Anomaly Detection at the IoT Edge using Deep Learning

Darmawan Utomo and Pao-Ann Hsiung[†]
Department of Computer Science and Information Engineering,
National Chung Cheng University, Taiwan, R.O.C.
E-mail: [†]pahsiung@cs.ccu.edu.tw

***Abstract*—Irregularities in the use of electrical energy could result in failure of power grids and blackouts. Anomaly detection is required not only for ensuring grid safety, but also to prevent illegal hacking. Long-term data are recorded for such anomaly detection. However, due to the nonlinear characteristics of the time series data, correlation regression becomes difficult when using non-deep learning techniques. To address this issue, we use Long-Short Term Memory (LSTM) recurrency techniques. We experiment with two datasets from two different smart meters, which are sampled once every 30 seconds for one month. A total of 120,000 data samples were used for training and 40,000 data samples for testing. From the experiment results, the testing accuracy, True-Positive Rate, and False Positive Rate were 0.92, 0.81, and 0.50, respectively. Further, to demonstrate that the LSTM model can actually be designed at the network edge, we implemented the model and the trained weights on a Raspberry Pi platform. The inference time for each sample was 935 μs, which is short enough for realizing edge-based anomaly detection.**

***Index Terms*—Anomaly, Deep Learning, LSTM, Raspberry Pi, Edge-based Prediction.**

I. Introduction

In the traditional cloud-based Internet-of-Things (IoT) architecture design, all data must be sent to the cloud resulting in unpredictable delays, privacy concerns, and non real-time characteristics. It is often found in IoT systems that if network edge devices (gateways, nodes, etc.) could be made smart enough to process data adequately for some purposes such as anomaly detection, then computing could be more real-time and efficient. Taking smart meters as example, we demonstrate in this work how network edge devices can be made smart by implementing deep neural network models such as Long Short Term Memory (LSTM) [1] at the edge and evaluating its efficiency on a simple Raspberry Pi platform.

In Advanced Metering Infrastructures (AMIs) smart meters are a temporary place for collecting, computing, and storing data. Due to the sheer number of smart meters in any city region, AMI cannot cope with processing all the data in the cloud. Thus, several an edge-based model is proposed for smart grid with smart meters [2]. Smart meters have several advantages for both producers and consumers of electricity, such as the reduction of reader staff in place, increasing system reliability and quality, reducing the waste of energy, gas, and water, leak and theft detection [3]. Andrysiak et al. proposed a statistics-based anomaly detector for predicting smart meter data outliers [4]. To address the same issue using a novel deep learning approach, this work proposes an IoT edge device design that uses LSTM to detect anomalies in smart meter data.

II. Related work

In [5], waste energy in building operations is investigated by utilizing normal patterns and abnormal patterns. Datasets from normal patterns are obtained from real-world datasets from Powersmiths, Canada. While abnormal data is made artificially. Each timestep data is processed in parallell by several n-Learner Models, n-Threshold Determinators, and n-Anomaly Classifiers that aim to produce optimal positive and negative error values to find the most optimal Threshold value. The output of this block are values above and below the threshold that are considered as anomalous and normal. Then by using majority voting, the occurence of anomaly or normal is determined.

In [5] there is no information regarding hardware record, processing time, and amount of data used. Abnormal data is synthesized data so there is a possibility of emergence of incompatibility with the real data. In addition, a large number (25) features are used, which would result in large memory consumption and high latency when implemented in an edge device such as on the Raspberry Pi. In contrast, in our work a technique is proposed to enable the realization of an anomaly detector within a network edge, using only one feature, and by using data from real datasets.

III. Anomaly Detection System Design

The design starts from processing datasets, optimizing hyper-parameter searches on the PC, and implementing them on the Raspberry Pi. Two datasets are arbitrarily selected from [6]. As shown in Figure 1, the dataset is then normalized to a real value between 0 and 1. The characteristics of the dataset and the data sharing used can be seen in Figure 2. This dataset is then traced based on the model design and produces model output and weights. The output is then exported to Raspberry Pi which already has Python, Tensorflow, and Keras installed. It is then implemented as PC-Python code, which is executed with the training and testing datasets to evaluate the accuracy level, TPR, FPR, and processing times.

For the proposed anomaly detector, two real datasets from Non-Intrusive Load Monitoring [6] are used. Each of these datasets consists of 86400 timestamps of energy data obtained by sampling once every 30 s for one month. To adjust batch and timesteps, only 80000 data samples were taken from each dataset. Figure 2 displays the data samples that are considered normal and abnormal, during both the training and testing phases.

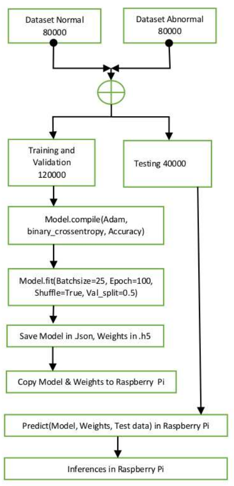
Dataset Normal 80000
Dataset Abnormal 80000
Training and Validation 120000
Testing 40000
Model.compile(Adam, binary_crossentropy, Accuracy)
Model.fit(Batchsize=25, Epoch=100, Shuffle=True, Val_split=0.5)
Save Model in Json, Weights in .h5
Copy Model & Weights to Raspberry Pi
Predict(Model, Weights, Test data) in Raspberry Pi
Inferences in Raspberry Pi

Fig. 1. Training and testing phases following the Normal and abnormal dataset.

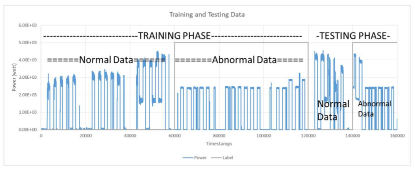

Fig. 2. Data characteristics and Dividing phases.

The model used is a combination of Convolution1D followed by MaxPooling1D before being processed with LSTM. The purpose of Conv1D and MaxPooling1D is to capture important information from a timesteps block. While the three layers of LSTM applied are useful for capturing better generalizations than if you use only one layer. Each timesteps consists of 200 timestamps. The training process uses Batchsize of 25 and is repeated 100 times. The data used before training is data that has been shuffled before. The Keras framework is used to implement this model design [7].

IV. Experiments

From a total of 160,000 normal and abnormal combined data samples, 120,000 training data samples were selected, where normal and abnormal data were of 60,000 samples each. The amount of data for training and validation is 60,000 data each. The remaining 40,000 data samples are used for testing.

The accuracy of training, validation, and testing are 0.986, 0.973, 0.920, respectively, with no overfitting. Of the 200 test data samples, each consists of 100 anomaly and 100 normal samples. 184 samples were predicted correctly, while only 16 predictions were incorrect. The True Positive Rate (TPR) and the False Positive Rate (FPR) are 0.81 and 0.50, respectively, where True Positive = 34, False Positive = 8, True Negative = 8, False Negative = 150. TPR of 0.81 represents 34 positive results occurring among 42 positive samples.

These models and weights were then copied to the installed Raspberry Pi, Python, Tensorflow, and Keras. By testing the same data, Raspberry Pi takes 935 μs to input each timestep or about 1,069 inferences per second. The prediction results show the same quality as on a PC. We conclude that it is very possible to implement the DL-LSTM module in a network edge device, which is Raspberry Pi here.

Raspberry Pi-3B has four cores which when running on average only uses one core or 25% of CPU usage. When loading the library there are two cores used. The timer application is used to record time interval from several important parts relevant to the Figure 1. Loading datasets with normalization consumes 7.81 s. Compiling the model utilized 19.69 s. Loading model.json and weights.h5 consumes 23.40 s. When running for the first time, this program used 3.78 s for 600 samples or around 6 ms/step because of loading the dynamic library from the MMC to main memory. Running predictions for successively, time was reduced to 0.56 s or around 935 μs. The accuracy test result evaluated from 200 samples was as high as 92%.

V. Conclusions

From the example of two smart meter datasets, the ability to distinguish anomalous data from the normal data can be obtained with a high accuracy, TPR and FPR of 92%, 0.81, and 0.50, respectively. In a network edge device such as Raspberry Pi, each prediction inference took 935 μs. The storage of models and weights consume large memory spaces. In the future, a new technique to reduce the memory usage will be required.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[2] F. Y. Okay and S. Ozdemir, "A fog computing based smart grid model," in *2016 International Symposium on Networks, Computers and Communications, ISNCC 2016, Yasmine Hammamet, Tunisia, May 11-13, 2016*, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISNCC.2016.7746062

[3] J. Lloret, J. Tomás, A. Canovas, and L. Parra, "An integrated iot architecture for smart metering," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 50–57, 2016. [Online]. Available: https://doi.org/10.1109/MCOM.2016.1600647CM

[4] T. Andrysiak, L. Saganowski, and P. Kiedrowski, "Anomaly detection in smart metering infrastructure with the use of time series analysis," *J. Sensors*, vol. 2017, pp. 8 782 131:1–8 782 131:15, 2017. [Online]. Available: https://doi.org/10.1155/2017/8782131

[5] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778817306904

[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. [Online]. Available: http://arxiv.org/abs/1408.6595

[7] https://keras.io/getting-started/sequential-model-guide/.