Practical and Robust Activity Modeling and Recognition

Eunju Kim and Sumi Helal, *Member, IEEE*

***Abstract*—We point to limitations inherent in the well accepted and assumed activity theory that underpins most of the current body of research in human activity recognition. We briefly present a generic activity model as a superior alternative and show how it could be used advantageously (over the traditional model) in neural-network based recognizers. We also show how the hierarchical aspects of our generic model allow for semantics to be used to decouple the observation sub-system (sensor set) from the rest of the activity model. We demonstrate the value of this decoupling by experimentally comparing the level of effort needed in making sensor changes and the ramifications of such changes on model updates. We compare the level of effort under the original and our alternative model.**

***Index Terms*—Activity Recognition, Activity Modeling, Semantic based Activity Recognition, Activity Recognition Performance.**

I. Introduction

Activity recognition (AR) research is critical to the enablement of human centric computing and its broad range of ubiquitous applications. Understanding human desires and intentions is a key prerequisite to determining the needed services a pervasive space should offer the user in a variety of contexts. Several activity recognition approaches involving complex activity modeling and recognition methodologies have been developed in the past decade [1][2] [5][8][11]. These approaches represent significant and promising contributions to ubiquitous computing and its applications. However, existing AR technology is not robust enough for implementation in the real world. For a robust and a scalable AR technology, we need to address two major issues.

A. Activity Model Limitation

Activity model is an abstract and often simplified conceptual representation of human activities in the real world [17]. Activity modeling is a systematic approach for organizing and representing the contextual aspects of tool use that is both well grounded in an accepted theoretical framework and embedded within a proven design method [5]. Activity modeling is done based on a set of assumptions related to the detailed definition of activities (Activity Framework). Activity framework is important because it can affect many other activity techniques including activity model and AR algorithm.

Activity theory has been frequently used as a framework for AR research [9][12][13]. Even though activity theory is well known and is often used in activity recognition research, it is not sufficient for real world activity recognition because of several limitations such as ambiguity and inaccuracy [5][6].

Moreover, *activity theory* was created and developed during the 1920s and 1930s by psychologists to understand human activities – that is, understanding by humans who have both high cognitive and understanding abilities. Modern activity recognition techniques are however being employed to understand human activities automatically by intelligent computing systems. Since cognition by a computing system is very different from cognition by a human, it is necessary to revise the activity theory, or at least develop a new activity framework to extend this theory, to better support computer-based recognition systems. The new activity framework should capture human activities in the real world with greater precision adequate to automated activity recognition (AR) system. Then, activity models based on the new framework will be more robust as they inherit the advantages of the new activity framework.

B. Research Effort Scalability

Many AR systems use sensor data gleaned from a smart space as input observations to the recognition system. The AR system in this case is directly coupled to the sensor environment (the number of sensors and their types) [1]. However, the sensor set originally selected for the activity may not be adequate or appropriate. It may need to be changed or extended. Sometimes, new technologies (e.g., a new powerful MEMS sensor or a new nano-sensor) present better sensor choices, which could trigger the need for change, hoping for better recognition performance [1]. Sensor change should be facilitated and made easy – it should not affect the AR algorithm or model significantly, or require researchers to adjust their models. Currently, and as a direct consequence of the limitations of the assumed activity framework, this is hardly the case. A scalable AR approach that can accommodate changes and experimentation is therefore urgently needed.

This work was supported by the National Institutes on Health by a Grant number 5R21DA024294.

Eunju Kim is with the Mobile and Pervasive Computing Laboratory, Computer & Information Science & Engineering Department, University of Florida, Gainesville, FL 32611, e-mail: ejkim@ cise.ufl.edu).

Sumi Helal, directs the Mobile and Pervasive Computing Laboratory, Computer & Information Science & Engineering Department, University of Florida, Gainesville, FL 32611, e-mail: helal@ cise.ufl.edu).

Scalability of existing AR systems is limited due to two reasons. Firstly, without careful design consideration (e.g. hierarchical structure), many probability-based activity recognition algorithms such as the Hidden Markov Model (HMM) or Conditional Random Field (CRF) model are not scalable because their emissions and observations are closely tied to the sensor set [4]. To illustrate, the emission of hidden Markov model and conditional random field model are significantly dependent on an observation sequence obtained from the sensor data. In this case, recognition performance will be influenced considerably by any sensor change.

Secondly, AR algorithms require re-training when sensors are changed. Specifically, sensor data needs to be labeled in order to glean meaningful information from collected raw data. Since this labeling is tedious, many machine learning based approaches are developed to automate it. However, those approaches themselves require training. Whenever there is a sensor change, new training with a new learning data set is needed. This re-training requires significant time and effort by the researcher, which hampers and limits the scope of possible exploration and experimentation. Therefore, new methodologies addressing this issue of scalability are needed.

The rest of the paper is organized as follows. Section II explains the proposed new activity framework along with an algorithm for a scalable AR system. Evaluation, comparison and experimental results are presented in Section III.

II. Proposed Approach for Robust Activity Modeling and Recognition System

Our proposed AR system is composed of a domain specific activity model, an AR algorithm and an activity knowledgebase. All three subsystems are based on a "*generic activity framework*" shown in Fig. 1.

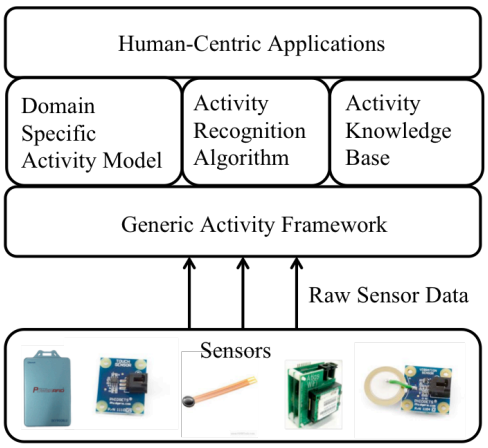


Fig. 1. Architecture of Activity Recognition System

A. Generic Activity Framework for Activity Modeling

The generic activity framework provides a hierarchical structure, in which each layer of the structure consists of activity components. In total, there are nine primitive components in our proposed generic activity framework as shown in Fig 2. It is not necessary for every activity to contain all nine components as long as the activity is recognized clearly. For example, the walking activity does not require any object. Descriptions of the nine primitive components are summarized below, and described in details in [3]:

Subject. A subject is an actor of the activity. Subject has an important role as an activity classifier especially when there are multiple people.

Time. This is the time when an activity is performed. It consists of start time and end time. We can also calculate the *duration* of an activity using time.

Location. Location is the place where an activity is performed. If an activity is performed in several places, location will have multiple values.

Motive. Motive is the reason or objective why a subject performs a specific activity.

Tool. Tool is an artifact that a subject uses to perform an activity. Tool provides essential information to classify activities. For example, a spoon or a fork is a tool for *eating* or *cooking*. Therefore, an AR system can expect those activities when it detects that a user uses a spoon or fork.

Motion. Motion. We define motion as the movement performed by a subject for handling tools. *Motion* explains what a subject does with a tool. For example, *cutting* and *chopping* are both performed using the same tool i.e. knife. The different motions associated with *cutting* and *chopping* can be used to differentiate between them.

Object (Target of actions). An object can also be any artifact like tool. But, object is the target of an activity while as a subject uses a tool. Distinction between tool and object is important for accurate activity recognition because some artifacts are both tool and object depending on an activity.

Order. Order is the sequence in which actions of an activity are performed. Usually order does not matter for many activities. But order is important for some activities. For example, to eat food, we should serve food first and cut, pick or scoop food.

Context. Context provides information about the "vicinity" in which an activity is performed. Installed sensors directly find some contexts such as temperature or humidity. Other primitive components such as time or location contribute towards finding contexts. On the other hand, some contexts like motive of an activity need some artificial intelligence techniques such as reasoning or inference to elicit them.

Fig. 2 shows a composition diagram of the generic activity framework. Rectangles are layers and ellipses are primitive components. According to the composition of components, the activity framework has a hierarchical structure. And the components of each layer are clearly defined. Brief description for each layer is given below (more details in [3]):

Sensors. Sensors are installed in the pervasive space (e.g. a smart home) to collect event information of the space. Based on the source of sensor data, sensor is classified into four types: *motion, tool, object,* and *context* sensor.

Operation. Operation is a composition of *tool* and *motion*. The user operates tools with specific motion. For example, if computer is a tool, some hand or finger motion will be performed for typing a keyboard.

Action. Action is determined by combination of *operation* and *object*. For example, if a user types a command to open a file, typing on the keyboard is an operation and the file is an object and this combination is open file action.

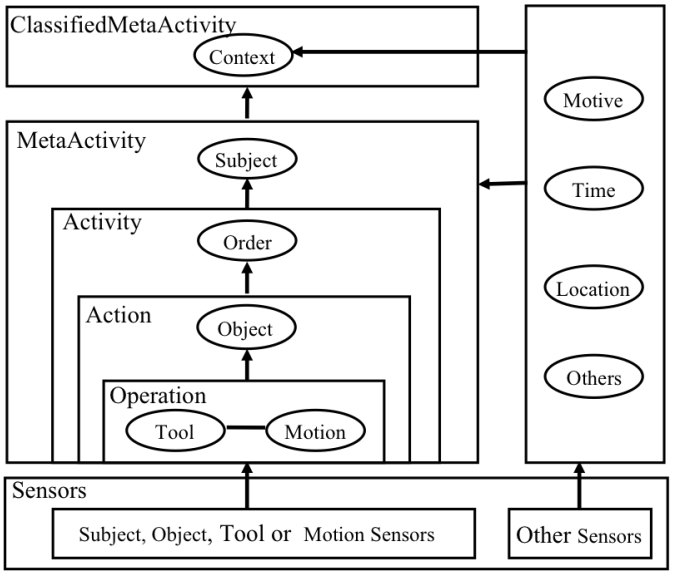


Fig. 2. Composition diagram of a generic activity framework. It is composed of several hierarchies and each hierarchical layer contains classifier components.

Activity. Activity is a collection of *actions*. Activity may involve multiple actions. For some activities, the order of actions is important. For example, to modify a file, we need to open the file first.

Meta activity. A meta activity is a collection of *activities*. When an activity is complicated, it is composed of several simple activities. For instance, a meta activity hygiene is composed of washing hands, brushing teeth or taking a bath.

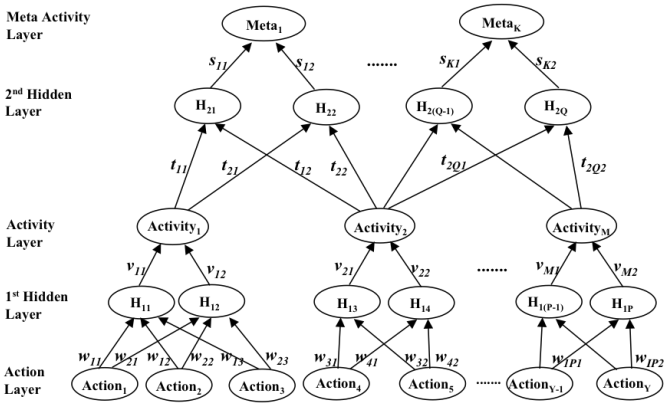
Classified meta activity. When *meta activity* is combined with *context* including *time* or *location*, the *meta activity* can be more specialized. For example, a meta activity *having a meal* is classified into several meta activities such as *having breakfast, lunch,* or *dinner* according to time of the activity performed.

The hierarchical structure has several advantages. Firstly, it makes the activity recognition system more tolerant to sensor environment change [1]. For instance, even if more sensors are inserted in the AR system, the upper layers in the hierarchy will not be seriously influenced. Secondly, activity recognition using hierarchical structure is analogous to the way people recognize, so it is easier to design more natural and intuitive AR algorithm [3].

B. Neural Network Based Activity Recognition Algorithm

We are currently exploring and designing several AR algorithms based on our generic activity framework. In this paper we report on one of these AR algorithms that utilizes a Multi Layer Neural Network (MLNNK). We chose MLNNK because it possesses a hierarchical structure that maps very well with the generic activity framework and its algorithm structure. Fig. 3 illustrates clearly how layers of our generic activity framework are directly mapped to the layers of the MLNNK network. Also, MLNNK is a localized neural network, which means it has less training burden than a unified neural network. A unified (single) neural network that captures all activities is not feasible because of the high computational cost and time involved. For example, when two unrelated activities such as cooking and laundry are performed, their inputs are also unrelated. However, in a unified neural network, these inputs are computed together. Therefore, cooking activity should compute laundry sensor data and vice versa. Moreover, especially if there are hundreds of target activities of an AR system, even one sensor change can overwhelm the system because the system should find the relationship between the new sensor and all target activities. This explains why neural networks are not commonly used in activity recognition.

To adopt neural networks while eliminating the unnecessary waste of computational resources, we design our AR system in a way that allows unrelated activities and meta activities to have their separate neural networks so that these networks focus on more relevant relationships. We shall refer to these networks as localized neural networks. Fig. 3(a) shows an example of a two layer neural network. In this example, there are three localized neural networks (or sub-neural network) in activity layer and two localized neural networks in meta activity layer. Activity layer is the output of the first hidden layer and the input for the second hidden layer. A neural network computes an output according to an activation function as shown in Fig. 3(b).



(a)

$$NET_{H_{11}} = [Action_1, Action_2, Action_3]\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \end{bmatrix},\ H_{11} = f(NET_{H_{11}}) = \left(1 + exp^{-NET_{H_{11}}}\right)^{-1}$$

$$NET_{H_{12}} = [Action_1, Action_2, Action_3]\begin{bmatrix} w_{21} \\ w_{22} \\ w_{23} \end{bmatrix},\ H_{12} = f(NET_{H_{12}}) = \left(1 + exp^{-NET_{H_{12}}}\right)^{-1}$$

$$NET_{Activity_j} = [H_{11}, H_{12}]\begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix},\ Activity_1 = f(NET_{Activity_j}) = \left(1 + exp^{-NET_{Activity_j}}\right)^{-1}$$

(b)

Fig. 3. Localized Activity Recognition Neural Network. It is composed of several sub-neural networks, which are localized according to the relationship between inputs and outputs [10].

One major advantage of MLNNK is that it considers only effective relationships for activity recognition so that it can

reduce error. Both HMM and CRF are time sequential graphical models, which are popularly used for activity recognition. HMM requires finding all possible orders between input actions. Due to the complex nature of human activities, finding all possible orders is cumbersome for a practical system. Furthermore, missing orders will cause the HMM to produce errors. CRF solves this problem by neglecting the order constraint. CRF does not consider order and it considers only state and transition relationships. The former is the relationship between input observation (actions or activity in Fig. 3) and output (activity or meta activity in Fig. 3). The latter is the relationship between output activities in the same layer (relationship between previous activities at time slice *t* and next activities at time slice *t+1*). CRF could outperform HMM even though it removes orders from an activity model [8]. Similarly, we can remove transition relationships of CRF because it is not only difficult but also meaningless to enumerate every possible relationship unless target recognition scope is small and its relationships are clear. For example, in highly abstract recognition domains such as daily living activities, after getting up in the morning, we can do a lot of things such as eating, cooking, taking a bath, running, etc. Enumerating all possible relationships is difficult and if there is a missing relationship, it will increase the error rate. MLNNK ignores the transition relationship between activities and considers only state relationships. Therefore, MLNNK is expected to outperform CRF by removing the unnecessary constraints.

C. Supporting Scalability of MLNNK using Knowledgebase

When sensor environment of an AR system changes because of insertion or deletion of sensors, an AR system needs to change a neural network with the new sensor. Also it needs to retrain the neural network with new training data. These tasks represent a huge burden for many machine learning based algorithms because training and re-training require a lot of human effort. Our AR system supports sensor changes by avoiding human intervention and re-training, using as an alternative approach, a knowledgebase. In the following, we describe activity knowledgebase first, and then show how the knowledgebase can be utilized by the MLNNK algorithm during sensor changes and neural network training.

Activity Knowledgebase (KB). An activity knowledgebase defines and stores activity entities such as action, activity or meta activity and precise relationships between the various activity layers. This knowledge is used to maintain a multi-layer neural network. It is also used to generate training data for the neural network. The followings are parts of the activity KB schema.

Sensor(sensor_id, sensor_type, type_name, value_type, min_value, max_value, installed_place)
Tool(name, feature_function)
Motion(name, type, feature_function)
Object(name, feature_function)
Action(action_name, motion_name, tool_name)
Activity(activity_name, action_name, object_name)
MetaActivity(meta_activity_name, activity_name, context).

Support of sensor insertion and Deletion. When a system detects a newly added sensor, the AR system sends queries to the knowledgebase to obtain the actions, activities, and meta activities corresponding the new sensor, to build a new sub-neural network. Then, the new and existing neural networks are merged together. Therefore, sensor insertion affects only an isolated part of the neural network. If the new sensor cannot be found in the knowledgebase, the user is prompted to edit the knowledgebase and add the required information about the sensor, including the relationships between the sensor and motion, tool, object and action. When a sensor is deleted, it doesn't have any effect on the neural network. The action related to the sensor is simply inactivated. Only when the activity model is changed, the neural network is affected.

TABLE I
ERROR BACK PROPAGATION ALGORITHM [10]

I (Input data), H_1 (1st hidden layer), A (Activity),
H_2 (2nd hidden layer), M (Meta_activity)
s, t, v, w: weight vectors of each layer, (see Fig. 3)
d_1, d_2: desired activity, meta_activity of the input
p; the number of training data pattern pairs
E_1, E_2: Output error, initially zero
m1, m2: the number of hidden units in each hidden layer

(1) Assign initial weight (s, t, v, w) with random value.
(2) Set learning rate α (>0) and maximum error (E_{max}).
(3) For each training pattern pair (x, d_1, d_2),
Do step 4-8 until k = p. (x: training pattern).
(4) Compute output H_1, A, H_2, M (see Fig. 3. (b)).
(5) Compute output errors E_1, E_2.

$$E_1 = \frac{1}{2}(d_1 - A)^2 + E_1, \quad E_2 = \frac{1}{2}(d_2 - A)^2 + E_2$$

(6) Calculate the error signal

$$\delta_{Activity} = (d_1 - A)A(1 - A)$$

$$\delta_{MetaActivity} = (d_2 - M)M(1 - M)$$

$$\delta_{Hidden_1} = H_1(1 - H_1)\sum_{i=1}^{m_1}\delta_{Activity}v$$

$$\delta_{Hidden_2} = H_2(1 - H_2)\sum_{i=1}^{m_2}\delta_{MetaActivity}s$$

(7) Update weights s, t, v, w.

$$s^{k+1} = s^k + \Delta s^k = s^k + \alpha\delta_{MetaActivity}H_2^k$$

$$t^{k+1} = t^k + \Delta t^k = t^k + \alpha\delta_{Hidden_2}A^k$$

$$v^{k+1} = v^k + \Delta v^k = v^k + \alpha\delta_{Activity}H_1^k$$

$$w^{k+1} = w^k + \Delta w^k = w^k + \alpha\delta_{Hidden_1}I^k$$

(8) Increase a counter and go to Step 4.

$$k = k + 1$$

(9) Test stop condition.
If ($E_1 < E_{max}$ and $E_2 < E_{max}$) stop else, E_1=0, E_2 = 0

MLNNK Training. When a new sensor is installed, the neural network needs to be retrained with new training data. Generating a new training data requires a lot of human effort as we mentioned earlier. By utilizing the knowledgebase, our AR system is able to generate training data automatically,

using information about the types of sensors used in the target smart space. In other words, we store the sensor's information such as the range of the values of the sensor data, feature function, and possible install locations. Using this knowledge, we can generate possible combinations of data set that sensor set produce. For example, if we install a new snoring sensor for recognizing a sleep apnea activity, a possible location of the sensor is at the bedside in the bedroom. Also, we can find the features of the snoring sensor including value range from vendor datasheet (e.g. If the sensor value is 0 for no snoring and 1 for yes snoring). We store this information into a knowledgebase. Later when an AR system finds the snoring sensor installed in bedroom, it queries the knowledgebase, and retrieves all related information such as sensor information and model information like related actions and activities. Then, we can generate training data automatically with this information. Table I shows the training algorithm, which is based on the error back propagation algorithm.

III. Experiments and Results

In this section, we validate our approach through experimentation. Our evaluation goal is to answer the following two questions.

(1) How accurate is our activity recognition algorithm and approach and how does its accuracy performance compare to other algorithms/approaches?

(2) How scalable is our approach to sensor changes? And how does its scalability in terms of human effort compares with other approaches?

To perform this evaluation study, we used a real world data set, which is provided by University of Amsterdam (we will refer to it in this paper as the Amsterdam dataset) [8]. This data set records activities of daily living performed by a 26-year-old man living in a three-bedroom apartment for 28 days. Sensors are installed in several places in the apartment including doors, cup-boards, refrigerator, and toilet flush. Activities (such as 'Leaving', 'Toileting', 'Showering', 'Sleeping', 'Drink', 'Breakfast', and 'Dinner') are annotated by the subject himself using a Bluetooth headset [16] and used to compare the performance of activity recognition system.

A. Experiment Setup

As a first step, we built an activity knowledgebase suitable for the Amsterdam dataset. The knowledgebase classified activities and meta activities as shown in Table II.

TABLE II
Meta Activities and Activities

Meta Activity	Activity
Cleaning	WashingDishes
GoingOut	LeavingHouse
Hygiene	Drinking, GoingToBed, TakingShower, Toileting
Laundry	WashingClothes
PreparingBreakfast	Cooking at Morning
PreparingLunch	Cooking at Lunch
PreparingDinner	Cooking at Dinner

We then implemented an activity recognition system using the MLNNK algorithm. The AR system collects sensor data periodically according to a time slice (1 minute) and sends the data to the neural network, which recognizes activities from the data. The recognized activities are compared with the activities annotated by a subject to measure accuracy of activity recognition algorithm.

B. Experiment 1: Accuracy Performance

To measure the performance of our AR approach/system, we measured *precision*, *recall*, and *accuracy* in terms of *true positive, true negative, false positive,* and *false negative* [7]. These statistical measures are defined below:

$$Precision = \frac{tp}{tp + fp} \quad (1)$$

$$Recall = \frac{tp}{tp + fn} \quad (2)$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad (3)$$

True positive (tp): the number of correctly recognized activities.
True negative (tn): the number of gaps between activities, which had no activities.
False positive (fp): the number of activities that are not actually performed but recognized.
False negative (fn): the number of activities that are not recognized even though they were actually performed.

Table III shows the recognition performance of our approach. False positive is found to occur more frequently than false negative. We can see that meta activity recognition performance is not much different from activity recognition performance. This is not surprising because activity is input for recognizing meta activity, and meta activity recognition performance is highly depending on activity recognition accuracy.

TABLE III
Activity Recognition Performance

	Meta Activity	Activity
False positive	8.6 %	9.0 %
False Negative	2.1 %	2.1 %
Precision	87.7 %	87.4 %
Recall	96.7 %	96.6 %
Accuracy	89.9 %	88.7 %
Actual Accuracy	92.5 %	92.0 %

We compared the performance of our approach with HMM and CRF model based approaches as shown in Table IV. We utilized two statistical measures defined in [8]: time slice accuracy and class accuracy.

$$Timeslice = \frac{\sum_{n=1}^{N} [inferred(n) = true(n)]}{N} \quad (4)$$

$$Class = \frac{1}{C} \sum_{c=1}^{N_c} \left\{ \frac{\sum_{n=1}^{N_c} [inferred_c(n) = true_c(n)]}{N_c} \right\} \quad (5)$$

In Equation (4) and (5), [a=b] is a binary indicator yielding 1 when true, 0 otherwise. N is the total number of time slices, C is the number of different activity types (classes) and Nc is the total number of time slices for class c [8]. *Timeslice* accuracy counts true positives within a time slice. It offers a more meaningful measure of recognition performance than global accuracy especially when some activity classes are more frequent than others [8]. *Class* accuracy averages time slice accuracy over all activity types, which is bound to report lower numbers than time slice but is more representative of actual model performance. HMM and CRF performance has been compared in Table IV using the *Timeslice* and *Class* accuracy over the Amsterdam dataset.

Table IV shows the performance comparison among the three approaches. Our approach, MLNNK, shows higher accuracy than HMM and CRF for both *Timeslice* and *Class*. The *Class* accuracy is over 23% higher than CRF and about 14.6% higher than HMM. Another observation is that MLNNK shows similar performance for *Timeslice* and *Class* whereas *Timeslice* performance is higher than *Class* for HMM and CRF. There are a couple of reasons for this performance difference. First, the performance between *Timeslice* and *Class* in CRF and HMM are different because their training data are collected in a biased situation. In reality, some activities will be performed frequently whereas other activities will be rare. Then, the rare activity classes take more time to be trained because they need to wait until their training data is collected [8]. However, *Timeslice* does not have this difference. Therefore, the overall performance of *Timeslice* is higher than that of *Class*.

However, MLNNK shows similar performance for both *Timeslice* and *Class*. It is because MLNNK training is different from other approaches. MLNNK does not wait until training data is collected. In other words, since our AR system can produce initial training data using sensor and activity model information in knowledgebase, every class is trained almost equally. Also, because MLNNK is a localized network, the training data is not huge unlike a unified neural network. Therefore, MLNNK can start training every class from the beginning and shows similar performance.

TABLE IV
ACTIVITY RECOGNITION PERFORMANCE COMPARISON WITH OTHER MODELS

	Time Slice	Class
Hidden Markov Model [6]	94.5%	79.4%
Conditional Random Field Model [6]	95.6%	70.8%
Multi-Layer Neural Network (MLNNK)	96.9%	94.0%

Table V dissects the accuracy for each activity class. PreparingBreakfast has the worst accuracy while LeavingHouse, GoingToBed, and PreparingDinner have the best accuracy. This result is similar to previous experiments reported in [8].

TABLE V
DETAILED CLASS ACCURACY RESULT

Class	Accuracy
LeavingHouse	1.00
Toileting	0.94
TakingShower,	0.95
GoingToBed,	1.00
PreparingBreakfast	0.76
PreparingDinner	1.00
Drinking	0.91

C. *Experiment 2: Research Effort Scalability*

To examine the scalability of our approach in comparison with others, we accounted for and compared the human effort required in updating the *model* in response to sensor changes. As mentioned earlier in the paper, this scalability is very important to researchers who wish to experiment with various combinations of sensors to discover the sensor set that boosts the AR performance. We considered only sensor insertion in this study. Sensor deletion and sensor attribute changes could be studied similarly.

To measure human effort of our MLNNK approach, we counted the number of changes we had to make to the knowledgebase and MLNNK. We also estimated the human effort required in updating the CRF model for the same set of sensor changes we applied to our approach. Model changes in MLNNK means the number new relationships that must be added by the researcher to the knowledgebase (through a GUI interface), for each sensor insertion. We inserted 30 new sensors one by one, in a random order. We then counted the number of changes that needed to be made for both approaches. We used three sets of 30 sensors (each), repeated this experiment three times and calculated an average. For the MLNNK approach actual changes were made and counted. For the CRF approach, a graph was manually maintained and manipulated and every change to the graph was counted.

Fig. 4 and Fig 5 show the result of this experiment. Inserting all 30 sensors costs 105 changes under MLNNK, and 250 changes under CRF, an almost 58% advantage for MLNNK (Fig. 5). Also it was observed that under MLNNK, initial human effort was higher than CRF because of the initial updating cost of activity knowledgebase. However, human effort in CRF increased with the insertion of more sensors, and it overwhelmed the effort in MLNNK (Fig. 4). This is because MLNNK utilizes the knowledge such that some of the inserted sensors do not have any relationships with some of the activities. For example, insertion of a sensor installed in a cup is not related to sleeping activity. Since MLNNK has this knowledge (of no relationship), it could avoid unnecessary changes and hence reduce the number of changes per single sensor insertion. However, without knowledgebase, CRF had,

by definition, to consider all possible relationships between the new sensor and existing activities.

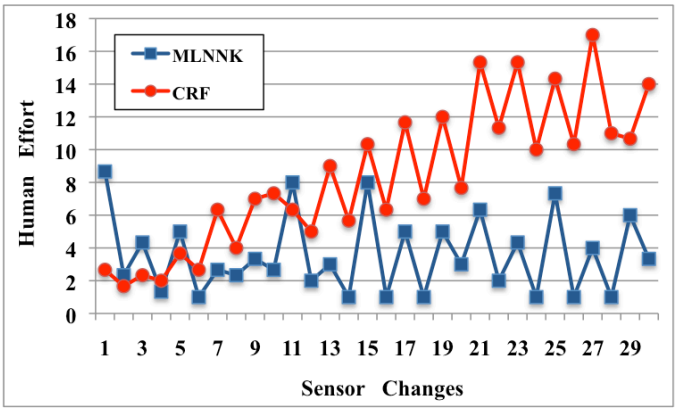


Fig. 4. Human effort and sensor changes

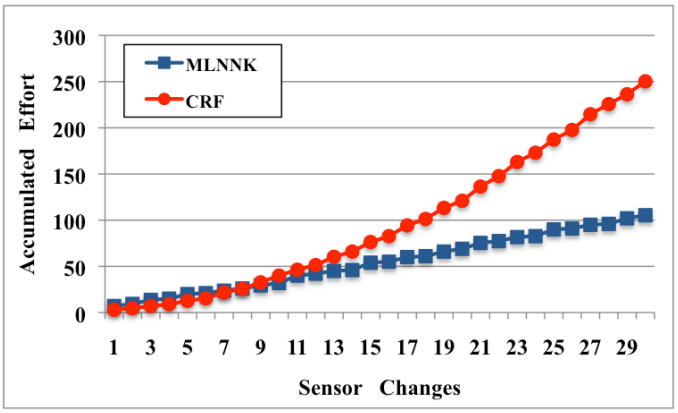


Fig. 5. Accumulated human effort and sensor changes

IV. Conclusion

This paper introduces a practical and robust activity recognition approach based on a generic activity framework, scalable AR algorithm and AR knowledgebase. A scalable AR algorithm is introduced based on the activity framework. The algorithm utilizes an activity knowledgebase to achieve higher levels of scalability and to enhance recognition performance. Our experimental results show the advantage of our approach over well established ones: Non Hierarchical HMM and CRF. Results also prove that utilizing invariant knowledge about human activities and their observation subsystems does improve the activity recognition performance and reduces the user's efforts required after sensor changes for model change and retraining, which boosts effort scalability.

References

[1] S. Helal, E. Kim, S. Hossain, "Scalable Approaches to Activity Recognition Research," in 8th International Conference Pervasive Workshop (2010).

[2] S. Helal, D. Cook, M. Schmalz, "Smart Home-based Health Platform for Behavioral Monitoring and Alteration of Diabetes Patients," in Journal of Diabetes Science and Technology, vol. 3, n. 1, Jan (2009)

[3] E. Kim, S. Helal, "Revising Human Activity Frameworks," in 2nd International Conference on S-Cube 2010.

[4] E. Kim, S. Helal, D. Cook, "Human Activity Recognition and Pattern Discovery," in IEEE Pervasive Computing vol. 9, n. 1, pp. 48-52 (2010)

[5] L. Constantine, "Human Activity Modeling: Toward A Pragmatic Integration of Activity Theory and Usage-Centered Design," in Human-Centered Software Engineering, p24-50 (2009).

[6] K. Kuutti, "Activity theory as a potential framework for human-computer interaction research," in B.A Nardi (eds.), Context and consciousness: Activity theory and human-computer interaction.Cambridge, MA: MIT Press (1996).

[7] S. A. Alvarez, "An exact analytical relation among recall, precision, and classification accuracy in information retrieval," in Technical Report BCCS-02-01, Computer Science Department, Boston College, 2002.

[8] T. Kasteren, A. Noulas, G. Englebienne, B. Krose, "Accurate Activity Recognition in a Home Setting," in Proceedings of the Tenth International Conference on Ubiquitous Computing (Ubicomp 2008), Seoul, Korea, pp 1-9, 2008.

[9] P. Lingras, Butz, C.J. "Precision and Recall in Rough Support Vector Machines," in 2007 IEEE Int. Conf. on Granular Computing, pp. 654–658 (2007).

[10] M. Mitchell, "Machine Learning," in WCB/McGraw-Hill, p 88-108, 1997.

[11] W. Pentney, "Sensor-Based Understanding of Daily Life via Large-Scale Use of Common Sense," in Proceedings of AAAI '06, Boston, MA, USA (Jul 2006).

[12] D. Surie, T. Pederson, F. Lagriffoul, L. E. Janlert, D. Sjolie, "Activity Recognition using an Egocentric Perspective of Everyday Objects," in UIC (2007) Proceedings of IFIP 2007 International Conference on Ubiquitous Intelligence and Computing. LNCS, vol. 4611, pp. 246–257. Springer, Heidelberg (2007).

[13] P. Lefrere, "Activity-based scenarios for and approaches to ubiquitous e-Learning,' in Personal and Ubiquitous Computing, vol. 13, n. 3, pp. 219-227 (2009).

[14] H. M. Wallach. "Conditional random fields: An introduction," Technical Report MS-CIS-04-21, University of Pennsylvania CIS 2004.

[15] Neuroph library.http://netbeans.dzone.com/articles/neurophmdashsmart-apps/

[16] T. Van Kasteren, Data set repository at Univ. of Amsterdam. https://sites.google.com/site/tim0306/datasets

[17] http://www.webster-dictionary.org/definition/Model