Distributed-neuron-network based Machine Learning on Smart-gateway Network towards Real-time Indoor Data Analytics

Hantao Huang, Yuehua Cai and Hao Yu
School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore 639798
Email:haoyu@ntu.edu.sg

***Abstract*— Indoor data analytics is one typical example of ambient intelligence with behaviour or feature extraction from environmental data. It can be utilized to help improve comfort level in building and room for occupants. To address dynamic ambient change in a large-scaled space, real-time and distributed data analytics is required on sensor (or gateway) network, which however has limited computing resources. This paper proposes a computationally efficient data analytics by distributed-neuron-network (DNN) based machine learning with application for indoor positioning. It is based on one incremental L_2-norm based solver for learning collected WiFi-data at each gateway and is further fused for all gateways in the network to determine the location. Experimental results show that with multiple distributed gateways running in parallel, the proposed algorithm can achieve 50x and 38x speedup during data testing and training time respectively with comparable positioning accuracy, when compared to traditional support vector machine (SVM) method.**

I. Introduction

Ambient intelligence (AmI) is to introduce machine intelligence within environment in response to the presence of human beings [1], [2]. Indoor AmI can perform behaviour or feature extraction from power, temperature, and lighting data, and hence further help improve comfort level for human occupants in building. The current indoor AmI cannot address dynamic ambient change with a real-time response under emergency because processing backend in cloud takes latency. Moreover, scalability and robustness have to be addressed in a distributed solution for a large space. As such, a real-time and distributed data analytics directly on sensor (or gateway) network is needed. However, there is always limited computing resource on the sensor (or gateway) network. A computationally efficient data analytics (or machine learning algorithm) is thereby required for indoor AmI such as indoor positioning based on WiFi-data [3], [4].

Many WiFi-data based positioning systems have been developed recently for indoor positioning based on received signal strength indicator (RSSI) [5]. As the RSSI parameter can show large dynamic change under environmental change (such as obstacles)[5], [6], the traditional machine-learning based WiFi-data analytic algorithms such as K-nearest neighbourhood (KNN), neural network, and support vector machine (SVM) all are centralized with large latency to adapt to the environmental change because the training has high computational complexity [7], [8] which will introduce large latency and also cannot be adopted on the sensor network directly.

In this paper, we introduce a low computational complexity machine-learning algorithm that can perform WiFi-data analytics for positioning on smart gateway network. A distributed-neuron-network (DNN) machine learning algorithm is introduced with the maximum posteriori probability based soft-voting. Each gateway utilizes the received signal strength indicator (RSSI) and corresponding location labels as input to train a single-layer-neuron network. To avoid the high complexity of backward propagation in training, we have developed a new training algorithm: input weight is randomly generated and output weight is directly calculated by least square. By adopting incremental Cholesky factorization method, complexity of least square is greatly reduced when performing the training on a number of smart gateways. In addition, the maximum posteriori probability based soft-voting is developed to improve the accuracy when fusing data from many smart gateways.

Experimental results show that the proposed distributed-neuron-network (DNN) based machine learning with soft-voting can be successfully deployed on the smart gateway network for indoor positioning, which has 38x improvement in training time compared to conventional SVM based method; and also achieves an improvement of 50x testing time compared to SVM. The accuracy is also greatly improved by soft-voting technique.

The rest of this paper is organized as follows. The system overview and problem formulation are introduced in Section II. In Section III, AmI indoor positioning system (IPS) using machine learning is elaborated in details. Then in Section IV, the IPS is further elaborated within the distributed-neuron-network (DNN) with the maximum posteriori probability based soft-voting. Experiment results are presented to validate the proposed algorithm in Section V with conclusion drawn in Section VI.

This work is sponsored by Singapore-JTC fund, Huawei Shannon Lab fund and JIP support from MediaTek Singapore.

978-3-9815370-6-2/DATE16/

II. Ambient Intelligence System Description

A. Indoor Positioning for Smart Home

Smart Home Management System (SHMS) is an ambient intelligence system built for residents to benefit from automation technology. By collecting environmental data including temperature, humidity and human activities, a system can react towards residents' best experience [1], [9]. Fig. 1 depicts the basic components and working strategies in our SHMS testbed:

- Smart gateways to be the control center, harboring the ability in storage and computation. Our smart gateway will be BeagleBoard-xM.
- Smart sensors to collect environmental information on light intensity, temperature, humidity, and occupancy.
- Smart sockets to collect current information of home appliances.
- Smart devices with GUI to interact with users; residents have access to environmental information and can control home appliances through a smart phone or tablet.

To ensure high quality performance of SHMS, a robust indoor positioning system (IPS) is indispensable because knowledge about occupants of a building and their movements is essential [10]. Applications can include the scenarios when a resident comes back and enters his room, SHMS automatically powers on air conditioner, heater, humidifier and sets the indoor environment to suit the fitness condition; when nobody is in the house, the system turns off all appliances except for fridge and security system for energy saving issue.

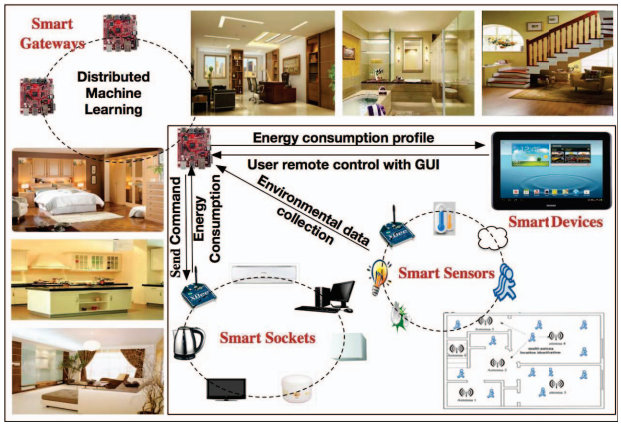

Fig. 1: The overview of smart home management system

B. Indoor Positioning Infrastructure

For the physical infrastructure, an indoor positioning system (IPS) by WiFi-data consists of at least two hardware components: a transmitter unit and a measuring unit. Here we use smart gateways to collect WiFi signal emitted from other smart devices (phone, pad) of moving occupants inside the building. The IPS determines the positioning with WiFi-data analyzed from the smart gateway network [11].

The central unit in SHMS is BeagleBoard-xM as shown in Fig. 2b, which is also utilized in our positioning systems. In Fig. 2c, TL-WN722N wireless adapter is our Wi-Fi sensor for wireless signals capturing. BeagleBoard-xM runs Ubuntu 14.04 LTS with all the processing done on board, including data storage, Wi-Fi packet parsing, and positioning algorithm computation. TL-WN722N works in monitor mode, capturing packets according to IEEE 802.11. They are connected with a USB 2.0 port on BeagleBoard-xM.

As depicted in Fig. 2d, Wi-Fi packet contains a header field (30 bytes in length), which contains information about Management and Control Address (MAC). This MAC address is unique to identify the device where the packet came from. Another useful header, which is added to the Wi-Fi packets when capturing frames, is the radio-tap header, which is added by the capturing device (TL-WN722N). This radio-tap header contains information about the RSSI, which reflects the information of distance [3]. With MAC address to identify objects and RSSI values to describe distance information, indoor positioning can be performed.

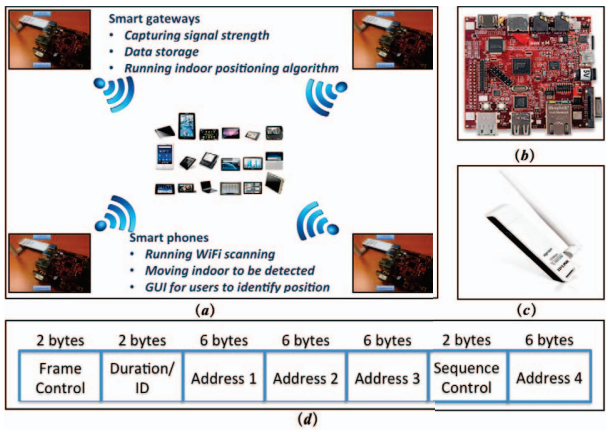

Fig. 2: (a) Indoor positioning components overview (b) BeagleBoard xM (c) TL-WN722N (d) MAC frame format in Wi-Fi header field

C. Machine Learning Based Indoor Positioning

Existing positioning systems can be classified into symbolic and geometric models. In a symbolic model, all objects are represented as symbols and referred by names or labels; in a geometric model, the physical space is represented as the Euclidean space and objects are described by the set of coordinates in the Euclidean space. The coordinate system is not really suitable for indoor environment since each indoor environment has its special layout to describe its position [12]. Therefore, we adopt a symbolic model and a classification algorithm based on machine-learning algorithm is required.

D. Problem Formulation

As mentioned in Section I, the primary objective is to locate the target as accurate as possible considering the scalability and complexity.

Objective 1: Improve the accuracy of positioning subject to the defined area.

$$\begin{aligned} &\min\ e = \sqrt{(x_e - x_0)^2 + (y_e - y_0)^2} \\ &s.t.\ label(x_e, y_e) \in \mathbf{T} \end{aligned} \tag{1}$$

where (x_e, y_e) is the system estimated position belongs to the positioning set $\mathbf{T}$ and (x_0, y_0) is the real location coordinates. Therefore, a symbolic model based positioning problem can be solved using training set Ω to develop neuron network.

$$\Omega = \{(s_i, t_i), i = 1, \cdots, N, s_i \in R^n,\ t_i \in \mathbf{T}\} \tag{2}$$

where N represents number of datasets and n is the number of smart gateways, which can be viewed as the dimension of the signal strength space. s_i is the vector containing RSSI values collected in ith dataset, $t_i \in \{-1, 1\}$ is the label assigned by the supervisor. Note that $\mathbf{T}$ labels the physical position. The more labels are used, the more accurate the positioning service is.

Objective 2: Reduce the training time on the distributed-neuron-network on Hardware BeagleBaord-xM. To distribute training task on gateways with n number, the average training time should be minimized to reflect the reduced complexity on such gateway system.

$$\min \frac{1}{n} \sum_{i=1}^{n} t_{train,i} \quad s.t.\ e < \epsilon \tag{3}$$

where e is the training error and ϵ is the tolerable maximum error. To make it clear, a list of variables and their descriptions are shown in Table I

III. Machine Learning Algorithm on Gateway

A. Single-hidden-layer neuron Network

Our neuron network with two sub-systems is shown as Fig. 3, which is inspired by ELM and compressed sensing [7], [13]. Unlike previous work [14], the input weight is only connecting nearyby hidden nodes. The input weight in our proposed neuron network is connected to every hidden node and is randomly generated independent of training data. Therefore, only the output weight is calculated from the training process. Assume there are N arbitrary distinct training samples $\mathbf{X} \in \mathbf{R}^{N \times n}$ and $\mathbf{T} \in \mathbf{R}^{N \times m}$, where $\mathbf{X}$ is training data representing scaled RSSI values from each gateway and $\mathbf{T}$ is the training label indicating its position respectively. In

TABLE I: List of variables with their description

Notation	Definition
(x_e, y_e)	System estimated location coordinates
(x_0, y_o)	Real location coordinates
Ω	Training set with RSSI values and according positions
N	Training data size
m	Number of symbolic classes (labels) for positioning
n	Number of gateways (dimension of signal strength space)
L	Number of hidden nodes in neural network
$\mathbf{X} = \{x_i, ..., x_N\}$	Set of training data, scaled RSSI values $x_i \in \mathbf{R}^n$
$\mathbf{S} = \{s_i, ..., s_N\}$	Set of received RSSI values $s_i \in \mathbf{R}^n$
$\mathbf{T} = \{t_i, ..., t_N\}$	Set of symbolic classes (labels) for positioning $t_i \in \mathbf{R}^m$
$\mathbf{A} = \{a_{ij}, ..., a_{nL}\}$	Input weight for first layer in neural network
$\mathbf{B} = \{b_{ij}, ..., b_{NL}\}$	Input bias for first layer in neural network
$\mathbf{H} = \{h_{ij}, ..., h_{NL}\}$	Activation matrix from Sigmoid function
$\boldsymbol{\beta} = \{bt_{ij}, ..., bt_{Lm}\}$	Output weight for second layer in neural network
$\mathbf{Q}$	Low triangular matrix
h_L	New added column of H_L by increase L
N_{slfn}	Number of sub-systems for neural network
$P_j(c_i\|\mathbf{x})$	Posteriori probability for sub-system j to choose class c_i
$P(c_i)$	Soft-voting based posteriori probability to choose class c_i
N_{pc}, N_p	Number of correct predictions
N_p	Total number of predictions

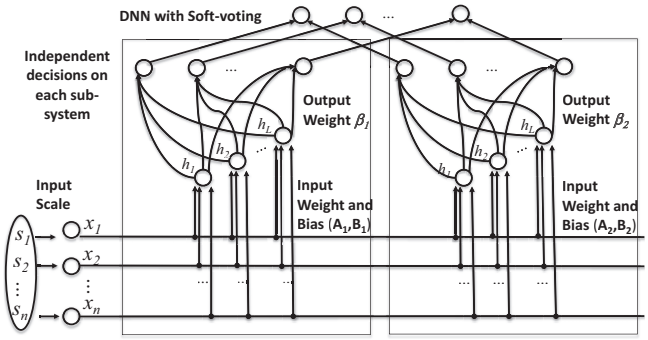

Fig. 3: Soft-voting based distributed-neuron-network with 2 sub-systems

our indoor positioning cases, the relation between the hidden neuron node and input training data is addictive as

$$\mathbf{preH} = \mathbf{XA} + \mathbf{B},\ \mathbf{H} = \frac{1}{1 + e^{-\mathbf{preH}}} \tag{4}$$

where $\mathbf{A} \in \mathbf{R}^{n \times L}$ and $\mathbf{B} \in \mathbf{R}^{N \times L}$. $\mathbf{A}$ and $\mathbf{B}$ are randomly generated input weight and bias formed by a_{ij} and b_{ij} between $[-1, 1]$. $\mathbf{H} \in \mathbf{R}^{N \times L}$ is the result from sigmoid function for activation. In general cases, the number of training data is much larger than the number of hidden neuron nodes (i.e. $N > L$), to find the output weight $\boldsymbol{\beta}$ is an overdetermined system. Therefore, estimate the output weight is equivalent to minimize $||\mathbf{T} - \mathbf{H}\boldsymbol{\beta}||$, the general solution can be found as

$$\boldsymbol{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T},\ \mathbf{H} \in \mathbf{R}^{N \times L} \tag{5}$$

where $\boldsymbol{\beta} \in \mathbf{R}^{L \times m}$ and m is the number of symbolic classes. $(\mathbf{H}^T \times \mathbf{H})^{-1}$ exits for full column rank of $\mathbf{H}$ [7]. However, such method is computational intensive. Moreover, as the number of hidden neuron nodes can not be explicit from the training data to have small training error, [15] suggests to increase the number hidden neuron node L during the training stage, which will reduce the training error but at the cost of increasing computational cost and required memory for neuron network. Therefore, an incremental solution for (5) is needed to adjust the number of hidden node L with low complexity. The algorithm of single hidden layer forward neuron network is summarized in Algorithm 1.

B. Incremental Least-square Solver

The key difficulty for solving training problem is the least square problem of minimizing $||\mathbf{T} - \mathbf{H}\boldsymbol{\beta}||$. This could be solved by using SVD, QR and Cholesky decomposition. The computational cost of SVD, QR and Cholesky decomposition is $O(4NL^2 - \frac{4}{3}L^3)$, $O(2NL^2 - \frac{2}{3}L^3)$ and $O(\frac{1}{3}L^3)$ respectively [16]. Therefore, we use Cholesky decomposition to solve the least square problem. Moreover, its incremental and symmetric property reduces the computational cost and save half memory required [16]. Here, we use H_L to represent the matrix with L number of hidden neuron nodes $(L < N)$, which decomposes the symmetric positive definite matrix $\mathbf{H}^T\mathbf{H}$ into

$$\mathbf{H}_L^T\mathbf{H}_L = \mathbf{Q}_L\mathbf{Q}_L^T \tag{6}$$

Algorithm 1 Learning Algorithm for Single Layer Network

Input: Training Set $(x_i, t_i), x_i \in \mathbf{R}^n, t_i \in \mathbf{R}^m, i = 1, ...N$, activation function $H(a_{ij}, b_{ij}, x_i)$, maximum number of hidden neuron node L_{max} and accepted training error ϵ.

Output: Neuron Network output weight $\boldsymbol{\beta}$

1: Randomly assign hidden-node parameters (a_{ij}, b_{ij}), $a_{ij} \in \mathbf{A}$, $b_{ij} \in \mathbf{B}$

2: Calculate the hidden-layer pre-output matrix $\mathbf{H}$ $\mathbf{preH} = \mathbf{XA} + \mathbf{B}$, $\mathbf{H} = 1/(1 + e^{-\mathbf{preH}})$

3: Calculate the output weight $\boldsymbol{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}$

4: Calculate the training error $error$ $error = ||\mathbf{T} - \mathbf{H}\boldsymbol{\beta}||$

5: IF ($L \leq L_{max}$ and $e > \epsilon$) Increase number of hidden node $L = L + 1$, repeat from Step 1

6: ENDIF

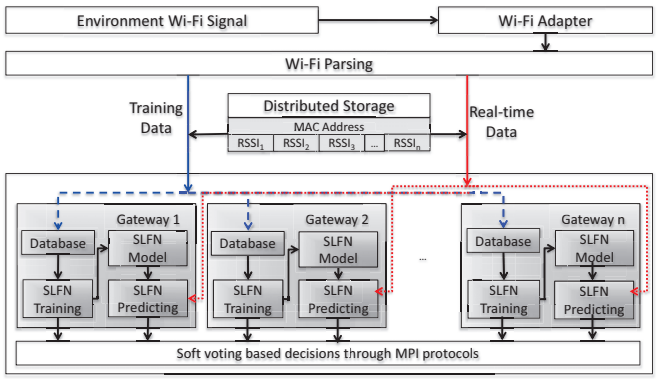

Fig. 4: Working flow of distributed-neuron-network indoor positioning system

where $\mathbf{Q}_L$ is a low triangular matrix and T represents transpose operation of the matrix.

$$\begin{aligned}\mathbf{H}_L^T\mathbf{H}_L &= \begin{bmatrix}\mathbf{H}_{L-1} & h_L\end{bmatrix}^T\begin{bmatrix}\mathbf{H}_{L-1} & h_L\end{bmatrix} \\ &= \begin{pmatrix}\mathbf{H}_{L-1}^T\mathbf{H}_{L-1} & \mathbf{v}_L \\ \mathbf{v}_L^T & g\end{pmatrix}\end{aligned} \tag{7}$$

where h_L is the new added column by increasing the size of L, which can be calculated from (4). The Cholesky matrix can be expressed as

$$\begin{aligned}&\mathbf{Q}_L\mathbf{Q}_L^T \\ &= \begin{pmatrix}\mathbf{Q}_{L-1} & \mathbf{0} \\ \mathbf{z}_L^T & p\end{pmatrix}\begin{pmatrix}\mathbf{Q}_{L-1}^T & \mathbf{z}_L \\ \mathbf{0} & p\end{pmatrix}\end{aligned} \tag{8}$$

As a result, we can easily calculate the $\mathbf{z}_L$ and scalar p for Cholesky factorization as

$$\mathbf{Q}_{L-1}\mathbf{z}_L = \mathbf{v}_L, \; p = \sqrt{g - \mathbf{z}_L^T\mathbf{z}_L} \tag{9}$$

where $\mathbf{Q}_{L-1}$ is the previous Cholesky decomposition result and $\mathbf{v}_L$ is known from (7), which means we can continue use previous factorization result and update only according part. Algorithm 2 gives details on each step since $l \geq 2$. Please note when $l = 1$, Q_1 is a scaler and equals to $\sqrt{H_1^T H_1}$. Such method will greatly reduce computational cost and allow the online training on smart gateway for positioning.

IV. Distributed-neuron-network with Soft-voting

A. Distributed-neuron-network for Indoor Positioning

The working flow of the distributed-neuron-network (DNN) is described as Fig. 4. Environmental Wi-Fi signal is received from the Wi-Fi Adapter and through Wi-Fi parsing the data with MAC address and RSSI of each Wi-Fi adapter is stored. Such data is sent to gateway for training with label first. Please note that the training process is on the gateway. As we mentioned in Section III, a single layer forward network (SLFN) is trained. A small data storage is required to store trained weight for the network. In the real time application, the same format data will be collected and sent into the well trained network to locate its position. In Fig. 4, the block for soft-voting is through message passing interface (MPI) protocols to collect all the testing result from each SLFN and soft-voting is processed in the central gateway. Note that n gateways together can form one or several SLFNs based on the accuracy requirement.

B. Soft-voting

As we have discussed in section III, the input weight and bias $\mathbf{A}, \mathbf{B}$ are randomly generated, which strongly supports that each SLFN is an independent expert for indoor positioning. Each gateway will generate posteriori class probabilities $P_j(c_i|\mathbf{x}), i = 1, 2, ..., m, j = 1, 2, ..., N_{slfn}$, where $\mathbf{x}$ is the received data, m is the number of classes and N_{slfn} is the number of sub-systems for single layer network deployed on smart gateway. During the testing process, the output of single layer forward network (SLFN) will be a set of values $y_i, i = 1, 2, ..., m$. Usually, the maximum y_i is selected to represent its class i. However, in our case, we scale the training and testing input between $[-1, 1]$ and target labels are also formed using a set of $[-1, -1, ...1..., -1]$, where the only 1 represents its class and the target label has length m. The posteriori probability is estimated as

$$P_j(c_i|\mathbf{x}) = (y_i + 1)/2, \; j = 1, 2, ..., N_{slfn} \tag{10}$$

A loosely stated objective is to combine the posteriori of all sub-systems to make more accurate decisions for the incoming data $\mathbf{x}$. Under such case, information theory suggests to use a

Algorithm 2 Incremental L_2 norm solution

Input: Activation matrix $\mathbf{H}_L$, target matrix $\mathbf{T}$ and number of hidden nodes L

Output: Neuron Network output weight $\boldsymbol{\beta}$

1: FOR $l = 2 : L$

2: Calculate new added column, $\mathbf{v}_l = \mathbf{H}_{l-1}^T h_l$ $g = h_l^T * h_l$

3: Calculate updated Cholesky matrix $\mathbf{z}_L = \mathbf{Q}_{L-1}^{-1}\mathbf{v}_L$, $p = \sqrt{g - \mathbf{z}_L^T\mathbf{z}_L}$

4: Form new Cholesky Matrix $\mathbf{Q}_L = \begin{pmatrix}\mathbf{Q}_{L-1} & 0 \\ \mathbf{z}_L^T & p\end{pmatrix}$

5: END FOR

6: Calculate output weight using forward and backward substitution $\mathbf{Q}_L\mathbf{Q}_L^T\boldsymbol{\beta} = \mathbf{H}_L^T\mathbf{T}$

cross entropy (Kullback-Leibler distance) criterion [17], where we may have two possible ways to combine the decisions (Geometric average rule and Arithmetic average rule). The geometric average estimates can be calculated as

$$P(c_i) = \prod_{j=1}^{N_{slfn}} P_j(c_i|\mathbf{x}),\ i = 1, 2, ...m \tag{11}$$

and the arithmetic average estimate is shown as

$$P(c_i) = \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(c_i|\mathbf{x}),\ i = 1, 2, ...m \tag{12}$$

where $P(c_i)$ is the posteriori probability to choose class c_i and will select the maximum posteriori $P(c_i)$ for both cases. In this paper, we use arithmetic average as soft-voting of each gateway since [17] indicates that geometric average rule works poorly when the posteriori probability is very low. This may happen when the object to locate is far away from one gateway and its RSSI is small with low accuracy of positioning. The final decision is processed at the central gateway to collect the voting value from each sub-systems on other gateways. Such soft-voting will utilize the confidence of each sub-system and avoid the pre-request that each sub-system maintains accuracy of more than 50 % for hard-voting.

TABLE II: Experimental set-up parameters

Parameter	Value
Traing Date Size	18056
Testing Date Size	2000
Data Dimension	5
Number of labels	48
No. of Gateway	5
Testing area	$80m^2$

V. Experimental Results

A. Experiment Setup

Indoor test-bed environment for positioning is presented in Fig. 5, with total area being about 80 m^2 (8 m at width and 10 m at length) separated into 48 regular blocks, each block represents a research cubicle. 5 gateways, with 4 at 4 corners of the map, 1 in the center of the map, are set up for experiment. As shown in Fig. 5, 5 gateways will receive different RSSI values as the object moving. To quantify our environment setting, here the positioning accuracy is defined as r, representing radius of target area. It is generated from $S = \pi r^2$, where S is the square of the whole possible positioning area. Besides, positioning precision is defined as the probability that the targets are correctly positioned within certain accuracy. The definition is as follow:

$$Precision = \frac{N_{pc}}{N_p} \tag{13}$$

where N_{pc} is the number of correct predictions and N_p is the number of total predictions. The summary for the experiment set-up is shown in Table II

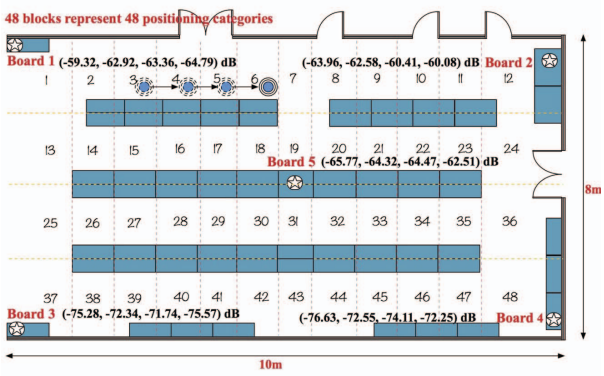

Fig. 5: An example of building floor with position tracking

B. Real-time Indoor Positioning Results

The result of the trained neuron forward network is shown as Fig. 6 and Fig. 7. The training time can be greatly reduced by using incremental Cholesky decomposition. This is due to the reduction of least square complexity, which is the limitation for the training process. As shown in Fig. 6, training time maintains almost constant with increasing number of neuron nodes when the previous training results are available. Fig. 7 also shows the increasing accuracy under different positioning scales from 0.73m to 4.57m. It also shows that increasing the number of neuron nodes will increase the performance to certain accuracy and maintains almost flat at larger number of neuron nodes.

TABLE III: Comparison table with previous works

System/Solution	Precision
Proposed DNN	58% within 1.5m, 74% within 2.2m and 87% within 3.64m
Proposed SV-DNN	62.5% within 1.5m, 79% within 2.2m and 91.2% within 3.64m
Microsoft RADAR [18]	50% within 2.5m and 90% within 5.9m
DIT [19]	90% within 5.12m for SVM; 90% within 5.40m for MLP
Ekahau [20]	5 to 50m accuracy (indoors)
SVM	63% within 1.5m, 80% within 2.2m and 92.6% within 3.64m

C. Performance Comparison

In Table III, we can see that although single layer network cannot perform better than SVM but it outperforms other positioning algorithms proposed in [18], [19], [20]. Moreover, by using maximum posteriori probability based soft-voting, SV-DNN can be very close to the accuracy of SVM. Table IV shows the detailed comparisons between proposed DNN positioning algorithm with SVM. Please note that the time reported is the total time for training data size 18056 and testing data size 2000. It shows more than 120x training

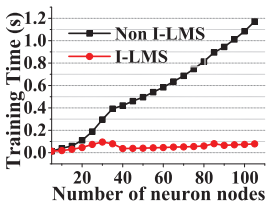

Fig. 6: Training time for SLFN by Incremental Cholsky decomposition

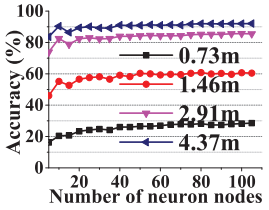

Fig. 7: Testing Accuracy under different positioning scale

TABLE IV: Performance precision with variations on proposed DNN with soft-voting

	Testing time (s)	Training time (s)	0.73m	1.46m	2.19m	2.91m	3.64m	4.37m	5.1m	No. of Nodes
SVM	9.7580	128.61	31.89%	63.26%	80.25%	88.54%	92.58%	94.15%	94.71%	N.A.
Acc. &Var.			0.530	0.324	0.394	0.598	0.536	0.264	0.0975	
DNN	0.1805	1.065	23.94%	57.78%	74.14%	82.61%	87.22%	90.14%	91.38%	100
Acc. &Var.			1.2153	0.0321	0.1357	0.2849	0.0393	0.0797	0.0530	
SV-DNN (2)	0.1874	2.171	29.36%	61.23%	77.20%	86.24%	90.25%	92.19%	93.14%	2 Sub-systems
Acc. &Var.			0.358	0.937	0.526	0.517	0.173	0.173	0.124	Each 100
SV-DNN (3)	0.1962	3.347	30.52%	62.50%	79.15%	87.88%	91.20%	92.92%	94.08%	3 Sub-systems
Acc. &Var.			0.325	1.952	0.884	1.245	0.730	0.409	0.293	Each 100

time improvement and more than 54x testing time saving for proposed SLFN with 1 sub-network comparing to SVM. Even adding soft-voting with 3 sub-networks, 50x and 38x improvement in testing and training time respectively can be achieved. Please note that for fair training and testing time comparison, all the time is recorded using Ubuntu 14.04 LTS system with core 3.2GHz and 8GB RAM. Variances of the accuracy is also achieved by 5 repetitions of experiments and the reported results are the average values. We find that the stability of proposed DNN is comparable to SVM. Moreover, the testing and training time does not increase significantly with new added subnetworks. Please note that SVM is mainly limited by its training complexity and binary nature where one-against-one strategy is used to ensure accuracy with a cost of building $m(m-1)/2$ classifier and m is the number of classes. Fig. 8 shows the error zone of proposed SV-DNN.

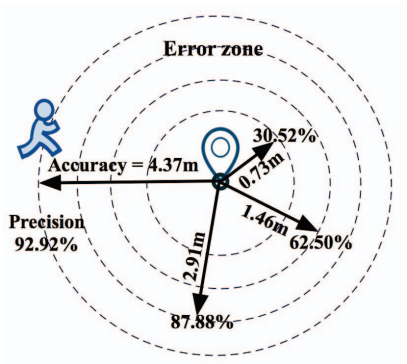

Fig. 8: Error Zone and accuracy for indoor positioning

VI. Conclusion

To address dynamic ambient change in a large-scaled space, real-time and distributed data analytics is required on gateway network, which however has limited computing resources. This paper proposes a computationally efficient data analytics by distributed-neuron-network (DNN) based machine learning with application for indoor positioning. It is based on one incremental L_2-norm based solver for learning collected WiFi-data at each gateway and is further fused for all gateways in the network to determine the location. Experimental results show that with 5 distributed gateways running in parallel for a $80m^2$ space, the proposed algorithm can achieve 50x and 38x improvement on testing and training time respectively when compared to support vector machine based data analytics with comparable positioning precision.

References

[1] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277–298, 2009.

[2] M. Altini, D. Brunelli, E. Farella, and L. Benini, "Bluetooth indoor localization with multiple neural networks," in *IEEE ISWPC*, 2010.

[3] M. O. Ergin, V. Handziski, and A. Wolisz, "Node sequence discovery in wireless sensor networks," in *IEEE DCOSS*, 2013.

[4] J. Xu, W. Liu, F. Lang, Y. Zhang, and C. Wang, "Distance measurement model based on RSSI in WSN," *Wireless Sensor Network*, vol. 2, no. 08, p. 606, 2010.

[5] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 6, pp. 1067–1080, 2007.

[6] J. Torres-Solis, T. H. Falk, and T. Chau, *A review of indoor localization technologies: towards navigational assistance for topographical disorientation*. INTECH Publisher, 2010.

[7] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[8] Y. Cai and H. Yu, "Indoor positioning by distributed machine-learning based data analytics on smart gateway network," in *IEEE IPIN*, 2015.

[9] S. Nikoletseas, M. Rapti, T. P. Raptis, and K. Veroutis, "Decentralizing and adding portability to an iot test-bed through smartphones," in *IEEE DCOSS*, 2014.

[10] S. Helal, B. Winkler, C. Lee, Y. Kaddoura, L. Ran, C. Giraldo, S. Kuchibhotla, and W. Mann, "Enabling location-aware pervasive computing applications for the elderly," in *IEEE PerCom*, 2003.

[11] C. Drane, M. Macnaughtan, and C. Scott, "Positioning GSM telephones," *Communications Magazine, IEEE*, vol. 36, no. 4, pp. 46–54, 1998.

[12] D. Li and D. L. Lee, "A topology-based semantic location model for indoor applications," in *ACM SIGSPATIAL*, 2008.

[13] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[14] R. Battiti, A. Villani, and T. Le Nhat, "Neural network models for intelligent networks: deriving the location from signal patterns," *Proceedings of AINS*, 2002.

[15] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *Neural Networks, IEEE Transactions on*, vol. 20, no. 8, pp. 1352–1357, 2009.

[16] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[17] D. J. Miller and L. Yan, "Critic-driven ensemble classification," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2833–2844, 1999.

[18] P. Bahl and V. N. Padmanabhan, "Radar: An in-building RF-based user location and tracking system," in *IEEE INFOCOM*, 2000.

[19] M. Brunato and R. Battiti, "Statistical learning theory for location fingerprinting in wireless LANs," *Computer Networks*, vol. 47, no. 6, pp. 825–845, 2005.

[20] I. Ekahau. (2015). [Online]. Available: http://www.test.org/doe/