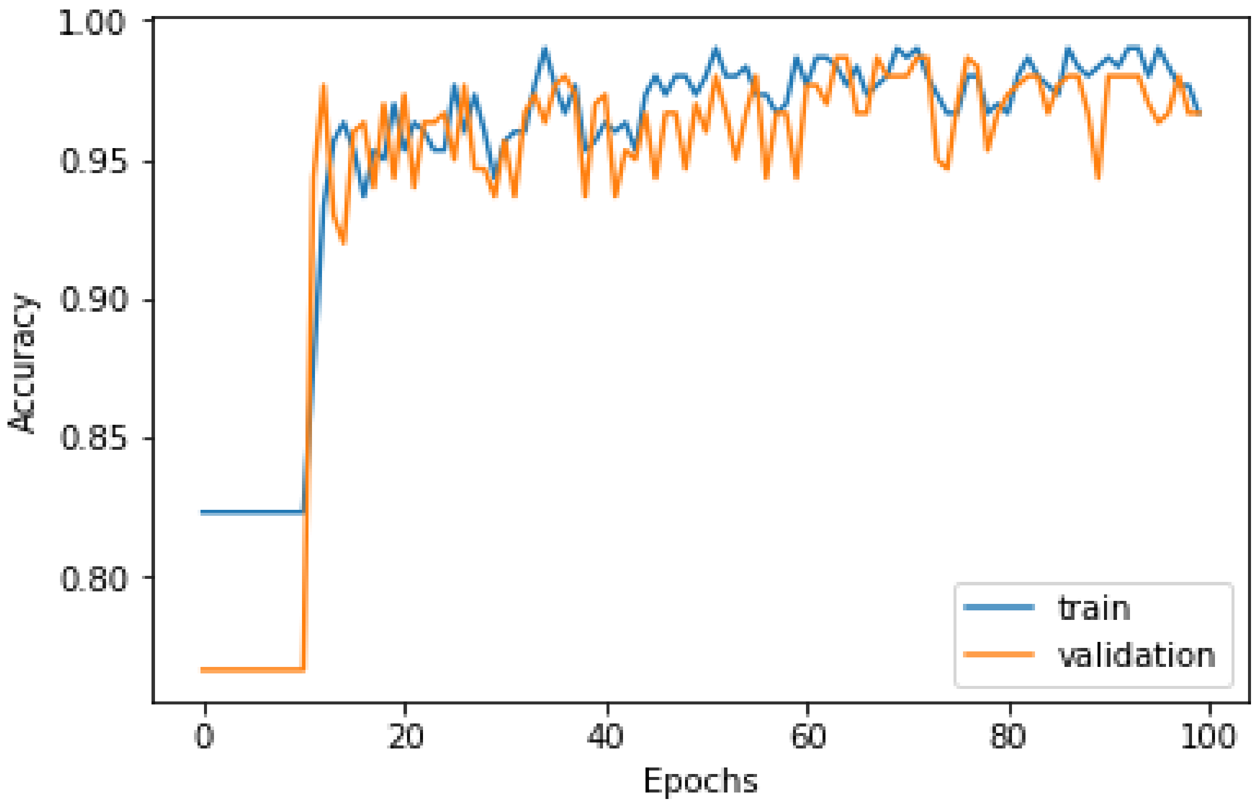
```
BATCHSIZE = 25;  #INTERVAL = TIME-STEPS
EPOCH = 100
model = Sequential ()
model.add (Conv1D (4, 3, strides = 3, activation = 'relu', input_shape = (INTERVAL, 1)))
model.add (Conv1D (8, 3, padding = "same", activation = 'relu'))
model.add (MaxPooling1D (3))
model.add (Conv1D (16, 5, padding = "same", activation = 'relu'))
model.add (MaxPooling1D (4))
model.add (Conv1D (32, 7, padding = "same", activation = 'relu'))
model.add (Conv1D (64, 7, padding = "same", activation = 'relu'))
model.add (MaxPooling1D (3))
model.add (Conv1D (128, 9, padding = "same", activation = 'relu'))

model.add (LSTM (30, return_sequences = True, batch_input_shape = (BATCHSIZE, INTERVAL, 1)))
model.add (LSTM (40, return_sequences = True))
model.add (LSTM (50))
model.add (Dense (1, activation = 'sigmoid'))
model.compile (loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

Experiments (cont.):

After 20 epochs, the Training and Validation accuracy have already achieved a more than 93% of accuracy.

The model and weights are then saved and transfered to Raspberry Pi.



The Performance of the Anomaly Detections

Infer the TRAINING data:

982 us/step

Accuracy: 95%

Infer the TESTING data:

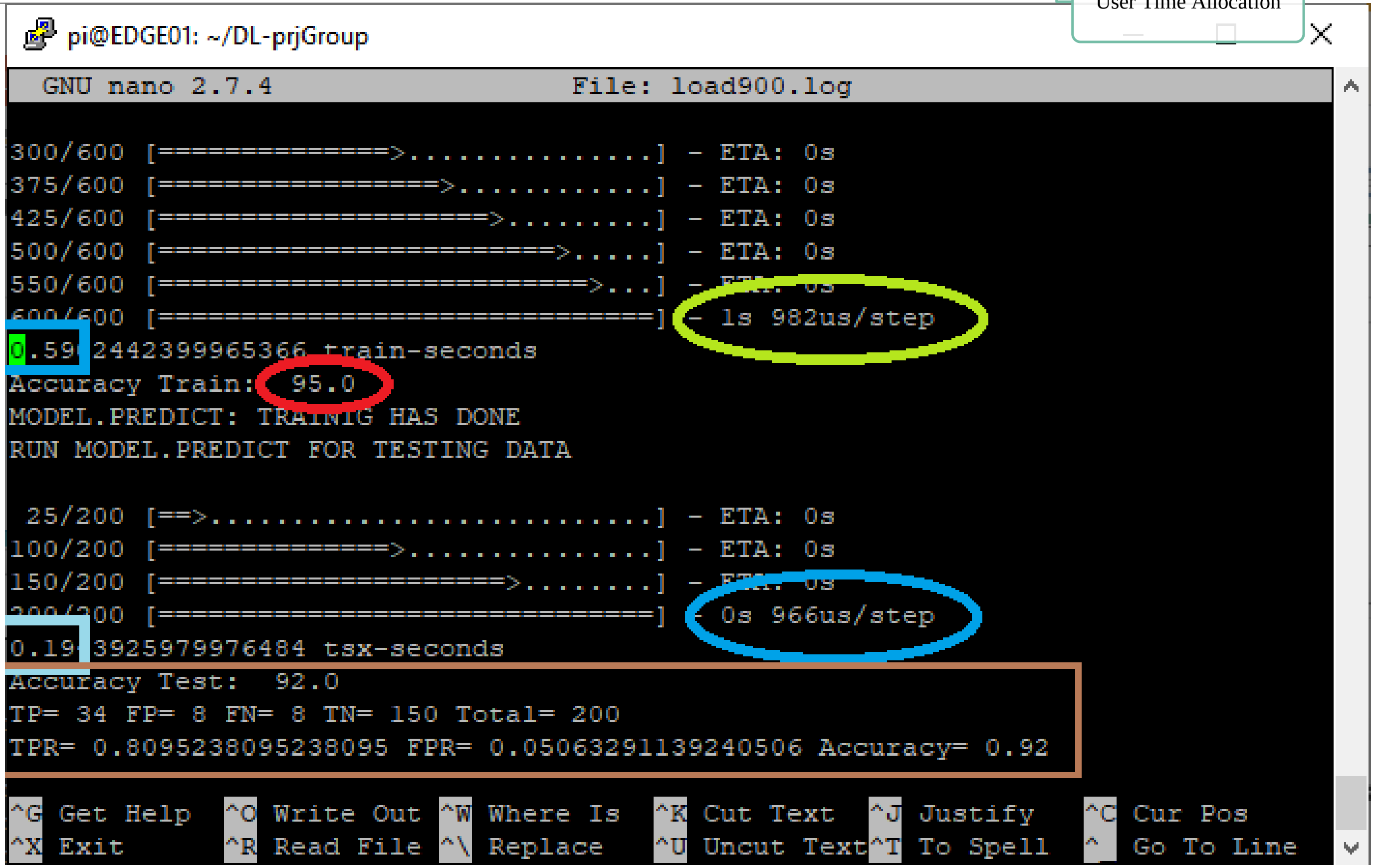
966 us/step

Accuracy: 92%

TP = 34, FP=8

FN=8, TN=150

TPR=81%, FPR= 5%



Experiments (cont.)

- Testing Result Summary :

Time steps	TP	FP	TN	FN	TPR	FPR	Accuracy
200	34	8	150	8	**81%**	5%	**92%**

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

$$Validation\ Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

CPU Time Allocations

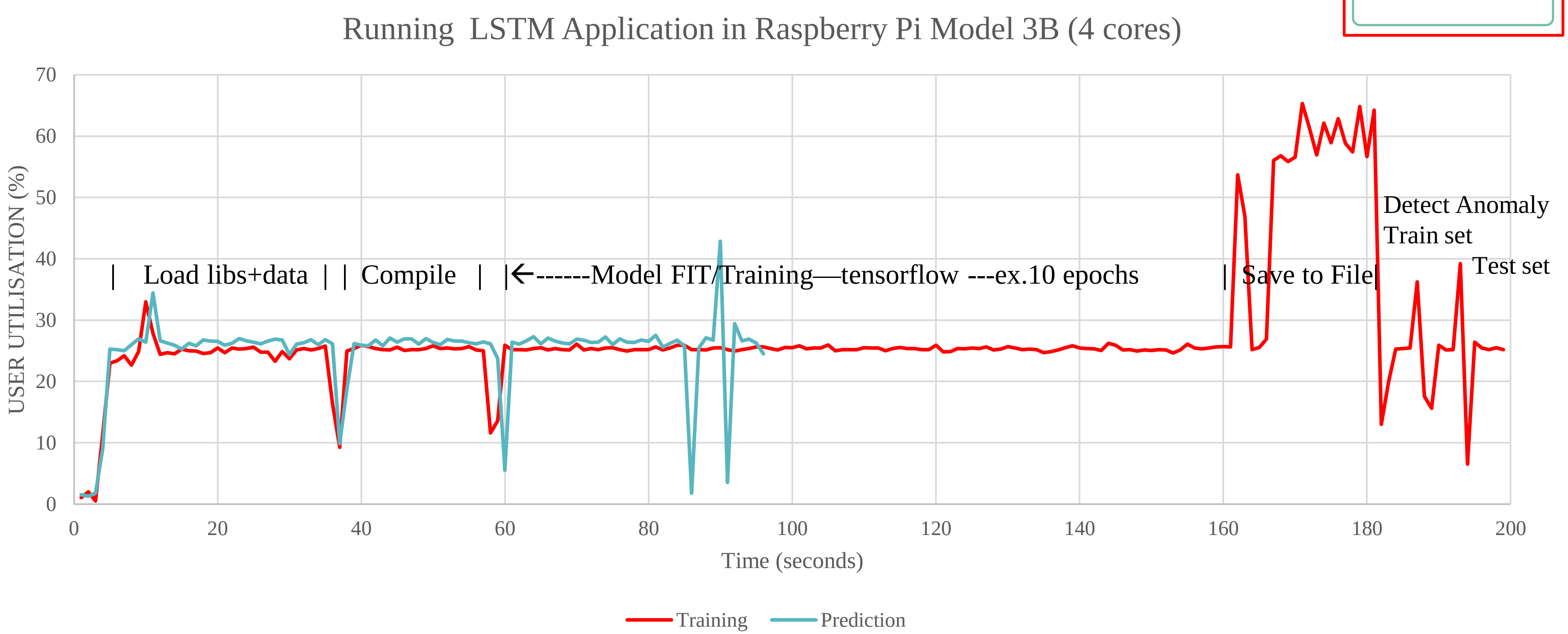
To capture the %user time allocation during run-time, systat[7] is installed and system activity reporter (sar) command is used here.

Example output of sar

Time	%user	%system	%iowait	%idle
13:50:09	0	0.25	0	99.75
13:50:10	0.25	0.25	0	99.50
13:50:11	12.12	1.01	0	86.87
13:50:12	22.92	2.52	0	74.56
13:50:13	24.69	0.76	0	74.56
13:50:14	24.24	1.26	0	74.49

[] http://sebastien.godard.pagesperso-orange.fr/tutorial.html

User Time Allocations



Getting the allocation time

```
from time import sleep, perf_counter as pc
t0 = pc() ; fx(a);  print( pc() - t0 ,"seconds")
```

Startup time before inferencing : 53.8 s

No	Job	Time
1	Load Library (keras, tensorflow, etc)	22.7 seconds
2	Upload dataset from MMC to Main Memory, Normalisation, Split Train & Test.	7.7 seconds
3	Compile the Model (optional)	19.6 seconds
4	Load the Weight.h5 & model.json (for Inference only)	23.4 seconds
5a	Detect Anomaly of Training Data (load the model.predict) (600 timesteps)	6 ms/step 3.8 seconds
5b	Detect Anomaly of Training Data (600 timesteps)	910 us/step 0.55 seconds
6	Detect Anomaly of Testing Data (200 timesteps)	947us/step 0.19 seconds

Outline

- Introduction
- Anomaly Detection System Design
  - Data Modeling
  - Anomaly Detection Modeling
- Experiments
- Conclusions and Future Work

Conclusions and Future Work

- We proposed an Anomaly Detection at IoT Edge using Deep Learning that can detect the anomaly in a single board computer system using Convolution and LSTM.
- Experiments show that the accuracy, TPR, and FPR of this system are 92%, 81%, and 5%, repectively.
- Detection for each timestep can be done in less than 1 ms.

Conclusions and Future Work (cont.)

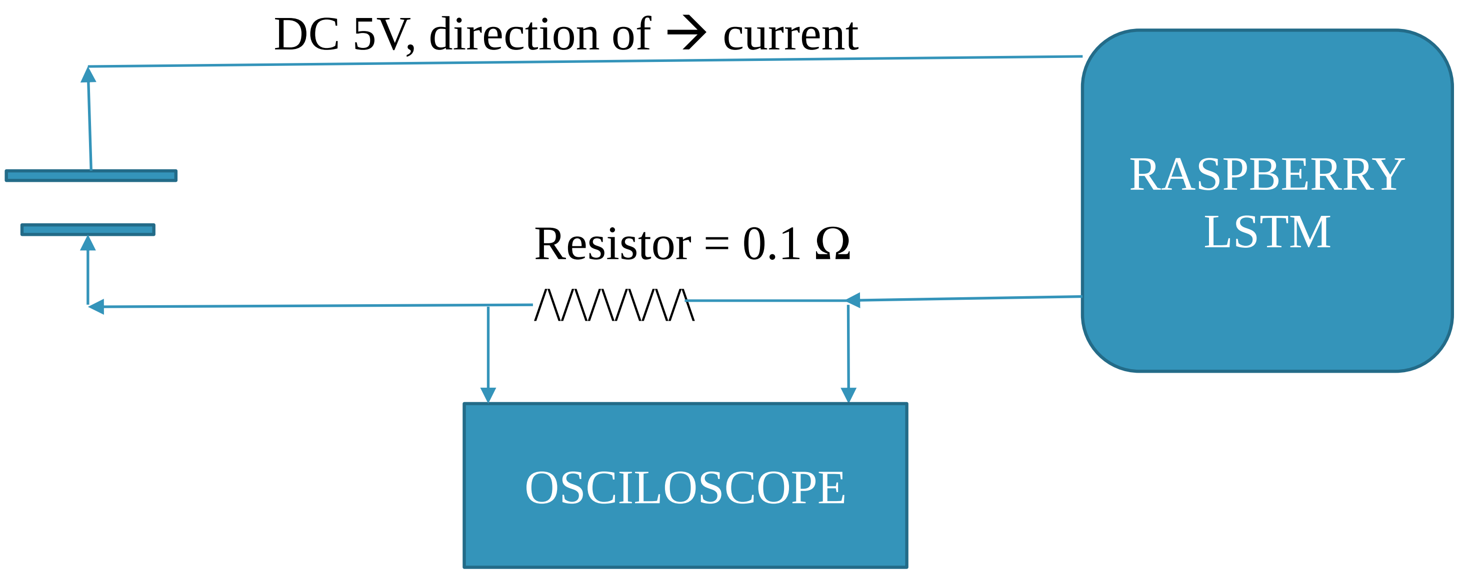
- We expect to increase the adaptability of the system by expanding the range of multi smart meters data samples.

- We will also focus on longer prediction times to detect electricity use anomalies.

Thank you:

1. We thank you for the opportunity to present this manuscript.
2. We also express our gratitude for constructive feedbacks from reviewers.
3. In addition, we also thank you for the positive questions and constructive feedback from the attendees.

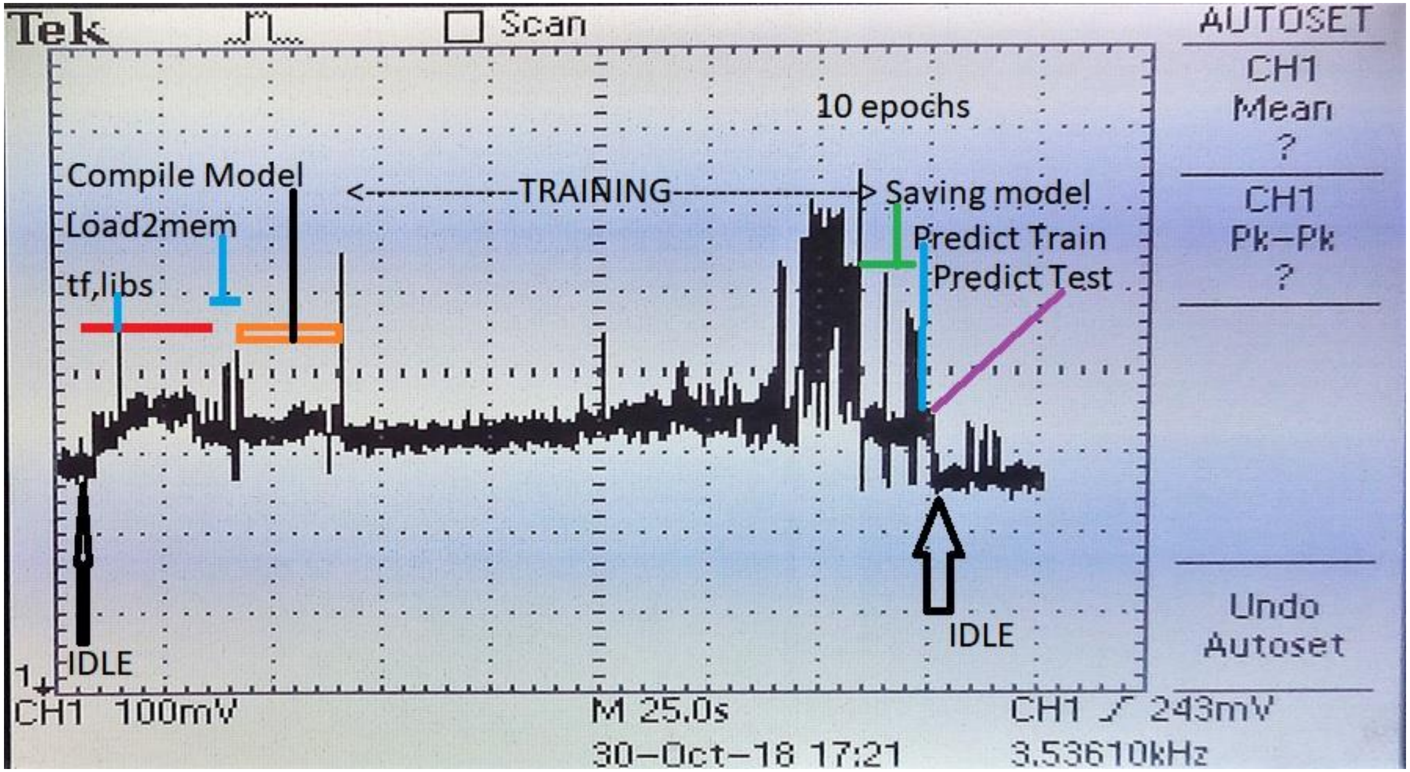
Measure the DC Current



Use Digital Storage Osciloscope (DSO) with time/div 25s, volt/div 100mV.

- Capture the voltage when in LSTM implement train and when using pretrained

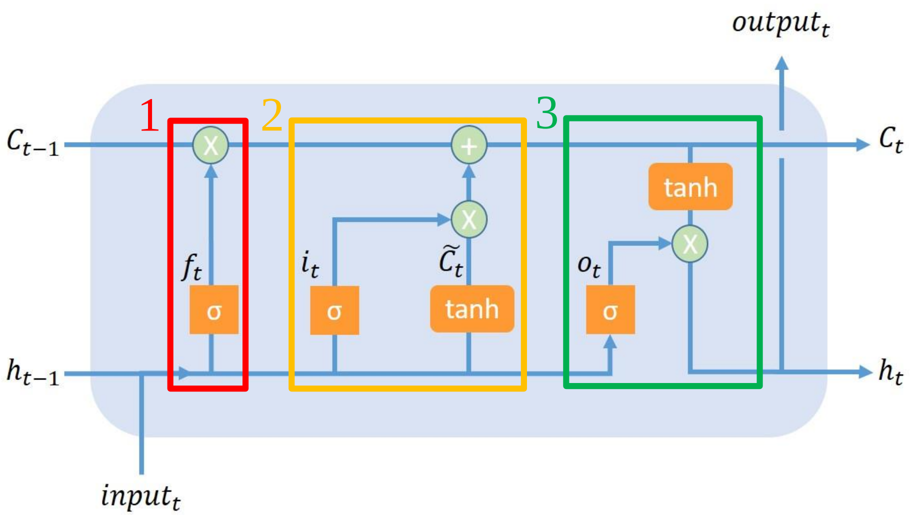
Getting Data from Current flow



500 mW/ vertical Step and 25 seconds/ horizontal step

Detection Module

- 1 block/timestamp LSTM designed prediction model [6]



$$i_t = \sigma(W_i x_t + W_i h_{t-1} + b_i)$$
$$f_t = \sigma\left(W_f x_t + W_f h_{t-1} + b_f\right)$$
$$c_t = f_t * c_{t-1} + i_t * tanh(W_c x_t + W_c h_{t-1} + b_c)$$
$$o_t = \sigma(W_o x_t + W_o h_{t-1} + b_o)$$
$$h_t = o_t * tanh(c_t)$$

i_t: t_{th} input gate of LSTM
f_t: t_{th} forget gate of LSTM
o_t: t_{th} output gate of LSTM
W: weight matrix
b: bias vectors
x_t: current input data
h_{t-1}: previous hidden state output
h_t: current hidden state output
c_t: cell state
σ: sigmoid control layer, output value between 0 to 1
$tanh$: tanh control layer, output value between -1 to 1