Enabling Edge Intelligence for Activity Recognition in Smart Homes (Invited Paper)

Shaojun Zhang[1], Wei Li[1], Yongwei Wu[2], Paul Watson[3], Albert Y. Zomaya[1]
[1]Centre for Distributed and High Performance Computing, School of Information Technologies, The University of Sydney, Australia
[2]Department of Computer Science and Technology, Tsinghua University, China
[3]Digital Institute, Newcastle University, United Kingdom
szha6955@uni.sydney.edu.au, weiwilson.li@sydney.edu.au, wuyw@tsinghua.edu.cn,
paul.watson@newcastle.ac.uk, albert.zomaya@sydney.edu.au

***Abstract*—In recent years, Edge computing has emerged as a new paradigm that can reduce communication delays over the Internet by moving computation power from far-end cloud servers to be closer to data sources. It is natural to shift the design of cloud-based IoT applications to Edge-based ones. Activity recognition in smart homes is one of the IoT applications that can benefit significantly from such a shift. In this work, we propose an Edge-based solution for addressing the activity recognition problem in smart homes from multiple perspectives, including: architecture, algorithm design and system implementation. First, the Edge computing architecture is introduced and several critical management tasks are also investigated. Second, a realization of the Edge computing system is presented by using open source software and low-cost hardware. The consistency and scalability of running jobs on Edge devices are also addressed in our approach. Last, we propose a convolutional neural network model to perform activity recognition tasks on Edge devices. Preliminary experiments are conducted to compare our model with existing machine learning methods, and the results demonstrate that the performance of our model is promising.**

***Index Terms*—Edge computing, Edge intelligence, smart home, activity recognition, data parallelism, model parallelism, convolutional neural network, deep learning**

I. Introduction

With the prevalence of smart sensors, the number of Internet of Things (IoT) [1] devices has been growing rapidly in recent years. Researchers from both academia and industry are working on the development of smart objects with IoT devices. Many popular IoT applications have been developed by understanding how to interpret sensor readings to interact effectively with the environment. One such type of IoT application is to provide smart living environment support from heterogeneous sensors for various human-centric purposes, ranging from the development of smart homes [2] to the design of smart cities [3], [4], [5]. It is believed that such IoT applications have unlimited possibilities that can reshape our daily lives.

Though developing rapidly, there are still many open research issues in realizing the smart living environment. One of the core research problems is activity learning, which generally refers to the learning and understanding of the observed human activities in an urban environment [6]. According to [6], activity learning technology is recognized as a key component for multiple real-world problems, such as surveillance and security systems. Activity learning technology focuses on recognizing activity types from the behaviors performed by an individual interacting with the environment, which would always be composite and composed of a series of actions. To take cooking breakfast as an example, one needs to walk to the kitchen, open the fridge, take out the milk and bread, put them into a microwave oven and heat up them. The high-level cooking activity is a combination of the low-level actions. Furthermore, in the real world, some such actions are in a specific order, while others could be performed randomly.

With respect to the development of activity recognition, existing solutions can be unequivocally classified into two types: the video-based approach and the sensor-based approach. In past years, the video-based approach witnesses the rapid development of the deep learning algorithms. Deep-learning generative models [7] are proposed to classify the activity from the video stream, such as spatial-temporal networks, multiple stream networks, deep generative networks, and temporal coherency networks. They have been widely used in surveillance systems, which could extract human actions and label the high-level activities from the video frames. However, for smart assistance applications used in households, the video-based approach would face use restrictions due to privacy and security concerns raised by the residents.

As an alternative and more reliable solution, a sensor-based approach could certainly help to keep the privacy of residents as well as providing the data necessary for the recognition of different actions. Based on the installation location of sensors, the sensor-based approach can be further categorized into ambient sensor solution and wearable sensor solution. For the ambient sensor solution, the sensors are placed in the home rather than attached to residents. These sensors are not customized for specific people, but for specific types of actions. Several common sensors provide such solutions, including but not limited to: passive infrared sensor, magnetic door sensor, temperature sensor, light sensor, humidity sensor, pressure sensor and global positioning system (GPS) sensor [6]. For example, a motion sensor is a kind of infrared sensor which provides readings when someone moves into its cover-

age area. However, wearable sensors are always placed on the clothes or body of the resident. Examples of such sensors are accelerometer, gyroscope, and magnetometer. They can detect the motion of the body and record the respective physical quantitative change. Both ambient sensor-based and wearable sensor-based approaches have their pros and cons. For smart home applications, the ambient sensor-based approach is more popular since it has little need for deploying new sensors and adjusting the existing sensor when the number of the residents changes. As a result, we focus on the ambient sensors based approach.

To conduct activity recognition in smart home applications, the most common solution is to adopt cloud computing. Using such a method, the end user needs to submit the collected data from sensors to one or more far-end datacenter(s) for further processing, and then the result will be sent back. There are two critical issues that this type of method has to address. First, this method often encounters uncertain data transmission delays over the Internet and limited throughput caused by the network bandwidth used in households. The other is the privacy concerns due to the latent exposure of private sensor data on public servers. To address these issues, researchers advocate a new solution: Edge computing for smart home applications. Edge computing is an emerging distributed computing paradigm, which moves the computation power from the datacenter to the edge of the network. To do so, data processing would be conducted on Edge devices deployed in or near the households, which has good potential to resolve the above two critical issues caused by using cloud computing.

It is not a trivial task to develop a home-based edge system for activity recognition with high accuracy because of the diversity of users' behaviors and the performance of the hardware available to residents. In this work, we propose our preliminary design and prospective trials towards addressing this problem, from both system and algorithm perspective. The contributions of this work are as follows:

- We propose an Edge-based computing paradigm for smart home applications, which is an open avenue of research and opportunity to bring IoT and artificial intelligence together. Meanwhile, it significantly reduces the communication cost for sending the collected data from IoT devices to the far-end cloud servers.
- We develop a consistent, scalable, easy-to-deploy and low cost Edge computing platform for smart home applications with the focus on activity recognition.
- We design a convolutional neural network for activity recognition, the precision of which outperforms the existing works.

The remainder of the paper is organized as follows. Section 2 reviews related works in the field. Then, in Section 3, we propose our system architecture for smart home applications, and introduce multiple key technologies for the implementation of the Edge system. Section 4 provides the details of the algorithm for activity recognition, including the data processing and the convolutional neural network model. In Section 5, we present the experiments and the results. Section 6 concludes the paper and outlines future work.

II. Related work

In this section, we introduce the related work on activity recognition and edge computing in recent years from three different perspectives, namely, sensor systems for human activity surveillance and data collection, machine learning algorithms for activity recognition, and Edge-based solutions used for smart home applications.

A. Ambient Sensing Systems for Human Activity surveillance

The existing smart sensing systems generally serve three functions, namely data collection, activity labeling or annotating, and activity recognition.

In [8], the researchers proposed a ubiquitous sensing system to detect and recognize human activities. They installed the "tape on and forget" devices in the residential environment to measure changes in the state of an object in the home. In terms of three functions for the smart sensing system, it incorporates the following components: the sensors to collect raw binary data, the context-aware experience sampling tool (EMS) to label the residential activities manually, and supervised classification algorithms to recognize activities.

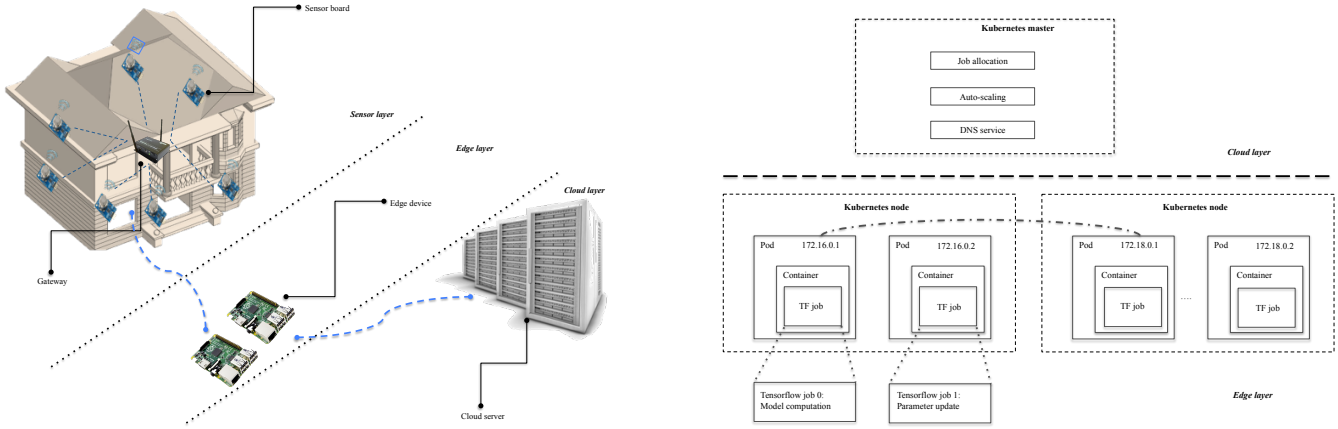
In [9], the authors applied both ambient sensors and wearable sensors to the smart home. A wireless network was constucted with the kits RFM DM 1810 to easily include and integrate new sensors into the running system. Then the annotation was conducted with the help of specific speech commands. The work also concluded several probabilistic models, such as Hidden Markov Model (HMM) and conditional random fields (CRF), to recognize the activities.

In [10], the authors presented the ARAS human activity dataset. This dataset is generated from the activities of two residents by deploying 20 sensors of seven different types connected with the ZigBee protocol in a house. The deployed sensors were able to detect 27 different activities. They collected the data from these two residents simultaneously. A basic HMM algorithm was applied to the sequential sensor events data, and some positive results were obtained.

In [11], the researchers built a series of smart home testbeds, which cover different residence types as well as environment settings. They used a sensor network and middleware connected with a Jabber-like protocol to collect and transmit data. Specifically, the testbeds had tens of sensors and enabled activity detection of multiple residents with pets. After data was collected, they employed three models on their datasets: naive Bayes classifier (NBC), HMM and CRF to recognize the activity.

B. Machine Learning for Activity Recognition

Machine learning (ML) and deep learning (DL) prove to be highly effective for addressing a wide range of recognition and classification problems. Recently, the adoption of ML algorithms [12] in the application of activity recognition also shows potential. In addition, some exemplary algorithms, such



(a) The smart home environment

(b) The container-based Edge computing architecture

Fig. 1: The Edge-based smart home architecture

as support vector machine (SVM) and convolutional neural network (CNN), are also adopted for conducting activity recognition.

In the literature, researchers tended to use the traditional machine learning methods and reasonably good results were demonstrated accordingly. In [10], the authors used a HMM model for analyzing the sequential sensor events data, and obtain an average accuracy of 61.5% and 76.2% for the activity recognition in two homes with different environment settings. [11] conducted more extensive experiments on 11 testing environments with HMM, NBC and CRF models. They also try to integrate three classifiers as a hierarchical one and employ an additional decision tree classifier on top of it. The results show that the ensemble classifier increases the accuracy of recognition for most datasets.

C. Edge Computing Paradigms

Edge computing [13] is a newly emerging distributed computing paradigm, which aims to move the computation power from the remote cloud data center to the edge of the network so as to meet the location-aware data processing, energy conservation and data privacy requirements of time-constrained IoT applications. The edge devices are very varied, ranging from a low power single-chip computer, to a powerful desktop.

Recently, machine learning and deep learning algorithms are also introduced to the Edge computing systems, which enables providing intelligence solutions for the IoT applications in a real-time or a quasi real-time manner. [14] uses the deep learning algorithm on an IoT application at the Edge for conducting video recognition. The adopted neural model is first divided into layers. The lower layers are deployed at the Edge, and the upper layers are deployed at the cloud. It also addresses the corresponding scheduling problem caused by the layer distribution, which tries to guarantee the low time cost of both the processing of and communication among deep learning tasks by carefully allocating different layers to the Edge or the cloud. There are also other works focusing on different placement strategies for the layers of a deep learning model. [15] studies the problem of how to automatically distribute the deep neural network (DNN) layers to the sensors, the Edge devices, and the cloud servers. A joint training and aggregation scheme is thus proposed to enhance sensor fusion, system fault tolerance and data privacy for DNN applications.

III. The Edge-based architecture and implementation for the smart home

In this section, we provide a detailed introduction to our edge-based smart home design from a system perspective. As shown in Figure 1(a), our system design is composed of three functional layers, the sensor layer, the Edge layer, and the cloud layer. The computation capability of the layers becomes more powerful towards the cloud. By using applicable ambient sensor systems that have been developed in the real world, we mainly focus on the design of the interactions between the Edge layer and the cloud layer in this work. Compared to existing solutions, such a design allows us to effectively reduce the amount of data transmitted from the sensor(s) to the Cloud (data centers). Please note that, the sensor system [11] used in this work does not restrict the generalization of our approach, any other sensor systems could be easily integrated into our Edge-based solution as it is or with minor adjustment.

A. The Sensor layer

In Figure 1(a), the working pattern of sensor nodes in the sensor layer at a smart home is illustrated. First, the ambient sensors collect the desired data from residents, for example, the motions in sequence, the item usage and the door movement. Once the data is gathered, the data will be pre-processed by a gateway. The gateway is capable of producing normalized sequential data sample. An example fragment of normalized data from [11] is shown in Tab. I. Each row of the data represents the details of a sensor event, namely date, time, sensor id and sensor state. Using the first row as an example, the sensor ID M08 represents a motion sensor and

its state 'ON' means that the resident enters into its coverage at a specific time of a specific day. The details of how to process the sensor data samples will be given in Section IV.

B. The Edge layer

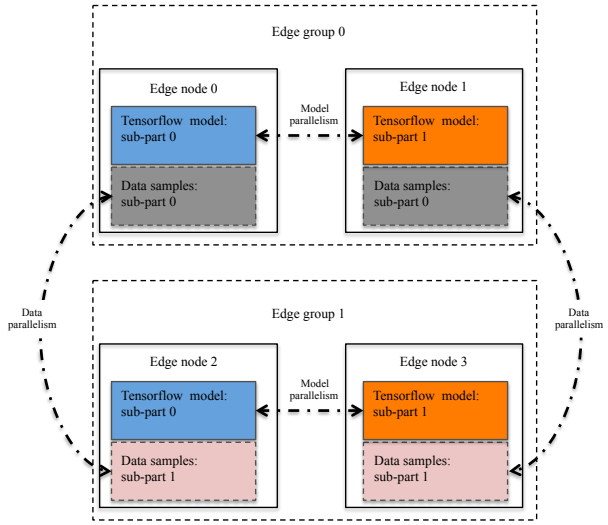


Fig. 2: The parallelism of the Edge-based Tensorflow jobs

When the data collection is completed at the sensor layer, the normalized data samples are sent to the Edge layer for further processing and the inference of the resident's activity pattern will be also completed here. The core of our approach is the design and the implementation of the Edge layer. To realize the Edge layer, we employed open source software, *docker* [16] and *kubernetes* [17] to build our own Edge computing environment, and to use *Tensorflow* to perform the deep learning related jobs.

1) The architecture and components: Docker is a Linux container framework, which enables the creation of light-weight virtualization of the computation resources and production-quality environments on Edge nodes that are often resource constraint devices. Docker has also been widely used in cloud-based applications for its advanced features on separating the applications development and execution from the underlying hardware infrastructure. By extending these features to Edge nodes, it provides us a flexible and adaptive way to interact with cloud servers and integrate with the existing solutions if needed.

Kubernetes is a container management system for managing containerized applications. It automatically deploys and scales dockers among the active Edge nodes, which could perform the resource scheduling of Edge nodes without human intervention and make sure that each application has enough resources to be used during its running. The DNS service provided by the Kubernetes master at the Cloud layer can allocate unique virtual IP addresses to different pods. With such unique IP address allocation, the pods can communicate with each other directly regardless of whether they are in the same node or at different nodes.

Tensorflow is a machine learning system that operates at large scale and in heterogeneous environments [18]. In our design, the inference phase of the deep learning and the parallelism of distributed jobs are both realized by Tensorflow.

The architecture of the container-based Edge computing system is shown in Figure 1(b). For each Edge node, it runs multiple Kubernetes pods. Within each pod, a docker is hosted. In the docker, several containerized Tensorflow jobs are running, which could be generally categorized into model computation jobs and parameter update jobs for the neural network. The model computation is key to the classification of data samples at the step of inference, and the parameter update job is designed for effective distribution of the model parameters and to realize the model parallelism.

2) The parallelism of Tensorflow jobs: To enable Tensorflow jobs to run smoothly on the Edge node still poses a number of technical challenges. The most critical one is how to run computationally complex tasks on the Edge nodes with relatively limited resources. Using the popular Edge device Raspberry Pi 3B as an example, it carries a 1.2GHz ARM quad-core CPU processor with only 512KB L2 cache, and 1GB in-built memory. In contrast, the neural network models often have millions of parameters, and the samples collected from smart home applications also need large space for the storage. Therefore, the resources available at the Edge device are not capable of holding a neural network model or conducting inference locally. In order to fully load the neural network model and all data samples into the local memory, we take advantage of the parallelism offered by Tensorflow.

In the design of Tensorflow jobs, two mechanisms of parallelism can be used to deal with the large amount of inputs and the neural network model parameters. The pattern of both types of parallelism is illustrated in Figure 2. As shown, the available Edge nodes are divided into groups, each of which contains at least two Edge nodes.

First, the data parallelism is applied among Edge groups. Data samples are split into different subparts accordingly, and then each subpart is duplicated among the Edge nodes within a group. In Figure 2, data samples are split into gray and pink parts, and then duplicated within groups. After that, the model parallelism is applied within Edge groups. As an Edge group has at least two Edge nodes, the large neural network model could also be divided into subparts and loaded accordingly. Each Edge node only holds a part of the model parameters, so as to reduce its memory cost. As shown in Figure 2, in either Edge group, the neural model is split into the blue part and the orange part, each of which runs on an Edge node.

C. The Cloud layer

In the Cloud layer, two main tasks are performed. One task is the management of Edge nodes, and the other is the training of the neural network model. Our implemented Kubernetes-based Edge nodes management module employs various load balancing strategies to perform the dynamic job allocation in order to achieve better resource utilization. Considering the computational complexity of the data training for the neural

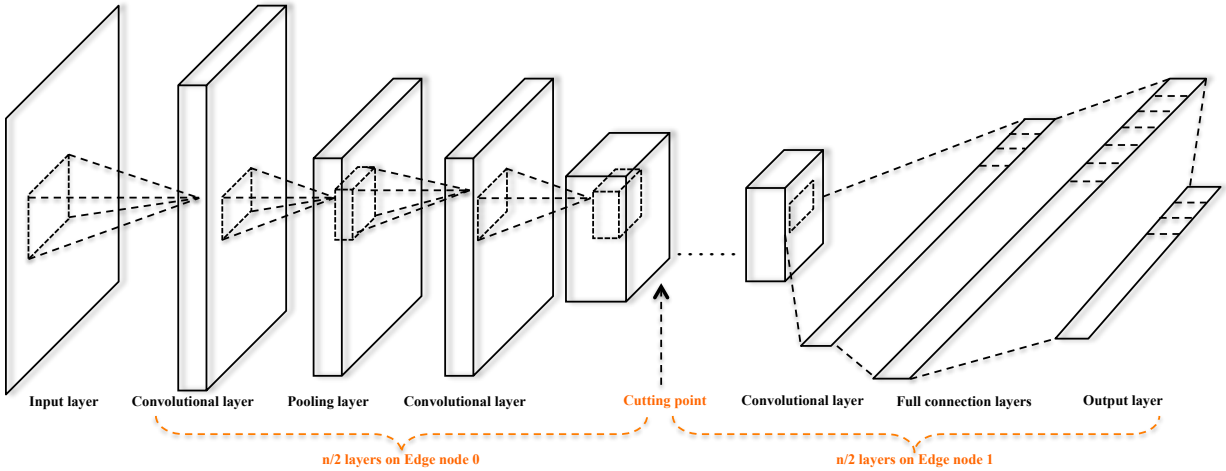


Fig. 3: Illustration of the even cutting between the convolutional neural model layers

network models, we still prefer to train the models at the Cloud and then distribute the trained model to the Edge nodes when it is needed.

Once the software deployment on the Edge layer is completed, the Cloud communicates with Edge nodes and manages pods and Tensorflow jobs allocated to them. The first goal of the management is to keep consistency of the jobs running in the pods, particularly when an Edge node is no longer live but the job is not yet finished. At all times, the Kubernetes master deployed in the cloud is responsible for detecting the liveness of the pods periodically in a thread, and move the whole pods to the next available nodes if the original ones are dead. The pseudo code of this approach is shown in Algorithm 1. Please note that A is the list containing all candidate Edge nodes, and L is the live Edge node list, which is renewed by an independent thread every few seconds. Since a pod in an Edge node is stateless, the states of the jobs in the pod are recorded in the cloud before their executions. Also, the backups of the states of the jobs are renewed every few seconds. With the help of the above mechanisms, the pod can thus run consistently.

The second goal of the management is dynamic scaling. When the number of data samples allocated to an Edge group exceeds a given threshold, the data needed to be stored can exceed the disk capacity of an Edge node. At this time, the Cloud needs to dynamically scale up the running pods by duplicating the existing pods to the next available Edge node.

After the Cloud allocates pods to the Edge nodes, it is time to train the neural network model at the cloud server. We utilize the Tensorflow framework to design and train the convolutional neural network model, which will be introduced in detail in the next section.

After the training of the neural network model is completed, the Edge node in each Edge group will fetch part of the model respectively under the rules of model parallelism. The authors in [19] show that the cutting of a neural model can significantly affect the processing time, the CPU utilities and memory costs of the allocated machines. In addition, the experiments of [19] demonstrate that the even cut strategy for a convolutional neural model is the best solution when the model is running on (near-)homogeneous devices when the above performance metrics are jointly considered.

In this work, the computation capability and hardware resources in each Edge node are almost the same. We thus employed the strategy of even cutting to divide the neural model into two parts, each of which runs on Edge nodes in a group as illustrated in Fig 3. If heterogeneous devices are used in the system, other cutting strategies can be adopted to optimize the above performance metrics for enabling model parallelism among Edge nodes.

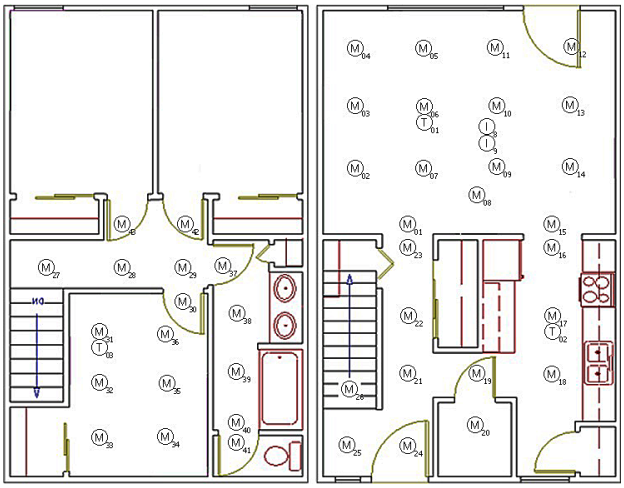


Fig. 4: The layout of sensor deployment in the smart home testbed from Washington State University [11]

IV. The Convolutional Neural Network Model for Residential Activity Recognition in Smart Homes

The convolutional neural network has been proved to be effective and efficient in resolving pattern recognition and classification problems, particularly in image processing and

Algorithm 1 The management of the consistence and dynamic scaling of Edge nodes

Thread 0: Update the live node list L of the candidate node list A

```
while true do
    for node in A do
        if node.state is live then
            L.append(node)
        end if
    end for
    sleep(seconds)
end while
```

Thread 1: Consistent management of jobs running in pods

```
cloud.recordAllPods()
if currentNode.state is live then
    for pod in currentNode.podList do
        pod.job.run()
        cloud.getPod(pod.ID).state.bankup()
    end for
else
    originalNode = currentNode
    currentNode = L.nextLiveNode()
    for pod in cloud.getPods(originalNode.ID) do
        transfer pod to currentNode
        pod.state.recoverFromCloud()
        pod.job.run()
    end for
    L.remove(originalNode)
end if
```

Thread 2: Dynamic scaling up of pods in Edge nodes

```
cloud.recordAllPods()
if currentNode.numDataSamples > threthod then
    partOfDataSamples = split(dataSamples)
    newNode = L.nextLiveNode()
    newNode.copyPods(currentNode)
    newNode.inputData(partOfDataSamples)
    for pod in newNode do
        pod.job.run()
    end for
end if
```

speech processing. However, as mentioned previously, it has not been widely applied in the field of activity recognition. The most popular machine learning methods are Hidden Markov Model (HMM), conditional random fields (CRF) and naive Bayes classifier (NBC) and they have delivered reasonably good accuracy results to the field. In this section, we will present our approach in detail, including the data processing method and how to use the CNN model to perform the activity recognition.

A. The residential activity data sample processing

As shown in Figure 1, sensors are deployed into the smart home with a communication gateway. The raw data is collected from the sensors and then sent to the gateway for pre-processing before it is sent out to the Edge layer. Different smart homes may have different sensor deployment layouts, but our Edge-based design is agnostic to the sensor systems, and is also compatible with any sensor deployment layouts. In our system, the data processing starts from the sensor events flow received by the gateway.

TABLE I: The sequential sensor events fragment of the residential activity 'making a phone call'

Date	Time	Sensor ID	State
2008-02-27	12:43:27.416392	M08	ON
2008-02-27	12:43:27.8481	M07	ON
2008-02-27	12:43:28.487061	M09	ON
2008-02-27	12:43:29.222889	M14	ON
2008-02-27	12:43:29.499828	M23	OFF
2008-02-27	12:43:30.159565	M01	OFF
2008-02-27	12:43:30.28561	M07	OFF
2008-02-27	12:43:31.491254	M13	ON
2008-02-27	12:43:31.491254	M08	OFF
2008-02-27	12:43:32.18904	M09	OFF
2008-02-27	12:43:33.108756	I08	ABSENT
2008-02-27	12:43:33.587637	M14	OFF
2008-02-27	12:43:59.493194	M13	OFF
2008-02-27	12:44:03.717043	M13	ON
2008-02-27	12:44:09.915839	M13	ON
2008-02-27	12:44:15.692528	M13	OFF
2008-02-27	12:44:17.967239	M13	ON
2008-02-27	12:44:25.496747	M13	OFF
2008-02-27	12:44:27.812921	M13	ON
2008-02-27	12:44:28.829018	M13	OFF
2008-02-27	12:44:29.582079	M13	ON
2008-02-27	12:44:35.42195	M13	OFF
2008-02-27	12:44:35.83683	M13	ON
2008-02-27	12:44:40.482497	M13	OFF

The sensor deployment is the same as the one from Washington State University [11], which is further illustrated in Figure 4. In total 39 sensors with different functionalities are deployed into two rooms to capture five types of activity. The types of sensors used are as follows: motion sensor, item sensor, water sensor, door sensor, and phone sensor. The five different residential activities include making a phone call, washing hands, cooking, eating, and cleaning. A fragment of data sample for "making a phone call" is presented in Table I. As shown, an activity includes a series of consecutive sensor events. Each row denotes that a sensor event triggered the state change. In the dataset, the state recorded by different type of sensors varies. For the motion sensor, the state always remains at 'off', and it will be switched to the status 'on' only if someone enters into its coverage area. The item sensors are attached to objects used at home, such as bowl, pot, and container. When the item is used by a resident, the sensor records the state as 'absent'. The door sensor keeps recording its state change using 'open' or 'closed'. And the water sensor and burner sensor record the real-time usage as floating numbers.

The diverse data representations used in the dataset bring extra difficulties for the following computation as well as the inference. This is because the collected values are represented

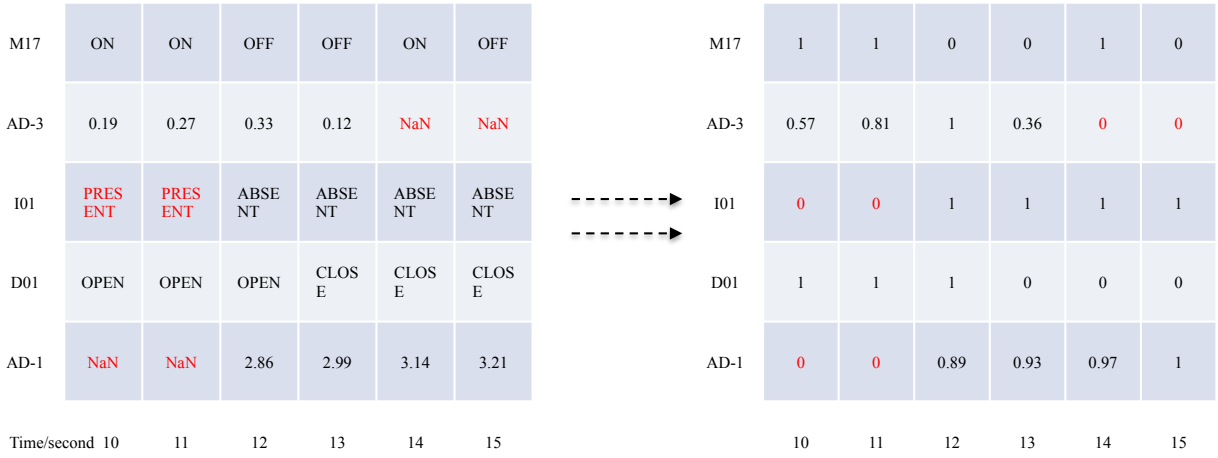


Fig. 5: The state matrix transformation from the sensor events of an activity

in either discrete types or consequent numbers. To unify the data representation, we convert the data samples into the same format before the computation begins.

When a resident performs a specific activity, such an activity might trigger a set of sensors in sequence. The sensor events are then represented by a state matrix. An example data transformation is illustrated in Figure 5. The row denotes a sequence of sensor events, and the column denotes the events occurring in every second. As shown in the state matrix on the left, the states in red mean that the sensors are not triggered and remain at default. On the right, we convert them into floating numbers. The states 'ON', 'ABSENT', and 'OPEN' are now represented as 1, and other analog states are now represented as 0. For the number readings, the normalization is also applied so that the readings stay in the range of [0,1].

After the conversion, the sensor events are represented by a state matrix, which has 39 rows. As data samples are acquired by performing different activities, the time span for an activity varies. Thus, the number of columns in different state matrices also varies. We use 0 to expand the matrices so that they can have the same number of columns.

B. The convolutional neural network model for the activity recognition

In this section, we introduce how to apply the CNN model for the recognition of activity. We utilize the Tensorflow for our CNN model design. The detailed structure of the dedicated CNN model is illustrated in Figure 6.

In our CNN model, we have designed two convolutional layers, each followed by a max-pooling layer. The specification of our model is as follows. The first convolutional layer has 15 kernels and the size of each kernel is a 5 by 5 matrix. The activation function is the rectified linear units function (ReLu). The following max-pooling layer's window size is 2 by 2, and the stride is 2. The second convolutional layer has 20 kernels, each of which is a 5 by 5 matrix, and the activation function is also ReLu. The specification of the second max-pooling layer is the same as the first one. If the max-pooling layer is flattened, it will be a full-connect layer with 1000 neural units. The softmax module is applied after the logits module to produce the probability of each sample's type.

For our model, we choose the ReLu function as the activation function, and cross entropy as the loss function to speed up the convergence of the training and avoid gradient explosion or disappearance. We also apply the drop-out strategy for the classification to avoid the over-fitting of the model when testing it with a different dataset.

In fact, the residential activity recognition is a classification problem. When an activity is being conducted at home, it is rather easy to find some similar patterns happening again. Take "making a phone call" [20] as an example, the general procedure includes moving to the phone, looking up a phone number from the phone book, dialing the number, and listening to the response. The state matrix is similar to the representation of an image, where a sensor event in state matrix is equivalent to a pixel in an image. And the texture of this image represents the activity pattern residents often follow. As the CNN is capable of capturing the high level relationship among pixels in an image, the activity pattern would also thus be well recognized by the CNN model.

V. The Experiments

In this section, the conducted experiments and the results are presented. Our test bed includes five Raspberry Pis serving as Edge nodes, and a tower server with 32GB memory, Intel i7 8700K CPU, 1TB disk, and a GeForce GTX 1080 graph card to act as a mini cloud server. Kubernetes v1.8 and Docker v17.03 are applied to manage the Edge architecture. And Tensorflow v1.6.0 is utilized at both the training and inference phase of the CNN model.

In our experiments, the dataset is from the WSU smart home project [20]. It has 51 volunteers participating in the tests, and each one of them performs five activities in the test environment. First, we split the data samples into the training set (50% of the total samples included), and the testing set (the

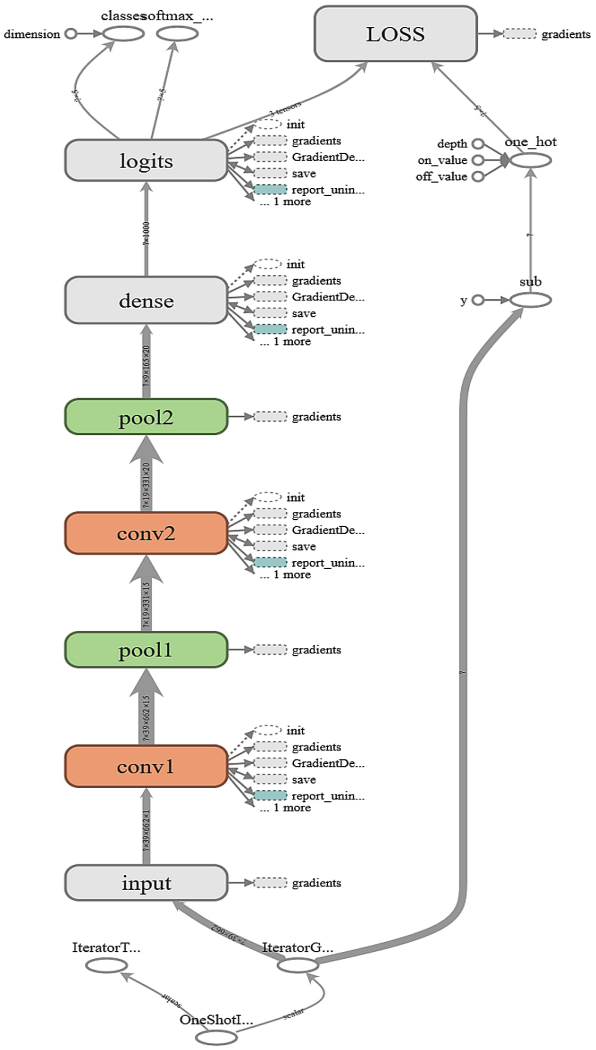


Fig. 6: The detailed structure of the convolutional neural network model for activity recognition

other 50% of samples). After the splitting, the data samples were not quite adequate for training a mature CNN model. To address the issue, we adopted the strategy of repetition, which is widely used in model training for scarce data samples. We repeated the training set 500 times, and randomly chose 50 samples as a batch to be the input each time. The repetition could effectively resolve the problem, while batch training could improve the convergence speed as well the memory usage.

In Figure 8, we illustrate the statistics for the training of our CNN model. The abscissa is the training steps, and the ordinates are model accuracy and loss respectively for the training set. It shows that, with the increment on the number of training steps, the accuracy increases smoothly to nearly 90%, and the loss value keeps decreasing to nearly 0. The blue point in the loss figure denotes that when we test our model on the

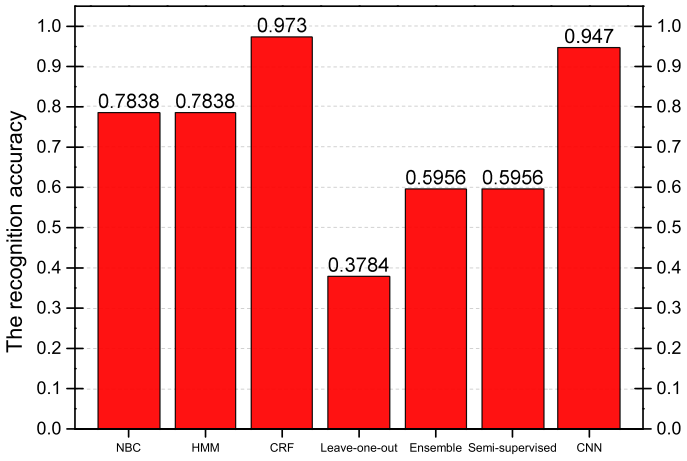


Fig. 7: The recognition accuracy for different models on the dataset Koyto1

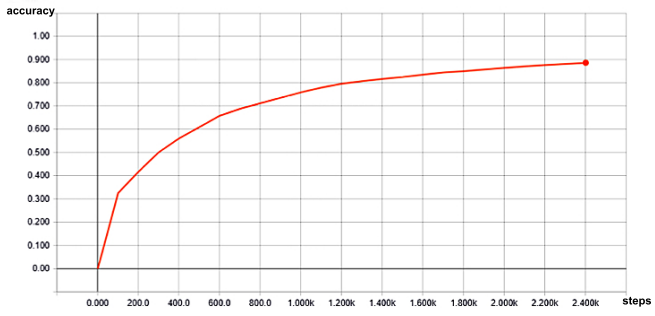
testing set, the loss reaches to a rather low value, around 0.2. It directly reflects the effectiveness and generalization of our CNN model. Meanwhile, the loss value for the testing set also shows it has avoided the possibility of over-fitting.

We also tested our CNN model on the Edge system introduced in Section III, for the real-time activity recognition. The accuracy of our CNN model, with the comparison of different models and settings proposed in [11] is illustrated in Figure 7. In [11], NBC, HMM and CRF models were tested on 11 different smart environments with different settings in a hybrid way. For example, the 'leave one out' setting meant the model was trained on several smart environments and tested on another one. The 'ensemble' setting meant an ensemble model composed of three classifiers was applied. And the 'semi-supervised' setting meant the experiment included unlabeled data for model training. Here we also took advantage of the same dataset for our model verification. As shown in Figure 7, our CNN model achieved high accuracy, nearly 95%, which was similar to the CRF model, and outperformed all other models and settings.

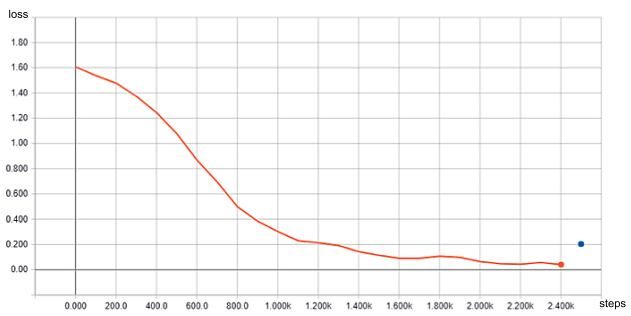
VI. Conclusion and Future Work

In this paper, we propose an Edge-based architecture for the smart home environment. Our design includes the sensor layer, the Edge layer, and the Cloud layer. In the implementation, we used Raspberry Pis as our Edge nodes, and a tower server as the Cloud server. With the deployment of Kubernetes, our Edge computing system can easily adapt to different sensor deployments and settings. We addressed several key issues in realizing our Edge computing system, such as the consistency and scaling problem, and the parallelism for the Tensorflow jobs among pods. Our Edge computing system enables the intelligence at the network edge, while reducing the communication cost of sending the sensed data to the cloud over the Internet.

We also propose a convolutional neural network model for activity recognition. First, we convert the sequential sensor



(a) The model accuracy with respect to the training step



(b) The loss value with respect to the training step

Fig. 8: The statistics for the training of our CNN model.

events data into a state matrix so that the latent activity pattern can be easily captured by the model. Our CNN model is designed by Tensorflow, and tested with the smart home dataset from [20]. The experiments show that our model can reach 95% accuracy on activity recognition.

In the future, we would like to incorporate more sensor environments with different settings to our Edge-based smart home architecture. In [20], tens of smart home testbeds with different settings are built. We can further study the performance of our design under different smart home environments. Other neural models would also be investigated in the future, such as recurrent neural networks, which is capable of processing the sequence data in a more convenient and natural manner.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[2] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "Mavhome: An agent-based smart home," in *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*. IEEE, 2003, pp. 521–524.

[3] A. Cocchia, "Smart and digital city: A systematic literature review," in *Smart city*. Springer, 2014, pp. 13–43.

[4] R. Huisman, "Amsterdam innovation arena," https://amsterdamsmartcity.com/projects/amsterdam-arena, accessed 2016.

[5] W. Li, T. Yang, F. C. Delicato, P. F. Pires, Z. Tari, S. U. Khan, and A. Y. Zomaya, "On enabling sustainable edge computing with renewable energy resources," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 94–101, 2018.

[6] D. J. Cook and N. C. Krishnan, *Activity learning: discovering, recognizing, and predicting human behavior from sensor data*. John Wiley & Sons, 2015.

[7] "Going deeper into action recognition: A survey," *Image and Vision Computing*, vol. 60, pp. 4 – 21, 2017, regularization Techniques for High-Dimensional Data Analysis.

[8] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *Pervasive Computing*, A. Ferscha and F. Mattern, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 158–175.

[9] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, ser. UbiComp '08, 2008, pp. 1–9.

[10] H. Alemdar, H. Ertan, O. D. Incel, and C. Ersoy, "Aras human activity datasets in multiple homes with multiple residents," in *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 232–235.

[11] D. J. Cook, "Learning setting-generalized activity models for smart spaces," *IEEE intelligent systems*, vol. 27, no. 1, pp. 32–38, 2012.

[12] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *arXiv preprint arXiv:1707.03502*, 2017.

[13] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[14] H. Li, K. Ota, and M. Dong, "Learning iot in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[15] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 328–339.

[16] "What is docker," https://www.docker.com/what-docker, accessed 2018.

[17] "Production-grade container orchestration, automated container deployment, scaling, and management," https://kubernetes.io/, accessed 2018.

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[19] P.-H. Tsai, H.-J. Hong, A.-C. Cheng, and C.-H. Hsu, "Distributed analytics in fog computing platforms using tensorflow and kubernetes," in *Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific*. IEEE, 2017, pp. 145–150.

[20] D. J. Cook and M. Schmitter-Edgecombe, "Assessing the quality of activities in a smart environment," *Methods of information in medicine*, vol. 48, no. 5, p. 480, 2009.