



ESDA: An Energy-Saving Data Analytics Fog Service Platform

Tiehua Zhang¹(✉), Zhishu Shen², Jiong Jin¹, Atsushi Tagami², Xi Zheng³,
and Yun Yang¹

¹ School of Software and Electrical Engineering, Swinburne University of Technology,
Melbourne, Australia

{[tiehuazhang](mailto:tiehuazhang@swin.edu.au), [jiongjin](mailto:jiongjin@swin.edu.au), [yyang](mailto:yyang@swin.edu.au)}@swin.edu.au

² KDDI Research, Inc., Fujimino-shi, Japan

{[shen](mailto:shen@kddi-research.jp), [tagami](mailto:tagami@kddi-research.jp)}@kddi-research.jp

³ Department of Computing, Macquarie University, Sydney, Australia
james.zheng@mq.edu.au

Abstract. The volume of heterogeneous data collected through a variety of sensors is growing exponentially. With the increasing popularity of providing real-time data analytics services at the edge of the network, the process of harvesting and analysing sensor data is thus an inevitable part of enhancing the service experience for users. In this paper, we propose a fog-empowered data analytics service platform to overcome the frequent sensor data loss issue through a novel deep autoencoder model while keeping the minimum energy usage of the managed sensors at the same time. The platform incorporates several algorithms with the purpose of training the individual local fog model, saving the overall energy consumption, as well as operating the service process. Compared with other state-of-the-art techniques for handling missing sensor data, our platform specialises in finding the underlying relationship among temporal sensor data series and hence provides more accurate results on heterogeneous data types. Owing to the superior inference capability, the platform enables the fog nodes to perform real-time data analytics service and respond to such service request promptly. Furthermore, the effectiveness of the proposed platform is verified through the real-world indoor deployment along with extensive experiments.

Keywords: Fog computing · Service-oriented networking · Deep autoencoder · Energy-saving algorithm

1 Introduction

The rapid development of the Internet of Things (IoT) technologies has unleashed the immense potential of large-scale data analytics. In the meantime, the deluge of IoT data produced by these interconnected IoT objects becomes one of the critical enablers for enhancing the intelligent human living environments [17]. The ultimate goal of aiding ambient living experience and improving

quality of life continues to drive the success in the intersected field of IoT and service-oriented computing.

As the core component of IoT that are generally located at the edge of the network, various sensors take most of the credits in terms of forming the intelligent living environment. These massively heterogeneous, dynamic sensor data are facilitating a myriad of IoT applications in order to offer real-time, context-aware, and highly personalised service. Moreover, there is no doubt that in conjunction with the prosperity of IoT technology, the number of sensors is growing exponentially as well.

In the act of being an integral part of the IoT network, heterogeneous sensors always play an indispensable role in some prevailing IoT implementations including smart city, agriculture and building, etc. [6]. However, the amount of energy consumed by a large number of sensors is not negligible and thus attracts the attention from both academia and industry to come up with different energy conserving algorithms and frameworks to deal with this long-standing problem, yet converging to the common goal - to improve the sustainability of the IoT network involved heavily with sensors [5, 13, 18]. Apart from that, data loss is inevitable in IoT network due to sensors' inherent characteristics such as malfunction or battery exhaustion. This phenomenon severely compromises the quality of service (QoS) in some data-driven applications and poses a big challenge [1].

Most of the IoT applications are in favour of providing the data analytics service to some extent, taking advantage of the mature cloud computing is thus a preferable option. However, as the most commonly used approach nowadays, the colossal amount of data collected in the IoT network is simply transferred to the cloud servers for further processing and analysing, which has severe side effects on network performance and further causes communication overhead and network congestion. Also, the majority of IoT applications deployed on the cloud starts to concern the QoS being returned to the interested users, in which the service transmission latency contributes a significant part in service consumers' perception with regard to the overall performance of the service invocation. Apart from that, the arising challenges like cloud energy waste and data privacy issues also suggest that the reliance on traditional IoT-Cloud schema alone to provide various IoT services is no longer effective and efficient, and an alternative computing paradigm that could seal the gap between IoT devices and cloud to provide better quality of service is needed [6].

Fog computing is emerged under this circumstance to complement the inadequacy of the IoT-Cloud schema. The idea of extending the cloud to the edge of network closer to end users is considered as an alternative with the overarching goal of "off-loading" from the cloud where fog, as the proxy, could be equipped with not only computation power, but also storage and networking resources to accommodate various IoT applications. Most importantly, IoT applications could be deployed in fog rather than the conventional approach on either resource-constrained IoT devices or the remote cloud. In this regard, fog and cloud complement each other and encompass other IoT devices to form a

seamless cloud-fog-things service continuum in which service could be disseminated [19].

Machine learning technologies, especially deep learning, have celebrated massive successes in domains of IoT and service-orientated computing owing to their adoptions in a wide range of IoT applications. The deployment of ubiquitous sensors has played a crucial role in IoT infrastructure to empower various real-time data analytic services such as health care monitoring, intrusive detection and smart building, etc. Applications provisioning such services require to collect, process, analyse and communicate enormous amount of sensor data consistently in order to provide the highly customisable services related to the sensed phenomena. However, these data-thirsty tasks are often plagued by the missing sensor data issue, which consequently compromises the performance of any learning algorithms [10]. While most of the data complementing algorithms and frameworks commonly fail to interpret underlying temporal data correlations and underestimate the high complexity of predicting multi-type sensor data [3], it is imperative to discover a powerful approach that could not only cope with the missing data but also respond to such service request promptly.

In this paper, we present a novel energy-saving data analytics fog service platform, namely ESDA, which serves the purpose of providing real-time, multi-type and large-scale data analytics services for IoT devices. More specifically, under the control of fog nodes, sensors no longer need to be constantly turned on to transmit sensed data upwards. Fog, instead, will initiate the in-fog learning process and use the trained local model to predict missed multi-type sensor data that are substantial for many service requesters. More importantly, the platform will automate the process of conserving energy cost of the platform by turning sensors into the sleep mode during the service operation.

The main contributions of this paper are as follows:

1. To better serve real-time services in IoT context, a fog-empowered energy saving data analytics platform is proposed. The fog can utilise dedicated computation and storage resources to facilitate real-time data analytics services. Besides, the platform offers a flexible deployment options through the selection of different energy saving patterns.
2. Taking advantage of the recurrent neural network (RNN), a novel fog-based deep encoder architecture is proposed to improve the accuracy of multi-type sensor data prediction, which takes the internal time-series correlation into account to enhance the inference accuracy.
3. An in-fog learning and predicting algorithm is designed and run at the fog layer, where each fog node will train the local model and ensure an acceptable data prediction accuracy to satisfy the need of data analytics services. Apart from that, an energy conserving algorithm is incorporated to minimise the overall energy consumption of the platform during the service operation.
4. Advantages of adopting the proposed platform are evaluated experimentally through a real-world indoor deployment, along with its effectiveness and applicability for many use cases in IoT environment empowered by fog computing.

The remainder of the paper is organised as follows. We review the related work on sensor data prediction and acquisition, deep learning based real-time data analytics and energy-saving approach in Sect. 2. In Sect. 3, we introduce the proposed fog service platform architecture followed by the detailed explanation of fog-enabled deep autoencoder (FEDA) and integrated algorithms. We then present the deployment of our real-world indoor testbed as well as the comprehensive experiments conducted in Sect. 4, and we draw the conclusion and point out the future work in Sect. 5.

2 Related Work

There are several previous efforts made towards developing methods in complementing missing sensor values and saving energy consumption in the IoT environment to fulfil different service needs.

The approaches to fill the incomplete data are actively studied in the area of image processing. The authors in [12] presented an unsupervised visual feature learning algorithm for image inpainting. By analogy with auto-encoders, this algorithm utilises convolutional neural networks (CNN) to generate the incomplete region of another arbitrary image based on the surrounding context of the incomplete part. In [8], the authors proposed a denoising autoencoder named Multimodal Autoencoder (MMAE) to handle the missing multimodal data. This method is an unsupervised learning technique using the deep neural network (DNN). The empirical evaluation verified that the MMAE could outperform the traditional principal components analysis (PCA) in prediction accuracy of the feature values from multiple missing modalities in the scale between 0 and 1.

Satisfactory performance of data prediction can eliminate the demand to periodically sending the original raw data to the cloud. The sensors whose data can be accurately predicted are permitted to sleep as long as possible during the network operation. Conceptually, the longer sleep time window that sensors are allowed to have, the more energy can be saved for the network. The critical task is how to precisely predict these incomplete data from sleeping or malfunctioning sensors [2]. For this reason, recently there are several works focusing on data prediction using machine learning for the IoT network: the authors in [11] developed a data prediction algorithm with an error-correction learning scheme. This algorithm is developed from recursive least squares (RSL) and is used to improve the value which is initially predicted by using a small number of data. The authors in [13] proposed a derivative-based prediction that uses a linear model to observe the trends of data in recently captured data to predict the future data. By analysing the latest data, this approach can produce a satisfactory model for predicting data in the short-term. However, frequent updates on trainable parameters of the model are required for holding this high accuracy in predicting long-term data trends. The authors in [15] introduced CNN to learn the correlation of the neighbouring sensor data to decide the sensors active/sleep status, i.e., the sensors whose data can be accurately predicted will be turned into the sleep mode in priority.

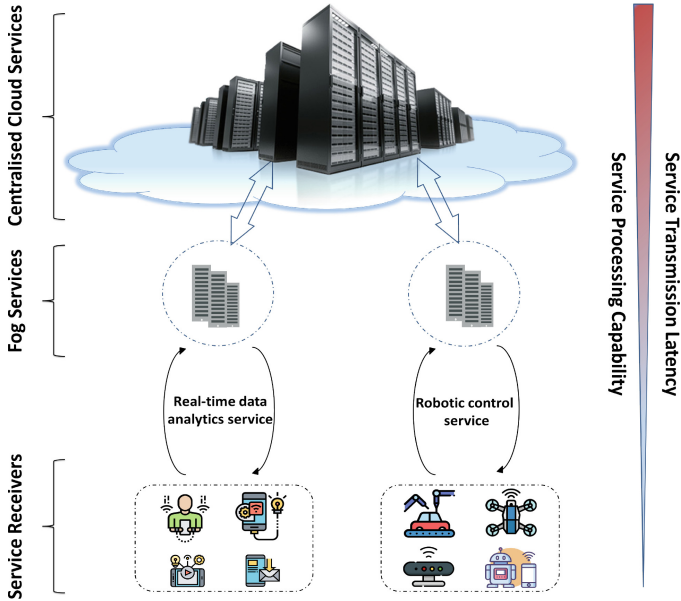


Fig. 1. Fog service architecture

However, with the rapid increase of the number of devices in the IoT environment, the heavy traffic volume generated by these sensors increases the processing burden on the cloud. Moreover, the expected connection latency leads to a degradation of the expectation of real-time data service. A potential solution is to place the service providers closer to end devices/users' side for the timely service provision process and reduce the redundant data communication to the cloud. For instance, the work [14] proved that for IoT environment like building energy management system, by utilising a collaborative fog platform for delivering data processing service in a real-time manner, a significant reduction of traffic volume to the cloud could be realised. Some works shift their attention to using resource-constrained IoT devices to fulfil the allegedly computation-intensive mission. For instance, the authors in [9] developed a light-weight, tree-based classification algorithm for data prediction on IoT devices. This algorithm learns a single, shallow, spare tree to reduce the required model size while forming a small number of prototypes data representing the entire data sets. This algorithm achieves high prediction accuracy while it can be executed on resource-constrained IoT devices, i.e., Arduino Uno board. Similarly, to reduce the data learning complexity, the authors in [4] proposed a k-nearest neighbour (k-NN)-based algorithm by using a spare projection matrix that projects all data sets into lower-dimension. One common issue for these two algorithms is that the model trained on a resource-scarce device is merely meant for the individual device, thus not suitable for providing data analytics service at a reasonable scale.

To the best of the authors' knowledge, this is the first work to propose a novel fog service platform to achieve prominent real-time data analytics in an energy-saving style.

3 Overview of the ESDA Fog Service Platform

A multi-tier fog service architecture can be observed in Fig. 1. Under this structure, end things/users act as the service requester/receivers, and fog nodes, on the other hand, are equipped with the computational and storage capability to enable a variety of real-time IoT services that can be activated on demand and delivered to the destination in a timely manner. In the service-oriented computing perspective, fog thus performs the role of the service provider. Besides, as the traditional service provider in IoT-Cloud schema, the cloud could still be the placeholder for a particular category of applications providing delay tolerant services, e.g., large-scale data backup service. This figure also well demonstrates what a hierarchical service continuum composed of IoT users/devices, fog and cloud visually looks like in the IoT network.

Since the service transmission latency drops with the shortening of the geographical distance between deployed service providers and end things/users (service requester/receiver), fog node could be placed in the vicinity of the raw data source in any autonomous environment. In a smart building scenario, multi-type sensors could interact directly with fog node in that area and feed raw data to the analytical applications running at the fog layer. Fog, at the same time, continually monitors the energy consumption status of each sensor to which it is connected, and utilises the energy-saving algorithm to put sensors to the sleep mode to extend their lifespan.

In the proposed ESDA fog service platform, a novel fog-enabled deep autoencoder model (FEDA), built on top of the long short term memory (LSTM)-based sequence-to-sequence structure [16], is adopted as the core part of the data analytics application to satisfy the real-time data service enquiries. The name "fog-enabled" comes from the fact that the whole training and inference process rely heavily on the support of the local fog nodes. Also, this platform fills in the gap concerning the missing sensor data when sensors are put into the sleep mode, recharging or malfunction. Afterwards, fog could respond to relevant IoT devices that depend on the returned service results to trigger corresponding actions, such as temperature change by air conditioner or luminous intensity adjustment by light controller.

Other than providing the reliable data analytics service, the platform also aims to accomplish the sustainability by taking an efficient energy utilisation. Both the in-fog learning algorithm for training the FEDA and the energy-saving algorithm are incorporated into the platform and explained in the following subsections.

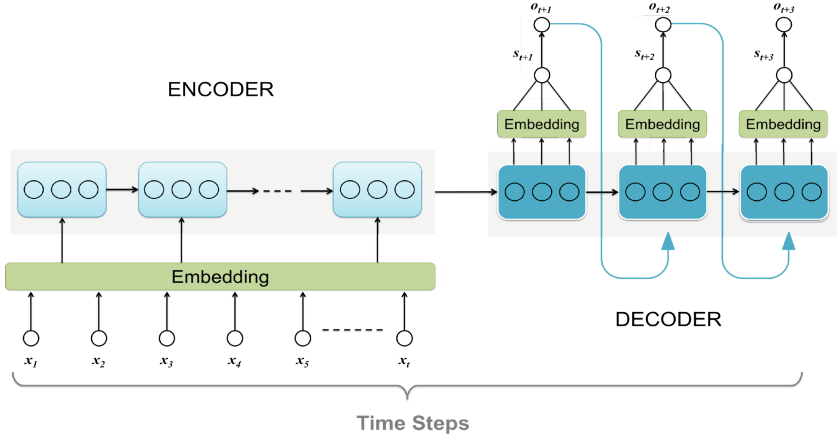


Fig. 2. FEDA model structure

3.1 Fog-Enabled Deep Autoencoder (FEDA)

An autoencoder is generally referred to as an unsupervised learning approach where a deep neural network tries to reconstruct an input X after the reconstruction error between the ground truth X and network output's X' is minimised through an optimizer. In this circumstance, the embedding layer is often presented in the compressed format and used as the compact representation of the data. One limitation of the traditional autoencoder is that the length of X' tends to be the same as X , which is not ideal for predicting the temporal data series that could end up being arbitrary length. Henceforth, we tend to leverage the LSTM-based sequence-to-sequence architecture as the fundamental part of our deep autoencoder owing to the capability of handling variable-length time series input and output another variable-length time series data.

The LSTM originally proposed in [7], enable several state variables, including hidden/control state h_t and memory cell state c_t to sustain the underlying correlation among temporal data, where t means a certain timestamp. A single LSTM unit's operations can be expressed as:

$$\begin{aligned}
 i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
 f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
 o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
 g_t &= \phi(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\
 c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
 h_t &= o_t \odot \phi(c_t)
 \end{aligned} \tag{1}$$

where σ is the sigmoidal non-linearity, ϕ is the tangent non-linearity and \odot is the element-wise dot product. The weights and biases metrics for computing each gate values are represented as $W_{i,j}$ and b_j , respectively.

Figure 2 presents the graphical representation of FEDA model structure. We apply LSTM in both encoder and decoder parts of the RNN, and the principal philosophy of this model is that, for both encoder and decoder, the value at x_{t+1} timestamp depends on previous state h_t and previous output value x_t to learn the underlying time series correlation. In order to encompass the flexibility required to predict the arbitrary missing sensor data given any previous time window, we convert the variable-length input data up until timestamp t into a fixed length embedding layer first, which afterwards will be fed into the encoding part on the left. The last state generated by the encoding phase is then passed to the right - the decoding phase, where the output starting from $t + 1$ is predicted for each forthcoming time step.

There are several points worth noting in the decoder part of the model. Firstly, since it could be multiple LSTM units (circles in each rectangle in Fig. 2) in each LSTM cell (rectangles in Fig. 2) throughout the network, in the decoding phase, the results generated by a number of LSTM units from a single LSTM cell, say n , is used as a mini-embedding vector whose size is equivalent to n . These mini-embedding vectors are then multiplied and added with their dedicated weight matrix W_{t+1} and bias b_{t+1} , respectively, to generate a single value (o_{t+1}) for that timestamp. Apart from that, the model generalises the coexistence of heterogeneous sensor data types, i.e., a multi-functional sensor can detect the value of temperature, humidity level, air pressure and luminous intensity at the same time in real world, and this sensor then produces four types of data repeatedly as the time elapses. To cope with that, each o_{t+1} is equipped with a trainable scalar variable s_{t+1} to scale the value of a specific data type up to its original magnitude as close as possible.

3.2 Integrated Algorithms

There are two main algorithms integrated into the platform designed with different purposes. The first one controls the process of fetching the preliminary trained model from the cloud based on historical backup data, communicating the controlled sensors to collect training data, and initiating the training process on each local fog node. The second one, on the other hand, focuses on the management of connected sensors per fog node, as well as carries out service operations from various service requesters. The details are broken down separately as Algorithms 1 and 2.

Algorithm 1 consists of two phases. Every fog node i ($i \in \mathcal{F}$) in phase one firstly checks the availability of local model $\mathcal{M}_{(i)}$, then it might use the preliminary pre-trained model from the cloud and restore it (lines 1–5 in Phase 1). As the preliminary model from the cloud is trained based on historical backup sensor data stored on the cloud, it does not reflect the real-time sensor reading in the environment and thus needs to be refined. The refinement process, in other words, can be addressed as the continuous in-fog training for the preliminary model. Each fog node collects sensor's reading in its managed area, then creates dataset \mathcal{D} (lines 1–6 in Phase 2). To train the FEDA model used by each fog node, \mathcal{D} needs to be split into two time series parts \mathcal{D}_s and \mathcal{D}_e randomly, where \mathcal{D}_s is

fed into the model as variable-length input mentioned in Sect. 3.1. In contrast, \mathcal{D}_p corresponds to the predicted data series, and the reconstruction error $\mathcal{E}_{(i)}$ is calculated between \mathcal{D}_p and \mathcal{D}_e using a distance error measurement (lines 7–9), e.g., Root Mean Square Error (RMSE) used here. It is worth noting the relation between error $\mathcal{E}_{(i)}$ and hyperparameter θ , where θ is a system-level parameter defined to balance the trade-off between data reconstruction volume and energy consumption amount. It could be explained as error tolerance threshold. Fog will stop training the local model until the error is no greater than threshold θ (lines 10–13). The pick-up strategy of the threshold value will be demonstrated in Sect. 4.2.

Algorithm 1. In-fog learning algorithm

Phase 1: *Preliminary Model Fetch*

- 1: fog f_i ($i \in \mathcal{F}$) fetches the preliminary model \mathcal{M} from the cloud
 - 2: **if** $\mathcal{M}_{(i)}$ not exists **then**
 - 3: $\mathcal{M}_{(i)} \leftarrow \mathcal{M}$
 - 4: **end if**
 - 5: restore local model $\mathcal{M}_{(i)}$
-

Phase 2: *Local Model Refinement*

- 1: $\mathcal{D} \leftarrow \emptyset$
 - 2: $\mathcal{S} \leftarrow$ sensors managed by fog f_i
 - 3: **repeat**
 - 4: **for** $\forall s \in \mathcal{S}$ **do**
 - 5: $\mathcal{D}.add(\text{current reading of } s)$
 - 6: **end for**
 - 7: Separate \mathcal{D} into two time series parts \mathcal{D}_s and \mathcal{D}_e
 - 8: $\mathcal{D}_p \leftarrow \mathcal{M}_{(i)}(\mathcal{D}_s)$
 - 9: $\mathcal{E}_{(i)} \leftarrow \sqrt{\frac{1}{n} \sum_{t=1}^n (\mathcal{D}_{p,n} - \mathcal{D}_{e,n})^2}$
 - 10: **if** $\mathcal{E}_{(i)} > \theta$ **then**
 - 11: Feed \mathcal{D} into $\mathcal{M}_{(i)}$ to continue training
 - 12: **end if**
 - 13: **until** $\mathcal{E}_{(i)} \leq \theta$
-

Pseudocode in Algorithm 2 concerns on inactivating managed sensors in a time period to reduce the overall energy consumption of the platform, as well as resolving data analytics service request. To start with, fog nodes are aware of the local model $\mathcal{M}_{(i)}$ trained from Algorithm 1 and system-level hyperparameter θ . Based on the selection of threshold θ , the platform is assigned with the corresponding energy-saving pattern P , which is used to calculate sensor's sleep time T . In addition, variables t_i^{start} and t_i^{end} are adopted as the indicator of the sleep window (lines 1–6). During the service operation process, fog would

receive periodic service request regarding the type and value of current sensor reading from different IoT equipments in every time interval t (lines 8–9). If a service request received falls into the sleep window of sensors, fog leverages $\mathcal{M}_{(i)}$ substantially to conduct the analytics service so that applicable results are able to be returned back regardless (lines 11 and 12). For example, inside a building, fog nodes may receive the temperature reading request from the air conditioner operating in the controlled area every minute to attain the thermal comfort. The sleep window needs to be re-calculated with the change of threshold θ along with energy-saving pattern P (line 17).

Algorithm 2. Energy-saving and service operating algorithm

```

1: Energy-Saving for  $\forall f_i \in \mathcal{F}$ 
2:    $\mathcal{M}_{(i)} \leftarrow$  local model trained by fog  $f_i$ 
3:    $P \leftarrow$  energy-saving pattern based on threshold value  $\theta$ 
4:    $\mathcal{S}_i \leftarrow$  sensors managed by fog  $f_i$ 
5:    $T \leftarrow$  sleep time calculated based on  $P$ 
6:    $(t_i^{start}, t_i^{end}) \leftarrow$  fog  $f_i$  de-activates  $\mathcal{S}_i$  for  $T$  and records the sleep window
7: Service Operation for  $\forall f_i \in \mathcal{F}$ 
8:   Repeat at every interval  $t$ 
9:      $\mathcal{R}_{(i)} \leftarrow$  service requests sent to fog  $f_i$  at current interval  $t$ 
10:    for  $\forall r \in \mathcal{R}_{(i)}$  do
11:      if  $t$  falls into window between  $(t_i^{start}, t_i^{end})$  then
12:         $f_i$  employs  $\mathcal{M}_{(i)}$  to conduct analytic service regarding  $r$ 
13:      else
14:         $f_i$  retrieves current sensor reading regarding  $r$ 
15:      end if
16:    end for
17:  Until  $\theta$  changes

```

4 Experiments

We present the real-world experimental settings in an indoor deployment as well as evaluation results in this section.

4.1 Experimental Setting

We introduce the building energy management system (BEMS) as the use case to evaluate our proposed platform. Figure 3 illustrates a deployment example of a BEMS application. The main objectives of BEMS include: 1. monitoring and managing the overall energy consumption; 2. constantly checking the indoor environment conditions (e.g., temperature, humidity) to ensure the comfort of occupants. Fog, as the placeholder of the application, acts as a real-time data analytics service provider to respond requests coming from various IoT devices.

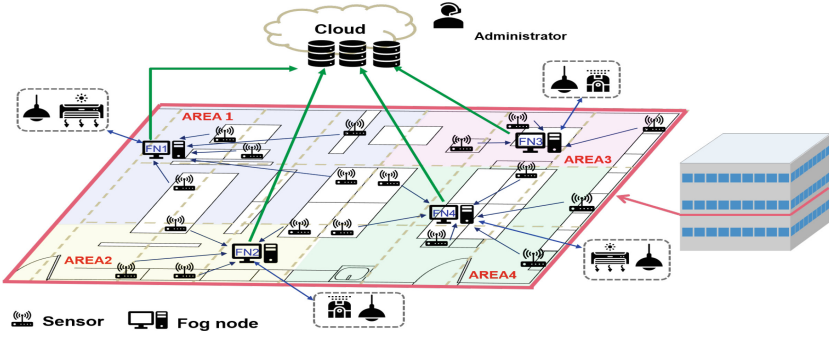


Fig. 3. Floor plan of real-world indoor deployment

Table 1. Setup parameters for experimental network

Parameters	Values
Sensor number	18
Network area size	162 m ²
Data collection interval	1 min
Data collection period	1 year
Data type	Temperature (Celsius), humidity (%), luminous intensity (Candela), barometric pressure (Pascal)

As mentioned earlier, the platform allows on setting of different energy-saving patterns aligned with the system-level parameter θ for versatile deployment scenarios, and such system-level parameters are controlled by system admin (demonstrated in Fig. 3). Apart from that, fog nodes remain connected to both sensors and other IoT devices such as humidifier, air conditioner and light, etc. to complete the service continuum. For example, in Fig. 3, as shown by blue arrows, fog nodes can send the analytics results straight to the air conditioning and lighting systems for adjusting the current status. It is worth mentioning that this real-world deployment happens inside an office building. The sensor (Texas Instruments CC2650 SensorTag) placement information is shown in Fig. 3, and the parameter details of the network topology are listed in Table 1.

4.2 Evaluation Results

It is of importance for applications such as BEMS deployed in the fog to provide real-time data analytics results as accurate as possible on which IoT service requester like air conditioner could faithfully rely and adjust the behaviour accordingly. We thus verify the effectiveness of the proposed platform through comprehensive experiments. To assess the performance of FEDA, three other state-of-the-art autoencoder variants are used as the benchmarks, including CNN encoder, DNN encoder as well as vanilla RNN. Two forms of comparisons are demonstrated here: real-world multi-type sensor data reading prediction at each

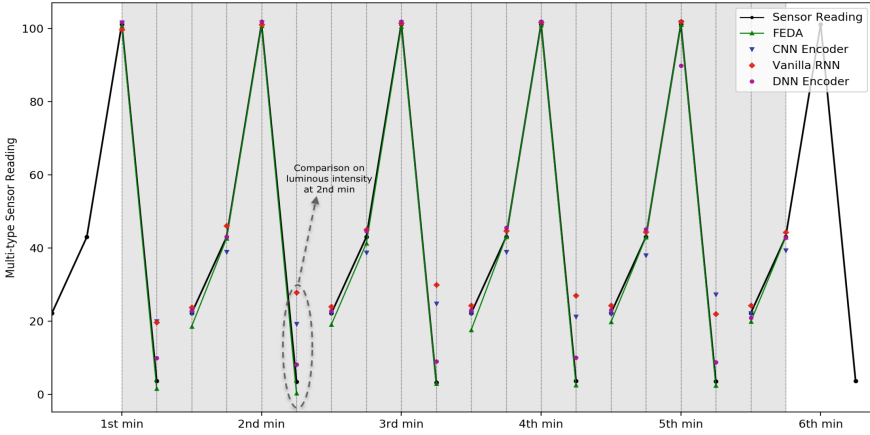


Fig. 4. Prediction on multi-type sensor reading (4 data types) (Color figure online)

time point, and the continuous single data type prediction in a given period. In addition, the experiment with regard to saving energy while retaining a satisfactory prediction result is demonstrated at last. In order to ensure a fair comparison, the Adam optimizer [8] is used across different learning techniques with the learning rate 0.001, which is observed to learn the fastest during the experiments.

Figure 4 displays the prediction results for multi-type sensor data reading at each time point. As stated in Table 1, each fog node collects managed sensor readings in 1 min interval for all four data types. To realise the test on the adopted learning model in terms of predicting heterogeneous data types all at once given a random length input, sensor reading in one time period has been randomly masked off indicating the loss of the data, and each model predicts on the lost heterogeneous data values, respectively. For each time interval (from 1 min to 6 min) in Fig. 4, sensors are set to read the surrounding phenomena in the order of temperature, humidity, barometric pressure, and then luminous intensity (four points connected into a line segment at every minute). The greyed-out area indicates the period when the mimic data loss happens (sensors in sleep or fault conditions), and the dashed-vertical line aligns with the prediction results for that data type generated among different models, e.g. dashed grey circle highlights on the prediction results on luminous intensity data type by different models at 2nd min. It could be observed that the green line with up-triangles markers (FEDA) generally have a closer distance to the ground truth sensor reading (black line with dots) on each data type at every time stamp compared with the other three.

Figure 5 amplifies the prediction results on each of the four data types in a 40 min time window, in which sensor data is mimicked to be missing from 10 to 30 mins. It is evident that for temperature, humidity and air pressure, the FEDA clearly outperforms other competitors, and even in the extreme case of luminous intensity where the value becomes 0 at some time point due to the office light

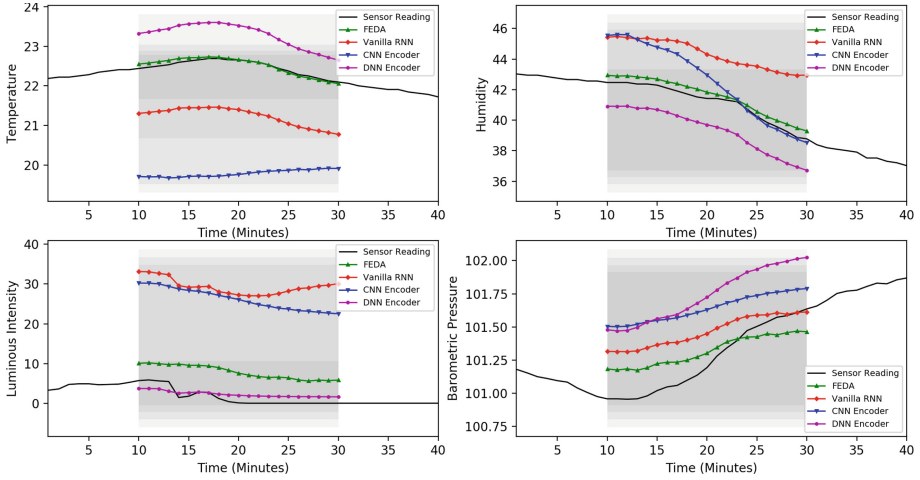


Fig. 5. Prediction on individual data type

being turned off at night, our model reserves a conservative prediction following the trend of the true sensor reading line. In general, both figures mentioned above verify that the novel FEDA model adopted in our proposed platform excels in delivering more accurate data analytics service from which the connected IoT service requesters could primarily benefit with better QoS.

The experiment result regarding the relation between the system-level error tolerance threshold θ and the total energy consumption in BEMS is shown in Fig. 6. The left y-axis in the graph represents the different values of θ after applying logarithm for better visualisation purpose, whereas the right y-axis tells the energy consumption by percentage (when energy consumption is 100%, all sensors are activated during the whole service operation period). The values of θ are derived from the calculated prediction errors using the test dataset. We conduct a grid search on θ and utilise four of them (errors calculated when 80%, 60%, 40% and 20% of total inactive times of all sensors, respectively) to form the corresponding energy-saving patterns (P1–P4), which is shown as the shared x-axis in the graph. It is as expected that the more sensors that are put into the sleep mode longer, the more reconstruction errors will be encountered, yet the less energy will be consumed for the whole system. We observe that the decrease of the θ is not strictly linear, and the slope between P2 and P3 is greater than other segments of the line. Henceforth, we adopt the threshold defined in P3 for both Algorithms 1 and 2, which empowers our platform to save energy consumption up to 40% while keeping an acceptable error tolerance. It is worth mentioning that the selection of patterns depends entirely on different use cases and deployment scenarios. For instance, if the platform is adopted in the smart agriculture use case where sensors and fog are placed in the geographically remote area, then the energy saving could come to the priority so as to extend the lifespan of energy-scarce sensors to a large extent, then the use of P1 might be more appropriate than others.

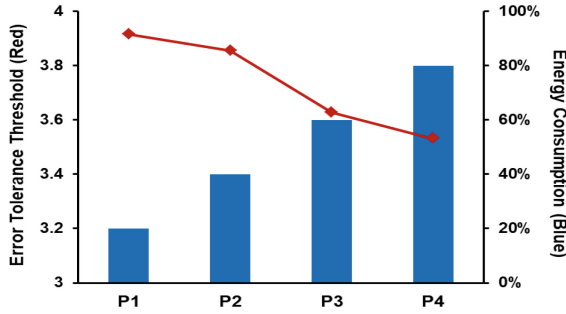


Fig. 6. Energy consumption and error tolerance threshold

5 Conclusion and Future Work

Our ESDA fog service platform is a low energy-cost IoT platform that provides real-time data analytics service and can be deployed in many scenarios. With the help of FEDA and in-fog learning, energy-saving algorithms, this platform could extend the lifespan of managed sensors without deteriorating much of the QoS thanks to the accurate data forecasting capability. We have deployed the platform into a real-world indoor IoT testbed and adopted BEMS to verify its effectiveness. The comprehensive experiment results demonstrate the superiority of the platform. The platform also enhances the flexibility by allowing to customise the error tolerance threshold θ to cater for different deployment requirements. Moving forward, we will investigate the possibility of using collaborative learning schema in this platform, where fog leverages each other's power to reach the consensus on the learning model.

Acknowledgements. This work is partly supported by Australian Government Research Training Program Scholarship, Australian Research Council Discovery Project Grant DP180100212 and NICT (Contract No. 19103), Japan.

References

1. Aldossary, S., Allen, W.: Data security, privacy, availability and integrity in cloud computing: Issues and current solutions. *Int. J. Adv. Comput. Sci. Appl.* **7**(4), 485–498 (2016)
2. Dias, G.M., Bellalta, B., Oechsner, S.: A survey about prediction-based data reduction in wireless sensor networks. *ACM Comput. Surv.* **49**(3), 58 (2016)
3. Gao, Z., Cheng, W., Qiu, X., Meng, L.: A missing sensor data estimation algorithm based on temporal and spatial correlation. *Int. J. Distrib. Sens. Netw.* **11**(10), 435391 (2015)
4. Gupta, C., et al.: ProtoNN: compressed and accurate kNN for resource-scarce devices. In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 1331–1340. *JMLR.org* (2017)
5. Harb, H., Makhoul, A., Laiymani, D., Jaber, A.: A distance-based data aggregation technique for periodic sensor networks. *ACM Trans. Sens. Netw.* **13**(4), 32 (2017)

6. He, J., Wei, J., Chen, K., Tang, Z., Zhou, Y., Zhang, Y.: Multitier fog computing with large-scale IoT data analytics for smart cities. *IEEE Internet of Things J.* **5**(2), 677–686 (2018)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
8. Jaques, N., Taylor, S., Sano, A., Picard, R.: Multimodal autoencoder: a deep learning approach to filling in missing sensor data and enabling better mood prediction. In: *Proceedings of the 7th International Conference on Affective Computing and Intelligent Interaction*, pp. 202–208. IEEE (2017)
9. Kumar, A., Goyal, S., Varma, M.: Resource-efficient machine learning in 2 KB RAM for the Internet of Things. In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 1935–1944. JMLR.org (2017)
10. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the Internet of Things with edge computing. *IEEE Netw.* **32**(1), 96–101 (2018)
11. Luo, X., Zhang, D., Yang, L.T., Liu, J., Chang, X., Ning, H.: A Kernel machine-based secure data sensing and fusion scheme in wireless sensor networks for the cyber-physical systems. *Future Gener. Comput. Syst.* **61**, 85–96 (2016)
12. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: feature learning by inpainting. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544 (2016)
13. Raza, U., Camerra, A., Murphy, A.L., Palpanas, T., Picco, G.P.: Practical data prediction for real-world wireless sensor networks. *IEEE Trans. Knowl. Data Eng.* **27**(8), 2231–2244 (2015)
14. Shen, Z., Zhang, T., Jin, J., Yokota, K., Tagami, A., Higashino, T.: ICCF: an information-centric collaborative fog platform for building energy management systems. *IEEE Access* **7**, 40402–40415 (2019)
15. Shen, Z., Yokota, K., Tagami, A., Higashino, T.: Development of energy-efficient sensor networks by minimizing sensors numbers with a machine learning model. In: *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 741–746. IEEE (2018)
16. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: *Advances in Neural Information Processing Systems*, pp. 3104–3112 (2014)
17. Tortonesi, M., Govoni, M., Morelli, A., Riberto, G., Stefanelli, C., Suri, N.: Taming the IoT data deluge: an innovative information-centric service model for fog computing applications. *Future Gener. Comput. Syst.* **93**, 888–902 (2019)
18. Trihinas, D., Pallis, G., Dikaiakos, M.D.: ADMin: adaptive monitoring dissemination for the Internet of Things. In: *Proceedings of the IEEE Conference on Computer Communications*, pp. 1–9. IEEE (2017)
19. Zhang, T., Jin, J., Yang, Y.: RA-FSD: a rate-adaptive fog service delivery platform. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds.) *ICSOC 2018. LNCS*, vol. 11236, pp. 246–254. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03596-9_16