DOI: 10.1109/JIOT.2018.2879454

Blind Entity Identification for Agricultural IoT Deployments

Anandarup Mukherjee, Sudip Misra, Narendra Singh Raghuwanshi, and Sushmita Mitra

***Abstract*—Integration of various technologies to an Internet of Things (IoT) framework share the common goals of a consistent and structured data format that can be applied to any device, given the vast application scope of IoT. Additional goals include minimizing channel traffic and system energy consumption. In this work, we propose to dismiss the requirement of certain seemingly crucial identifier fields from packets arriving through various sensor nodes in an agricultural IoT deployment. The proposed approach reduces packet size, thereby reducing channel traffic and energy consumption, as well as retaining the capability of identifying these originating nodes. We propose a method of a blind agricultural IoT node and sensor identification, which can be sourced and operated from a master node as well as a remote server. Additionally, this scheme has the capability of detecting the radio link quality between the master and slave nodes in a rudimentary form, as well as identifying the sensor nodes. We successfully trained and tested various multi-layer perceptron (MLP)-based models for blind identification, in real-time, using our implemented agricultural IoT implementation. The effect of changes in learning rate and momentum of the optimizer on the accuracy of classification is also studied. The projected cumulative energy savings across the network architecture, of our scheme, in conjunction with TCP/IP header compression techniques, are substantial. For a 100 node deployment using a combination of the proposed blind identification reduced sampling strategies over regular IPv4-based TCP/IP connection, an estimated annual saving of $\approx 99\%$ is projected.**

***Keywords*—** Agricultural IoT, Green Computing, Multi-layer Perceptron, Payload Compression.

I. Introduction

The use of IoT has gradually permeated all walks of life such as monitoring various macro-aspects of human habitations – cities, pollution mapping and monitoring, agriculture, irrigation management, health-care, power supply management, transportation, education and security – as well as the micro-aspects such as – smart home automation, child and elderly care, fire detection and vehicle monitors. The evolution of simple wireless sensor networks (WSNs) to IoT has significantly reduced end-to-end human intervention in routine tasks, such as agriculture, home automation, industrial monitoring, and environmental monitoring. The use of low-cost wireless sensors and robust addressing strategies are helping in the reuse and redeployment of sensors in various IoT applications and scenarios [1]. Starting from narrowly focused and homogeneous primitive sensor-based local processing and actuation methods, present-day networked systems have evolved into widely dispersed, heterogeneous sensor and actuator systems with remote or distributed processing. These systems are becoming popular due to their scalability, ease of integration, and reduced cost of implementation. Additionally, domains such as big-data processing and cloud computing [2], integrated with IoT platform and addressing schemes are enabling large-scale fusion and automated analysis of data [3].

A. Mukherjee & S. Misra are with the School of Information Technology, Indian Institute of Technology Kharagpur, India, e-mail: (anandarupmukherjee@ieee.org).

N. S. Raghuwanshi is with the Department of Agriculture and Food Engineering, Indian Institute of Technology Kharagpur, India

S. Mitra is with the Machine Intelligence Unit of the Indian Statistical Institute, Kolkata, India

A. Implementation Overview

In this work, as outlined in the architecture in Fig. 1, we use an ensemble of MLP [4] models on a master node as well as a remote server to automatically detect link errors, identify sensor nodes, and then, identify each sensor from the packets received from various field sensor nodes. MLP has proved to be a powerful tool for pattern differentiation tasks [5]. The solar-powered, field-deployed sensor nodes (slave nodes) transmit their data to a nearby master node by means of short-range wireless communication radios, which in our case are Zigbee S1 radio modules. The aggregated data from the

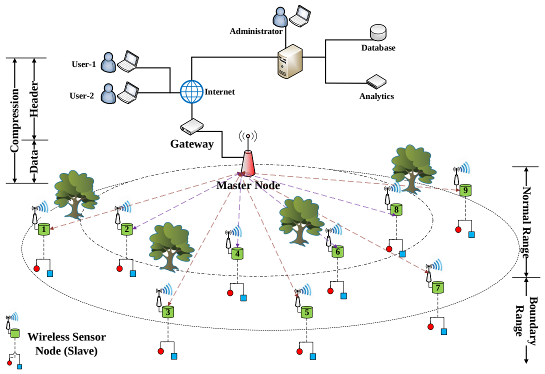
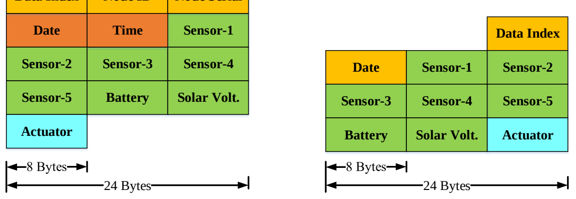


Figure 1: Architecture of the implemented agricultural IoT-based field monitoring system.

master node is then transmitted to a remote server through the Internet by making use of long-range wireless radios from a gateway device, which in our deployment is a cellular network supporting GSM module. The remote server acts as a data storage unit as well as an analytical engine prepped with the trained MLP models. Subscribers/users of this system can access the raw data as well as visualize the data arriving at the server via the Internet. In the field, we focus on our system deployed for monitoring various agricultural field parameters – soil moisture (S_m), soil temperature (S_{st}), humidity (S_h), rainfall (S_r), solar-radiation (S_{sr}), ambient temperature (S_{at}), battery voltage of each node ($\mathcal{S}_{vb}$), and solar voltage generated at the solar panel of each node ($\mathcal{S}_{sv}$) – the architecture of which is shown in Fig. 1. Practices such as precision agriculture, irrigation management, and agricultural management are rapidly shifting their focus on Information and Communication Technologies (ICTs). These modern methods rely on the use of customized sensors for weather, soil, water and plant monitoring for enabling the use of agricultural IoT [6]. The vast domains and agricultural expanses over which a multitude of sensors are deployed generate huge amount of data $\mathcal{D}$ such that for n sensor nodes, each generating data d per unit time, the overall data traffic generated in the network per unit time is represented in the form of a tuple as $\mathcal{D} = \{d_1, d_2, d_3, \cdots d_n\}$. The data from slave nodes $\mathcal{N}_i, \forall i \in (1, n)$ can be either from a soil parameter monitoring node or a weather monitoring station. Therefore, $d_i, \forall i \in (1, n)$ can be either from a soil sensor node or a weather station in our implementation. Further, each of these constituent data from individual sensor nodes d can be again represented as a tuple such that $d_{soil} = \{\mathcal{S}_{head}, \mathcal{S}_{m1}, \mathcal{S}_{m2}, \mathcal{S}_{m3}, \mathcal{S}_{m4}, \mathcal{S}_{st}, \mathcal{S}_{bv}, \mathcal{S}_{sv}, \mathcal{S}_{foot}\}$ and $d_{weather} = \{\mathcal{S}_{head}, \mathcal{S}_{h}, \mathcal{S}_{r}, \mathcal{S}_{sr}, \mathcal{S}_{at}, \mathcal{S}_{bv}, \mathcal{S}_{sv}, \mathcal{S}_{foot}\}$. $\mathcal{S}_{head}$, $\mathcal{S}_{foot}$, d_{soil} and $d_{weather}$ represent the packet header, footer, data from soil parameter sensor node, and data from weather monitoring station, respectively. Data of this magnitude cannot be locally processed and analyzed, albeit at the cost of making these implementations costly. The use of low-cost sensors with reconfigurable radios, and an easily accommodating architecture can only be sustained and made cheaper if the processing is performed remotely at a master node as well as a remote server, and not on the field-deployed sensor nodes themselves. To further enhance the usefulness of this approach, reduction in operating costs and channel traffic are required for seamless operation of this architecture.

Assumption 1. *The slave nodes $\mathcal{N}$, which are placed at the boundary of the master node's radio range, fail to transmit data due to link loss only.*

Assumption 2. *The consolidated readings from each node ($\mathcal{N}_1, \mathcal{N}_2, \cdots$), which consist of soil moisture, soil temperature, battery voltage and solar voltage sensors, have very minute variations. These are due to variations in depth of the soil moisture sensor, terrain type, temperature conditions and soil moisture heterogeneity [7], [8].*



(a) Original Packet format (b) Optimized Packet format

Figure 2: Format of packets sent from slave to master nodes.

Fig. 1 shows the characteristics of links between the master and slave nodes of our deployed system. The nodes are assumed to be fixed and well optimized, due to the use of low power solutions such as Zigbee or 6LoWPAN. Without changes in the existing radio standards, modifications in the radio links are not an option without increasing the cost of deployment. Two approaches are undertaken to reduce the network data-load as outlined in Fig. 1 – master-node data compression and header compression over the TCP/IP network. Energy conservation within the domain of a master-node make use of schemes, such as compressive sensing, sink placement, and cluster-based routing [9], whereas energy conservation follows techniques such as header compression [10], edge computing [11], fog computing and other paradigms [12] when the data is in the realm of TCP/IP.

Within the domain of the master node, to reduce $\mathcal{D}$, the sampling instants of the sensor nodes themselves are changed to achieve reduced network data load $\mathcal{D}_{opt}$. For an original sensor sampling frequency of $\omega_{original}$, a reduced sensor sampling frequency $\omega_{reduced}$, and a scaling factor k, $\mathcal{D}_{opt}(\omega_{reduced}) < \mathcal{D}(\omega_{original})$, such that $\omega_{reduced} = \frac{1}{k}\omega_{original}, \forall k > 1$. This is possible only for applications such as agriculture, as they deal with data which is not of temporal criticality and can be deemed as non-critical data. Delays in transmission of agricultural field data do not affect the purpose of deployment — monitoring changes in field and weather parameters, which tend to change slowly over time — as compared to time-critical systems such as fire monitors and health monitoring systems. The additional improvement is proposed in the communication link between the master node and the remote server. Prior to transmission to the remote server, the master node packetizes data in the correct order so that data from the individual nodes as well as sensors can be identified. A sample packet of the data from the master node to the server is shown in Fig. 2(a). Here, we propose a blind identification method using pre-trained multilayer perceptrons (MLP), where the master node periodically uploads data to the server with just the sensor data, data index, and date fields, as shown in Fig. 2(b). The learning algorithms in the remote server are responsible for identifying link errors, node identifiers, as well as sensor identifiers, from the incoming optimized data packets. The blind identification approach is evaluated at the remote server as well as the master node to reduce the unnecessary network transmission latency between the master node and the remote server.

Within the purview of each master node, we intentionally implement sensor nodes with low-computational capability, in order to allow for an economical large-scale sensor-node implementation. Any approach which increases the computational complexity of the slave nodes, such as compressive sensing, and intelligent routing, automatically increases the overall cost of the field implementation. Hence, our proposed scheme implements slave nodes, which simply collect data from various locations in a field and forward it to a master node, which inturn removes the local packet identifiers and tags, prior to forwarding it over the Internet. Once the sensed data leaves the master node, header compression is applied on it to further reduce the network data-load of the implementation on the Internet. Robust Header Compression (ROHC) is one of the most popular techniques dealing with packet header compression over the Internet [10]. In lieu of our implementation, ROHC accentuated 3G/LTE telecommunication uplinks connecting the master node to the remote server is considered in addition to our proposed scheme, for compression of the packets gathered from the master nodes.

This work is organised as follows. Section 2 explores the state of the art in this domain. Section 3 provides the system overview and describes the dataset, followed by the details of the experimental set-up in Section 4. Section 5 provides the performance analysis of our proposed scheme, followed by the conclusion of this work in Section 6.

II. Related Work

The involvement of IoT with various ancillary technologies such as cloud, machine learning, big data and others has resulted in rapid development of societally beneficial and impactful solutions boasting of a multitude of features – intelligence, scalability, automation, virtualization, and others. IoT platforms integrated with cloud-based infrastructure for load provisioning and execution of applications provide the additional advantages of virtualization of infrastructure as demonstrated by Truong *et al.* [13]. Similar works on IoT-Cloud framework for cooperation between cloud and smart devices was demonstrated by Kum *et al.* [2], whereas Nastic *et al.* [14] demonstrated the use of IoT-based software-defined units for controlling cloud systems. The use of re-configurable smart sensors for IoT in industrial environments was demonstrated by Chi *et al.* [15]; whereas, Mihai *et al.* [16] demonstrated the use of IoT and WSN in long-term environmental monitoring. Offloading data to a more powerful remote computing facility allows for processing intensive tasks such as automated analytics, learning, and automation, which is yet another important aspect of IoT [17].

Agriculture is a domain which is actively transitioning towards the use of IoT and machine learning for monitoring, analysis, prediction, and automation in tasks, which was previously manual and intuition-based. Transcending from traditional implementations of IoT, Kaloxylos *et al.* [6] described the implementation of an IoT-cloud-based precision farming system, achieving an Agri-IoT-Cloud based framework. More recently, IoT-based agricultural solution pilots are deployed across multiple countries in order to usher in food-security, reduce wastage of food, and scientifically optimize agricultural practices [18]. Similarly, IoT platforms are developed for precision agriculture as well as ecological monitoring [19].

The surging plethora of data types, architectures, and applications associated with IoT face the bottleneck of network bandwidth, which for most cases is established using legacy technologies, such as IPv4 over TCP or UDP. Various approaches for optimizing network bandwidth usage of in-place network infrastructure are proposed over the years. Approaches ranging from reduction of transmitted packet size [10] and judicious use of edge devices [20] to edge mining [11] have reported various degrees of successes in network bandwidth optimization. Header compression solutions such as ROHC [10], ROHCv2 [21], and others [12] build upon the fact that headers are static/constant in an IP stream.

Synthesis: Solutions addressing the problem of payload minimization of the gathered data from the local networks connected to an IoT infrastructure is lacking, especially in agricultural scenarios. As, the bulk of the nodes and sensors deployed in these scenarios are resource and energy constrained, strategies must be devised to minimize their energy consumption in the long run. Additionally, these nodes are expected to house multiple heterogeneous sensors, which increases the payload size generated from each of these nodes. The payload, being uploaded to the cloud through the Internet reaches gargantuan proportions, in cases of large IoT deployment with frequent data sampling.

III. System Overview

The architecture of our implemented agricultural IoT is divided into four parts – Slave nodes, Master node, IoT framework and the RSL (Remote Stochastic Learning) module. The architecture of this implementation is shown in Fig. 1. Each slave node $\mathcal{N}_i$ hosts a ATmega328P processor with a processing speed of $16MHz$, and is equipped with four soil moisture sensors S_{m1} to S_{m4}, placed at depths of $15cm$, $30cm$, $45cm$ and $60cm$. Additionally, the slave nodes are equipped with a soil temperature sensor, battery voltage detector, solar voltage detector circuitry which is connected to the battery and a solar panel. The radio used is a Zigbee module, which is configured to work in a mesh network. The customized mini-processor in the slave node also control relays, which may be connected to pumps for automatic irrigation by triggering them from the remote server. Some of the deployed slave nodes in our implementation are shown in Fig. 3.

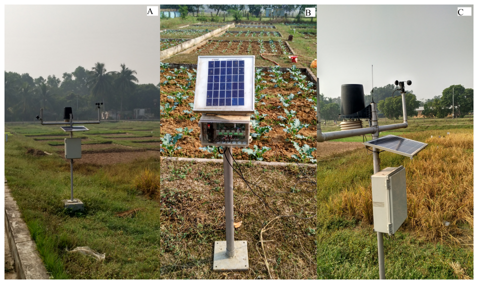


Figure 3: Our implemented agricultural IoT slave nodes. (A) Weather station, (B) Soil sensor node, (C) Weather station.

The master node communicates to the slave nodes via Zigbee radio links. The master node hosts a quad-core ARM Cortex $A53$ (ARMv8), with a processing speed of $1.2GHz$. It also acts as the local gateway for its networked slave nodes, providing them with locally identifiable addresses within its operational domain. The communications from and to the remote server to the slave nodes are redirected through this gateway, which also happens to be the master node for this IoT framework. Multiple such gateways can be easily incorporated into the IoT framework to send their data to the remote server via the Internet. Our implemented server runs a Windows 8 operating system with an Intel i3 processor, and clock speed of $3.72GHz$. The RSL module is hosted on the remote server. The data from all nodes $\mathcal{D}_{opt}$ in the field are continually uploaded to the remote server by means of web-sockets through the gateway. The pre-trained models (Algorithm 1) in the server are used for classifying the incoming data. The data, after being sorted by the RSL module are analyzed for determining the field stress conditions and water requirements of crops. The actuators controlling the irrigation pumps can be accordingly activated from the server based on the soil parameter threshold conditions programmed in the server module. The whole field-side, slave node placement is divided into two parts based on the radio range R of the slave nodes from the master node – *Normal range* and *Boundary range*. The nodes in the boundary range tend to lose connection to the master node occasionally due to bad weather conditions and other unavoidable circumstances.

A. Data

The data obtained from various sensors affixed to a slave node d_i is sent to the master node, from where it is uploaded to a remote server via the Internet using the IoT platform. Initially, the packets are uploaded in the format given in Fig. 2(a). Each slave sends $d_i = 14$ bytes of data per second. The data logging interval ω can be increased or decreased on-demand by means of text messages from its registered users/ owners, which changes the values of the scaling factor k accordingly. We, additionally, reduce some of the fields from the data packets, and henceforth, refer to it as the optimized packet (Fig. 2(b)). The reduction of 4 bytes of data per slave, per second, is a significant reduction in data volume, in case of long-term monitoring using large-scale sensor networks.

Specific to our implementation, Nodes 1 and 3 are placed in the boundary zone, as depicted in Fig. 1. They have a high probability of link errors (Assumption 1). Nodes 2 and 4 are in the normal range of the master node and suffer from relatively, minuscule errors due to link failure. Subsequently, even after reducing the packet size, we can further reduce the data load to the server by reducing the data polling interval to twice per hour. As the sensors, in this case, are used for non-critical data measurement and the soil parameters tend to change slowly over time, this drastic reduction in data sampling does not significantly affect the overall decision-making process. This strategy would have been unthinkable in case of critical data, such as those obtained from health-care, traffic [22] or disaster management implementations of IoT. However, we find that the packet size reduction works fine with all applications of this framework.

Assumption 3. *The individual sensors in a sensor node have negligible fingerprints, even if they are of the same type. This may be due to insufficient hardware calibration [23] or even due to atomic level irregularities induced at the time of device fabrication.*

Assumption 4. *The soil moisture sensors depend on the variation in dielectric constants between air, soil, and water to measure soil moisture content. This varies slightly due to varying electromagnetic interference (EMI) at various locations.*

Proof. The changes in dielectric properties of a substance, due to the introduction of an electromagnetic field, depends on the polarization effect of molecules. For a relative dielectric permittivity of $\mathcal{E}$, the variation of $\mathcal{E}$ with frequency ω as described by Romano *et al.* [24] is given as:

$$\mathcal{E}(\omega) = \mathcal{E}'(\omega) - j[\mathcal{E}''(\omega) + \frac{\sigma_{DC}}{\omega \mathcal{E}_0}] \quad (1)$$

where,
$\mathcal{E}'(\omega) \rightarrow$ real component and signifies the energy stored in the system due to alignment of dipoles and electromagnetic interference (EMI).
$\mathcal{E}''(\omega) \rightarrow$ imaginary component and accounts for the dissipation of energy.
$\sigma_{DC} \rightarrow$ signifies DC electrical conductivity and depends on actual transport of charge carriers.
$\omega \rightarrow$ angular frequency of the imposed electromagnetic field.
Changes in $\mathcal{E}'(\omega)$ due to unnecessary EM interference or soil properties give rise to variations in sensor readings from the same type of soil moisture sensors. However, these changes are very small [24], and are neglected from consideration in this study. □

IV. Experimental Set-up

This section discusses the experimental set-up used in the study. This covers the methodology used, remote server-based learning, and the computation of the carbon footprint for the implementation. Four assumptions are taken into account while applying the methodologies on the discussed architecture and learning modules in the server, as given by Assumptions 1 to 4.

A. Overview of the Experiments

As discussed in Section 3, the placement of nodes is mainly divided into two groups – *Normal range* and *Boundary range*. We classify the unlabeled data arriving at the server and fit against the models generated using historical data from both normal and boundary range slave nodes. The method of classifying the data is outlined in Algorithm 1. On the training data-set, we first choose the ideal learning rate and momentum values (described in the next sub-section) of the Multi-layer Perceptron (MLP)-based classifier. Upon finally arriving at an optimum value, which can be used for the three cases – *Link error detection*, *Node identification* and *Sensor identification* – we demonstrate the accuracy of detection and loss minimization by a Mean Squared Loss Estimator (MSE), for our method. As a typical master node is resource constrained in terms of computation power, we modify the MLP architecture at the remote server to make its execution feasible on the master node. We term the original MLP model at the remote server as the *Heavy model* and the modified one on the master node as the *Light model*. Finally, we calculate the carbon footprint of our approach against the previous unoptimized approach.

The carbon footprint of a system is calculated in terms of emission of CO_2 released in the atmosphere in order to generate the power consumed by the system. The emission factor for an electric system is 6.89551×10^{-4} metric tons of CO_2/kWh, as per e-GRID of the United States Environment Protection Agency, from the year 2010 data. In other words, consumption of $1kWh$ of electricity releases $0.16kg$ of CO_2 in the atmosphere. This quantity may seem insignificant, but it tends to build up in cases of large sensor network deployments, running over prolonged periods of time.

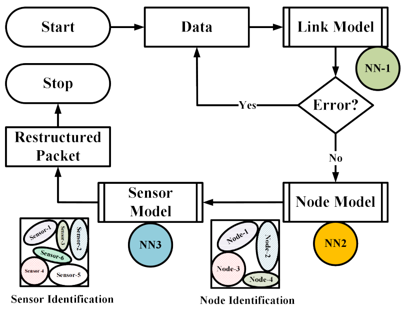


Figure 4: The dataflow representation of the server-side packet restructuring module or the RSL module.

B. Methodology

The data from *Normal range* nodes are grouped in a class and the *Boundary range* nodes are grouped into another class for training the MLP-based classifier. Our approach of dropping node and sensor identifiers from the packets makes it necessary to have an identification process in place which will categorize the uploaded data accordingly. Algorithm 1 identifies link errors as well as categorizes the incoming data into the various node and sensor classes according to their fingerprints. Although the sensors may be the same in all nodes, they do have minute variations of their own due to insufficient sensor and hardware calibration, a mismatch in placement depth [23], varying land topology and heterogeneity in soil moisture profiles [8], [7]. Similarly, the individual sensors are also differentiated from each other due to the inherent presence of minute calibration mismatch, fabrication irregularities and non-linearity; this irregularity proves beneficial for our individual sensor identification. Fig. 4 shows the overall collective scheme of using the three trained models – *Model Q*, *Model N*, and *Model S* – in detecting link errors, identifying sensor nodes and then identifying the individual sensors in each node.

C. Remote Learning

The following three modules – *Model Q*, *Model N*, and *Model S* – comprise the RSL algorithm in the server. The working of these modules is shown in Fig. 4. These models are trained based on previously collected data. New data from the sensor nodes coming into the server are classified based on these pre-trained models. The ensemble of these three models is used for blind identification of packets and its features from a re-structured and reduced packet, arriving at the remote server.

1) Optimizer Selection: The optimizer selection ensures precise and speedy decision making for any machine learning-based algorithm. A Stochastic Gradient Descent (SGD) [25] approach over back-propagation is chosen due to its high variance, low running cost, low-memory requirements and ability to handle large data sizes over the entire training set.

Algorithm 1 RSL – Remote Stochastic Learning

```
1: Inputs:
2:     $D_{m\times 10} = [d_1, d_2, .., d_{10}]$, $\delta_{m\times 10} = [d_1, d_2, .., d_10]$
3:   ▷ % $\delta$ is the live incoming data from the field deployed nodes to the server %
4:   ▷ % $D$ is the stored data from the field deployed nodes, used for training the models in the server %
5:
6: Output: $P(Q_k), P(N_k), P(S_k)$
7:   ▷ % probability of occurrence of a class. $Q_k, N_k, S_k$ denote the error in link quality, node ID and sensor ID, respectively. %
8: Initialize Parameters:
9:   ▷ % Divide $D$ in various classes for training %
10:     $[N_{near}, N_{far}] \in D_{m\times 10}$
11:     $[N_1, N_2, N_3, N_4] \in D_{m\times 10}$
12:     $[S_1, S_2, S_3, S_4, S_5, S_6] \in D_{m\times 10}$
13: Initialize:
14: while ($\delta$) do
15:     for ($i = 0; i < 2; i++$) do
16:         $Q_k[i] \leftarrow$ Predict(Model Q, $\delta_{m\times 10}$)
17:     for ($i = 0; i < 4; i++$) do
18:         $N_k[i] \leftarrow$ Predict(Model N, $\delta_{m\times 10}$)
19:     for ($i = 0; i < 6; i++$) do
20:         $S_k[i] \leftarrow$ Predict(Model S, $\delta_{m\times 10}$)
21:     Return $P(Q_k]) \leftarrow max(Q_k)$, $P(N_k) \leftarrow max(N_k)$, $P(S_k) \leftarrow max(S_k)$
```

```
procedure CREATE MODEL Q
                                   ▷ % Link Quality Learner %
    Inputs: $N_{near}[1, 2, ..n], N_{far}[1, 2, ..n], f(Activation)$, $\alpha_{learn} \rightarrow$ learning rate, $\Delta p \rightarrow$ momentum, Optimizer, $Layer_{hidden}, f(loss)$
    Output: Model Q
```

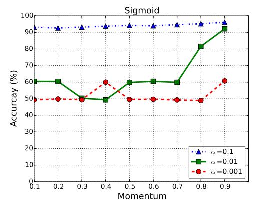
SGD operates by updating an objective $J(\theta)$, where θ is one of its parameters, as

$$\theta := \theta - \alpha \nabla_{\theta} E[J(\theta)] \tag{2}$$

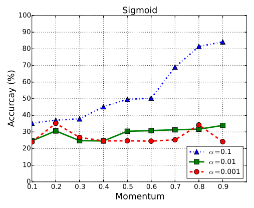
where, $E[J(\theta)]$ is the expectation of the objective and is calculated over the entire training set. However, in an SGD update, the expectation term is eliminated and is represented as $\theta := \theta - \alpha\nabla_{\theta} J(\theta; x^{(i)}, y^{(i)})$. The values of $(x^{(i)}, y^{(i)})$ are obtained from the i^{th} iteration of the training set. The objective updates are calculated based on small training data.

```
procedure CREATE MODEL N
                                   ▷ % Node ID Learner %
    Inputs: $N_1[1, 2, ..n], N_2[1, 2, ..n], N_3[1, 2, ..n]$, $N_4[1, 2, ..n], f(Activation)$, $\alpha_{learn} \rightarrow$ learning rate, $\Delta p \rightarrow$ momentum, Optimizer, $Layer_{hidden}, f(loss)$
    Output: Model N
```

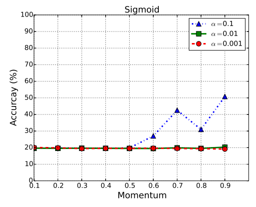
```
procedure CREATE MODEL S
                                   ▷ % Sensor ID Learner %
    Inputs: $S_1[1, 2, ..n], S_2[1, 2, ..n], S_3[1, 2, ..n]$, $S_4[1, 2, ..n], S_5[1, 2, ..n], S_6[1, 2, ..n], S_7[1, 2, ..n]$, $f(Activation), \alpha_{learn} \rightarrow$ learning rate, $\Delta p \rightarrow$ momentum, Optimizer, $Layer_{hidden}, f(loss)$
    Output: Model S
```



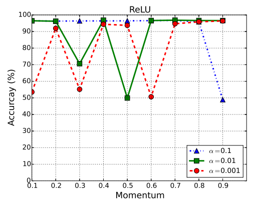
(a) LinkQ (Sigmoid)



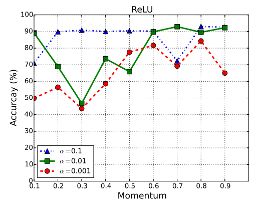
(b) NodeID (Sigmoid)



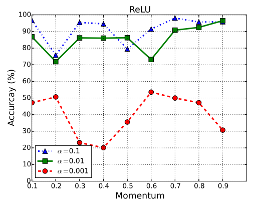
(c) SensorID (Sigmoid)



(d) LinkQ (ReLU)



(e) NodeID (ReLU)



(f) SensorID (ReLU)

Figure 5: Effect of γ and α variations on the accuracy of classification for the RSL algorithms for the heavy model at the remote server.

2) Activation Function Selection: A proper activation function selection is critical to our operation as it defines the distribution of the data to pass. For our approach, a Regularized Linear Unit (ReLU) [26] is used as the activation function, instead of the traditional Sigmoid function, as the output of the Sigmoid function lies in the range $[0, 1]$ and tends to saturate beyond this. The ReLU, on the other hand has a range of $[0, \infty]$ and can be used for modeling real numbers, instead of just probability (as is done by Sigmoid functions). A Sigmoid function is represented as $\sigma(x) = (1 + e^{-x})^{-1}$. In contrast to the Sigmoid function, the ReLU function is denoted as $f(x) = \Sigma_{i=1}^{\infty}\sigma(x - i + 0.5)$. $f(x)$ is similar to $f(x) = log(1 + e^x)$, which is also referred to as a *Softplus* function and is approximated as:

$$f(x) = log(1 + e^x) \approx max(0, x + N(0, 1)) \tag{3}$$

This is often referred to as Rectified Linear function (ReL). The gradient of the ReL function does not vanish, unlike the Sigmoid-based activation functions.

3) Learning: The data is generally shuffled prior to training in order to avoid biasing due to unforeseen, repetitive patterns in the data-set. Another function, the momentum (γ), is used to speed up the convergence of the objective. The update of momentum is given by:

$$v = \gamma v + \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)}), \quad \forall size(v) = size(\theta) \tag{4}$$

$$\theta = \theta - v \tag{5}$$

where, v is the current velocity and $\gamma \in (0, 1]$. Section 5 shows the learning rate (α) and momentum value (γ) selection for the given agricultural sensor data-set. It also demonstrates the superiority of the ReLU-based activation over Sigmoid-based functions in SGD implementation of our system.

D. Calculating the Carbon Footprint

The carbon footprint of the architecture and our proposed improvement is calculated as:

$$\mathcal{S}_{S-M} = \mathcal{S}_{payload-14} + \mathcal{S}_{zigbee-API} \quad \text{bytes} \tag{6}$$

$$\mathcal{S}_{S-M} = \mathcal{S}_{payload-14} + 24 \quad \text{bytes} \tag{7}$$

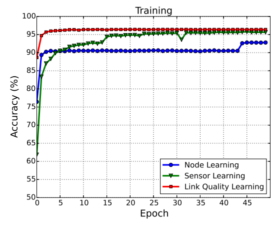
where, $\mathcal{S}_{S-M}$ is the size of data received at the master node from the slave, $\mathcal{S}_{payload-14}$ is the data collected from the sensors and other parameters of the slave node using the original packet format ($\simeq$ 14 bytes, in this case), as shown in Fig. 2(a). $\mathcal{S}_{zigbee-API}$ is the Zigbee API's frame format, which is fixed at 24 bytes [27]. Again, it takes $\mathcal{P}_{1-bit} = 184.9\mu W/bit$ to transmit 1 bit of data; Zigbee takes 1 second to transmit 24 bytes of data [27]. As we focus only on reducing the payload of the Zigbee data, from the slave to the master node. Henceforth, we shall consider only the payload part for our carbon footprint computation. Moreover, the slave and master nodes run on solar power and can be safely considered as zero-carbon footprint devices.

The part of the architecture, starting from the gateway to the remote server (as shown in Fig. 1) have high energy requirements and rely on traditional and mostly non-renewable energy sources. However, the use of approaches such as ROHC-based header compression significantly reduces the energy consumption of this part of the network architecture. The remaining part of the architecture consisting of sensor nodes is dependent on solar energy for regular operation and does not generate a carbon-footprint. Hence, only the path from gateway to the remote server is considered for carbon footprint computation of our approach. Since the master node discards the Zigbee-API frames, we represent the effective packet size sent to the remote server as $\mathcal{S}_U = \mathcal{S}_{payload-14}$, where $\mathcal{S}_U$ is the unoptimized packet size. Furthermore, $\mathcal{S}_O$ and $\mathcal{S}_{OR}$ are assigned to indicate optimized packet size and optimized packet size with reduced sampling, respectively. For $\mathcal{S}_{payload-10}$ and $\mathcal{S}_{payload-10-r}$ denoting the reduced payload and the reduced payload with reduced data sampling, we assign, $\mathcal{S}_O = \mathcal{S}_{payload-10}$ and $\mathcal{S}_{OR} = \mathcal{S}_{payload-10-r}$. Subsequently, for $\mathcal{P}(kWh)$ signifying the power consumed (in kW) for transmitting $\mathcal{S}_{payload}$ bytes of data per hour, and $\mathcal{C}_{\mathcal{P}}$ denoting the carbon footprint in $kg-CO_2$ released into the atmosphere, we denote $\mathcal{P}(kWh) = 185.9\times10^{-6}\times\mathcal{S}_{payload}\times3.6$ and calculate $\mathcal{C}_{\mathcal{P}} = \mathcal{P}(kWh) \times 0.16 \quad Kg - CO_2$.

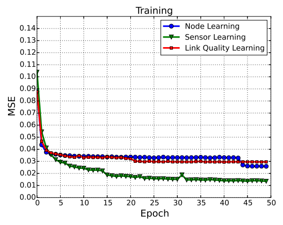
V. Performance Analysis

This section discusses the output obtained from the experimental setup. The MLP is trained over the data-set, for both Sigmoid and ReLU activations. The learning rate (α) and momentum (γ) are varied to check the accuracy of classification for all three RSL modules – $ModelQ$, $ModelN$

and $ModelS$ – for determining the optimum MLP parameters. To sum-up the MLP results,



(a) Change in Accuracy



(b) Change in Loss

Figure 6: The overall classification accuracy and loss of the trained models for Node identification, Sensor identification and Link quality detection for the heavy model at the remote server.

1) Figs.5(a) and 5(d) show the performance of the Sigmoid and ReLU-based activation functions in determining link errors. The ReLU-based parameters, with $\alpha = [0.01, 0.001]$ and $0.7 \geq \gamma \leq 0.9$, consistently perform better than all other cases.
2) Figs. 5(b) and 5(e) show that ReLU with with $\alpha = [0.1, 0.01]$ and $0.8 \geq \gamma \leq 0.9$, show superior performance over Sigmoid activation for classifying various nodes, from the incoming data-set.
3) Figs. 5(c) and 5(f) show performance of the MLP in classifying the various sensors in a node. Here, ReLU activation for $\alpha = 0.1$ and $0.7 \geq \gamma \leq 0.9$ give the best performance, for classifying individual sensors, from the incoming data.

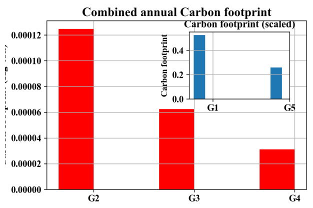
The optimum hyper-parameters, determined for the training of the three heavy RSL modules at the remote server – $ModelQ$, $ModelN$ and $ModelS$ – are evaluated using a RELU *activation*, SGD *optimizer*, MSE *loss function* with $\alpha = 0.01$ and $\gamma = 0.9$ for the *Decay*, *Epoch*, and *Batch-size* of the MLP set to 1×10^{-6}, 50, and 16, respectively. The model specific MLP parameters for the three models are, 2 inputs, 2 outputs and 2 hidden layers with $64, 128$ neurons each for $ModelQ$, 7 inputs, 4 outputs and 2 hidden layers with $128, 128$ neurons each for $ModelN$, and 4 inputs, 5 outputs and $64, 128$ neurons in 2 hidden layers for $ModelS$, respectively. The corresponding accuracy of classification, achieved by using these parameters are 96% for $ModelQ$, 92.5% for $ModelN$ and 95.6% for $ModelS$ units of the RSL module at the remote server. Figs. 6(a) and 6(b) show the changes in accuracy and the minimization of loss (using MSE) for the RSL modules during training of the data-set. The data-set was trained over 50 epochs, as beyond this the learning curve saturates and no further improvement in learning is achieved.

Fig. 7(a) shows the carbon footprint of the agricultural deployment for 5 selected approaches, viz. – (G1) unoptimized packet over TCP/IP, (G2) optimized packet over TCP/IP, (G3) optimized packet with reduced sampling (sampled every alternate second) over TCP/IP , (G4) optimized packet with reduced sampling (sampled every fourth second) over TCP/IP, (G5) unoptimized packet over ROHC TCP/IP – calculated with respect to varying nodes for a year. For a 100 node deployment, it is estimated that, as compared to the proposed scheme, the use of ROHC instead of compressionless TCP/IP achieves an overall reduction in carbon footprint by 50% for the same datasize as shown in Fig. 7(a) G1 and G5). Additionally, comparison between approaches G1 with G2, G3, and G4 shows that our proposed scheme achieves savings of over 99% CO_2 generated annually over regular IPv4 TCP/IP transmission. Again, comparing G1 with G5 (ROHC with comparable dataload), we observe that ROHC manages to attain further savings of 50% of the CO_2 generated annually for the same datasizes. It is to be noted that header compression schemes such as ROHC do not affect the payload size of the transmitted data. Additionally, as both uncompressed IPv4 packets over TCP/IP and ROHC work beyond the purview of the master node, massive savings in generated data within the master node's domain affects the carbon footprint beyond its operational boundaries by dictating the payload size. Our proposed scheme manages to massively reduce the payload size of the data to be transmitted over the network to the remote server. The additional use of compression methods such as ROHC, which is primarily a header compression scheme, further helps in the reduction of carbon footprint (as shown in the inset plot in Fig. 7(a)) in domains such as agriculture, where the need for real-time data is not necessary.

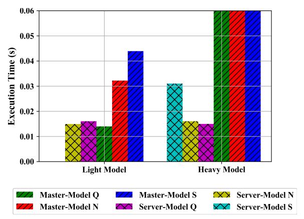
It is to be noted that having the learning models integrated directly with the master node would reduce the network load, and the latency caused thereof. However, we found that the latencies produced due to the operation of MLP models hosted at master node (as shown in Fig. 7(b)), were much higher than the 1 minute mark, rendering them useless for operation in a real-time manner. These models and their parameters are similar in all respects to the ones hosted at the remote server, which we label as *Heavy Model* in Fig. 7(b). Upon modifying the MLP model architectures to make them extremely lightweight for faster service at the master nodes (*Light Model* in Fig. 7(b), which has 2 hidden layers each with $5, 5$ neurons), albeit at the cost of detection accuracies, we found that the latencies were almost double of the ones incurred at the server. Additionally, it is worthwhile to mention that the detection accuracies of the three trained models fell below 70% in case of $ModelN$ and $ModelS$, whereas it fell below 30% for $ModelQ$. In summary, Fig. 7(b) highlights the importance and need of remote learning at the server in comparison to learning at the master nodes for the same MLP architectures.

VI. Conclusion

This work proposes an optimized method for detecting simple link failures between slave and master nodes. Additionally, the method for identifying slave nodes from data packets, even in the absence of node identifiers in the payload packet and identifying individual sensors in a node, in the absence of sensor identifier in the payload packet has also been designed and tested. This approach is tested on a remote server as well as a constrained master node, much nearer to the implementation site. Outsourcing the error detection and other identification tasks, which are computation intensive, to a remote server, allows the use of low-cost, low-specification



(a) Carbon footprint for 100 nodes over a year.



(b) Metrics for learning at the master node vs. at the remote server.

Figure 7: The carbon footprint of the agricultural deployment with 100 nodes, projected for all the 5 approaches for a year, and comparison between master-node and remote server-based learning approaches within the same implementation.

and low-energy processors for IoT deployment. This method of minimizing packet size of traffic between slave to master node and to the remote server, via the internet results in substantial reduction of network traffic, reduction of data and conservation of node energy. The choice of hosting the proposed scheme at a master node severely hampers the performance of the scheme by increasing the on-node execution time. Even implementing lighter MLP models incur higher execution time at a master node. We conclude that the benefits of hosting these MLP models at the remote server far outweigh the benefits of minor savings in network bandwidth offered by hosting these models at the master node.

In the future, we plan to implement unsupervised learning methods and other online learning methods to accommodate for heterogeneous data from sensor networks of similar or different types.

Acknowledgment

This work is supported by Information Technology Research Academy (ITRA), Media Lab Asia, under ITRA-Water Grant ITRA/15(69)/WATER/M2M/01.

References

[1] R. Fantacci, T. Pecorella, R. Viti, and C. Carlini, "A network architecture solution for efficient IOT WSN backhauling: challenges and opportunities," *IEEE Wireless Communications*, vol. 21, no. 4, pp. 113–119, August 2014.

[2] S. W. Kum, J. Moon, T. Lim, and J. I. Park, "A novel design of IoT cloud delegate framework to harmonize cloud-scale IoT services," in *IEEE International Conference on Consumer Electronics (ICCE), 2015*, Jan 2015, pp. 247–248.

[3] E. Gokce, A. K. Shrivastava, J. J. Cho, Y. Ding *et al.*, "Decision fusion from heterogeneous sensors in surveillance sensor systems," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 1, pp. 228–233, 2011.

[4] S. K. Pal and S. Mitra, "Noisy fingerprint classification using multilayer perceptron with fuzzy geometrical and textural features," *Fuzzy sets and systems*, vol. 80, no. 2, pp. 121–132, 1996.

[5] S. Mitra, S. N. Sarbadhikari, and S. K. Pal, "An mlp-based model for identifying qeeg in depression," *International journal of bio-medical computing*, vol. 43, no. 3, pp. 179–187, 1996.

[6] A. Kaloxylos, R. Eigenmann, F. Teye, Z. Politopoulou, S. Wolfert, C. Shrank, M. Dillinger, I. Lampropoulou, E. Antoniou, L. Pesonen *et al.*, "Farm management systems and the Future Internet era," *Computers and Electronics in Agriculture*, vol. 89, pp. 130–144, 2012.

[7] N. W. Chaney, J. K. Roundy, J. E. Herrera-Estrada, and E. F. Wood, "High-resolution modeling of the spatial heterogeneity of soil moisture: Applications in network design," *Water Resources Research*, vol. 51, no. 1, pp. 619–638, 2015.

[8] J. F. Rihani, F. K. Chow, and R. M. Maxwell, "Isolating effects of terrain and soil moisture heterogeneity on the atmospheric boundary layer: Idealized simulations to diagnose land-atmosphere feedbacks," *Journal of Advances in Modeling Earth Systems*, vol. 7, no. 2, pp. 915–937, 2015.

[9] S. P. Tirani and A. Avokh, "On the performance of sink placement in WSNs considering energy-balanced compressive sensing-based data aggregation," *Journal of Network and Computer Applications*, 2018.

[10] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu *et al.*, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," Tech. Rep., 2001.

[11] E. I. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith, and R. Rednic, "Edge Mining the Internet of Things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3816–3825, Oct 2013.

[12] W. Wu and Z. Ding, "On Efficient Packet-Switched Wireless Networking: A Markovian Approach to Trans-Layer Design and Optimization of ROHC," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4232–4245, July 2017.

[13] H.-L. Truong and S. Dustdar, "Principles for Engineering IoT Cloud Systems," *IEEE Cloud Computing*, vol. 2, no. 2, pp. 68–76, Mar 2015.

[14] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning Software-Defined IoT Cloud Systems," in *International Conference on Future Internet of Things and Cloud (FiCloud), 2014*, Aug 2014, pp. 288–295.

[15] Q. Chi, H. Yan, C. Zhang, Z. Pang, and L. D. Xu, "A Reconfigurable Smart Sensor Interface for Industrial WSN in IoT Environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1417–1425, May 2014.

[16] M. Lazarescu, "Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 1, pp. 45–54, March 2013.

[17] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Adaptive Clustering for Dynamic IoT Data Streams," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 64–74, Feb 2017.

[18] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis, "IoT in Agriculture: Designing a Europe-Wide Large-Scale Pilot," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 26–33, 2017.

[19] T. Popović, N. Latinović, A. Pešić, Ž. Zečević, B. Krstajić, and S. Djukanović, "Architecting an IoT-enabled platform for precision agriculture and ecological monitoring: A case study," *Computers and Electronics in Agriculture*, vol. 140, pp. 255–265, 2017.

[20] W. Hou, W. Li, L. Guo, Y. Sun, and X. Cai, "Recycling Edge Devices in Sustainable Internet of Things Networks," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1696–1706, Oct 2017.

[21] M. Tomoskozi, P. Seeling, P. Ekler, and F. H. P. Fitzek, "Applying Robust Header Compression Version 2 for UDP and RTP Broadcasting with Field Constraints," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, June 2017, pp. 1–5.

[22] . Kolozali, D. Puschmann, M. Bermudez-Edo, and P. Barnaghi, "On the Effect of Adaptive and Nonadaptive Analysis of Time-Series Sensory Data," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1084–1098, Dec 2016.

[23] J. Hwang, T. He, and Y. Kim, "Exploring In-Situ Sensing Irregularity in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 547–561, April 2010.

[24] N. Romano, "Soil moisture at local scale: Measurements and simulations," *Journal of Hydrology*, vol. 516, pp. 6–20, 2014.

[25] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 116.

[26] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013.

[27] (2011, Aug) Comparing-low-power-wireless-technologies. DIgikey Electronics. [Online]. Available: http://www.digikey.com/en/articles/techzone/2011/aug/comparing-low-power-wireless-technologies