Universitatea POLITEHNICA din București

Facultatea de Automatică și Calculatoare,
Departamentul de Calculatoare

LUCRARE DE DISERTAȚIE

Generarea grafurilor de context din date senzoriale folosind modele pentru recunoașterea activităților umane

Conducător Științific:
Ș.l.dr.ing. Andrei Olaru

Autor:
Cătălin Badea

București, 2016

University POLITEHNICA of Bucharest

Faculty of Automatic Control and Computers,
Computer Science and Engineering Department

MASTER THESIS

Generating context graphs from sensor data using human activity recognition models

Scientific Adviser:
Ș.l.dr.ing. Andrei Olaru

Author:
Cătălin Badea

Bucharest, 2016

I would like to thank my supervisor, Andrei Olaru,
for the support and guidance he offered me throughout
the development of this project.

Abstract

In the field of Ambient Intelligence(AmI), context-awareness has gained a lot of interest as a research topic. Context-awareness refers to the property of a computer system or device of reacting to information gathered from observing the user. Accurately representing user context is a very important task that needs to be solved for more complex AmI applications to be developed. A solution to this task is based on the idea of using graphs to store and operate on contextual information. Considerable research on this topic was carried to evaluate the applicability of context graphs to real world scenarios and optimized algorithms for matching context graphs to specific patterns were developed. In this paper we discuss a method for generating context graphs from ambient and on-body sensors using semantic attributes. Towards this end, we explore the use of machine learning models for identifying predefined sets of attributes from human activity recognition datasets and provide directions on how these attributes can be mapped in a context graph. We test these models using two open and representative datasets and provide performance analysis on the results using a benchmark of state-of-the-art models associated with these datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ambient Intelligence - commonly abbreviated as AmI - is a concept of computer systems embedded in common use items that are capable of interacting with human users by responding to their presence or actions. The purpose of such devices is to assist the user whenever possible, with the requirement that they should act in an unobtrusive manner. The range of AmI systems vary from simple house assistants to specialized social platforms focused on providing interaction between people with common interests (e.g. remote study sessions)[8].

A key feature of AmI system is context-awareness. Context-awareness[16] refers to the property of a computer system or device of reacting to data gathered by observing the users. Contextual information may include an individuals' current location, the list of items in his schedule, his posture or current arm movement. For a context-aware system to be effective one key requirement is the use of models that are able to extensively describe user context and facilitate decision making in response to this information.

An approach to model contextual information is based on the idea of context graphs[19]. The concept of context graphs is similar to semantic networks, but provides more flexible semantics. Nodes in the graph represent concepts, with edges representing associations between these concepts. Decision making is the result of matching context graph patterns against a given context graph. A pattern is stored in the form of a special context graph which contains generic components: nodes or edges that can match more than one graph component. As such, the problem is reduced to comparing two graphs. An algorithm for matching patterns to context graphs was described by Olaru et al.[17].

Before context graph models can be effectively used in real world application, a key problem must be solved: generating context graph models from sensor data. In this paper we discuss and analyse a method for generating context graph models using sensor output data. We focus on two categories of sensors:

- on-body sensors, such as accelerometers and gyroscopes.
- ambient sensors, door sensors, proximity sensors etc.

The next sections will first define concept graphs more thoroughly and provide some motivation for this work before giving a more detailed description of the problem.

1.1 Problem Context

1.1.1 Context graphs

The approach consists of representing a set of concepts as nodes in a graph with edges representing associations between these concepts. Specific situations can be identified by matching patterns against the current context graph. A pattern is a special context graph which contains generic components. When using context graphs, interpreting user context becomes a problem of comparing two graphs. Olaru et al.[17] propose an algorithm for efficiently matching graphs to patterns.

Using graphs for context modelling is based on existing knowledge representation methods like semantic networks, concept maps and conceptual graphs. The graph stores associations between a number of concepts which are relevant for the user's context. These concepts are part of a knowledge base used by an Ambient Intelligent (AmI) system[18]. Formally, a context graph is defined as:

$G = (V, E)$

$V = v_i, E = e_k, e_k = (v_i, v_j, value)$

where $v_i, v_j \in V, i, j = \overline{1, n}, k = \overline{1, m}$

The values of vertices and edges can be: Strings, URI identifiers, null.

URI identifiers can designate people, objects, relations or more advanced concepts. The value of null does not hold a special status, thus it can be assigned to nodes or edges.

Graph patterns come as an extension over context graphs by allowing wildcard nodes, marked as **?**. As an intuition, we consider that the context graph G matches a pattern P if:

- every vertex from P matches a different vertex from G.
  - generic vertices can match any vertex from G.
  - non-generic vertices can only match vertices with the same value.
- every edge not containing a regular expression from P matches a different edge from G.

A more thorough description of context graphs can be found in previous research [19]. Figure 1.1 shows an example of a context graph. The graph contains information about an user named Bob that is a attending a conference in Paris.

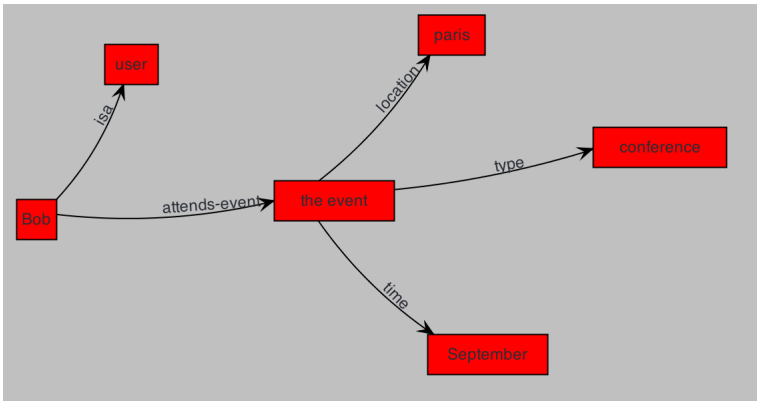

Figure 1.1: The knowledge of the AmI system: Bob attends a conference in Paris.

As previously mentioned, context graphs provide the means for building models used in AmI applications. However, real world AmI applications require multiple components in order to perform their tasks such as: means of acquiring data about its users as well as interfaces to communicate with the user. These components are often tailored for a specific task and are designed on a case by case basis.

To address this situation, the tATAmI agent platform was designed with the goal to facilitate some of these tasks by providing abstractions and APIs which greatly reduce the development time of AmI applications that use context graphs. In the following section we will give some background on this framework.

1.1.2 The tATAmI agent platform

tATAmI[23] is an agent based platform which uses the S-CLAIM language to implement behaviour rules. tATAmI allows the development and deployment of unified agent system. S-CLAIM is an extension of the CLAIM language: Computational Language for Autonomous Intelligent and Mobile agents. It is written in Java, and, using a custom class loading mechanism, was ported to both desktop and mobile platforms, running on Android, Windows, Linux and MacOS. Agents in tATAmI can use contextual information represented as context graphs and can take decisions based on full pattern matches as well as partial matches. The tATAmI framework is useful for quickly building context information models and performing reasoning tasks.

1.2 Generating context graphs from data

A typical AmI system with the purpose of assisting users in their daily activities would maintain a permanent context graph representing the state of the user. The graph should be updated automatically based on sensor inputs. The system uses graph matching for prompting the user when required. Sensor data is usually represented as a sequence $S = e_1, e_2, e_3..$ of sensor events. This representation also applies to continuous sensor data using sampling in specific time frames.

Given an initial context graph G and a sequence of sensor inputs S, the task is to mutate the graph G to a meaningful representation of the user's state at the end of sequence S. This is a very complex task because context graphs use an open-ended ontology of concepts and relations. Using the tATAmI agent platform, this problem may be solved for specific cases. This is achieved by first restricting the type of nodes that are used in the context graph model, using a fixed ontology that's relevant to the application's domain.

Next, because the graph models associations between high level concepts, making changes in the graph is not trivial. Using the tATAmI platform, the solution would be to provide the agent with a rule based system that applies the transformations on the graph, written in S-CLAIM. However, one final task remains. Mapping the sensor data to the ontology used by the rule based system. In practice, sensor outputs cannot be mapped directly to any meaningful symbolic representation in the context graph.

To solve this step, we propose using machine learning algorithms to extract symbolic attributes from sensor data, that can be fed to the rule based system, which will then be able to generate meaningful changes in the context graph.

Towards this purpose, we studied the use of machine learning models used in human activity recognition research to generate symbolic attributes that can be used to generate context graphs. The results presented in this paper are based on two public human activity recognition datasets:

WSU CASAS dataset[6] and the Opportunity dataset[3]. The sensor samples include readings from both ambient sensors and on-body sensors.

1.3 Dissertation plan

Chapter 2 will present current state of the art algorithms used in human activity recognition tasks that are relevant to the problem of learning semantic attributes for context graph generation. For each of these algorithms we discuss the usefulness for the current task. Next, Chapter refchapter:proposed will outline the steps of the algorithm for learning semantic attributes from sensor data and using them to produce context graphs. The following two chapters will present how the algorithm was applied on each of the datasets used, detailing specific requirements and methods used, with Chapter 4 focusing on ambient sensors dataset and Chapter 5 focusing on the on-body sensors dataset. Next, in Chapter 6 we describe the evaluation methods used, present the results for each experiment and provide an interpretation of these results. Finally, Chapter 7 will present the conclusion of this research and provide some directions for future work.

Chapter 2

Related Work

In the field of ubiquitous computing, one of the key problems that is currently being studied is human activity recognition (or HAR). Human activity recognition refers to the process of labelling sensor data with human-readable tags that describe the activity performed by the subject. Common HAR tasks include identifying activities based on input from sensors such as accelerometers or gyroscopes. Solutions to these task usually include the use of engineered features obtained through heuristic processes.

HAR systems usually differ based on sensor choice and the prediction model used. Example of sensor choices range from on-body accelerometers to ambient, fixed sensors (e.g. proximity sensor or video cameras). Yang et. al. argue that wearable sensors are preferred over information acquired from video sources citing less limitations related to the environment as well as privacy concerns[25]. Commercial applications include fitness wristbands and fall detectors for impaired individuals.

There are various different approaches proposed in literature, which differ primarily in terms of sensor choice, the machine learning(ML) model used and the environment in which the activity information was gathered, as well as the level of preprocessing applied on the sensor data. We consider HAR to be a relevant and closely related topic to the problem of automating context graph generation. However, context graphs provide an in detail symbolic description of the user's state, while current HAR system usually produce a more brief representation.

This chapter will present several state of the art approaches to the problem of human activity recognition from sensor inputs. For each approach, we will discuss how it can be applied for the task of generating meaningful context graphs from sensor data.

2.1 Human Activity Recognition

As previously mentioned, considerable work was carried in the field of Human Activity Recognition. Several different directions can be identified:

1. Learning predefined activity models from labelled data
2. Online activity recognition
3. Unsupervised activity classification
4. Identifying unseen activities based on supervised learning of semantic attributes
5. Hierarchical structuring of activities using probabilistic models

2.2 Learning predefined activity models from labelled data

For the purpose of recognizing a set of predefined activities, various machine learning models have been tested. These can be broadly categorized into template matching, generative and discriminative approaches. Template matching techniques use a K-nearest neighbour classifier based on euclidean distance or dynamic time warping. Generative approaches like Bayesian classifiers model activity samples as Gaussian mixtures[15]. Discriminative approaches, including support vector machines (SVM) and conditional random fields have also been used with great success[14][21].

Krishnan et al. successfully used support vector machines classifiers to learn activity models on sensor data collected from smart home environments[13]. With sensor events stored as tuples <Date, Time, SensorId, Message>, the learning problem was to map a sequence of K sensor events to an activity label. Other work which used data from body sensor targeted the same learning problem, the difference being that a time window was chosen to sample the events used as inputs for the classifiers[4].

The experiments discussed in this paper fall roughly under this category, the datasets that were used both contain labelled activity data. However, we used this samples to generate training datasets for targets that were not explicitly marked in the datasets, as explained in Chapter 3.

2.3 Online Activity Recognition

The same method can be applied for online activity recognition, however the size of the reference window holds a definitive role in the performance of the classifier. There are three different methods proposed in current literature for handling the size of the window[14].

Explicit segmentation. This is usually done in a two step approach. In the first step, the streaming sensor events are split into chunks, with each chunk corresponding to a possible activity. The second step performs classification on the chunks resulted from the previous step.

Time based windowing. The second approach for handling the streaming of data is to divide the sequence into equally sized time frames. This approach simplifies the complexity of handling the data and proved to be efficient when dealing with sensors that operate continuously in time. This approach is common with accelerometers and gyroscopes, where data is sampled at regular intervals. The challenge for this approach is choosing the appropriate time frame. If the window is too small, there's a chance that relevant information will be left out. On the other hand, if the time frame is too big, data related to multiple activities can be included in the same frame, leading to poor classification performance.

Sensor event based windowing. The third approach is to divide the sequence into windows of equal sizes. This method may lead to cases where events from different activities are included in the same window. The relevance of events from the same window is uniformly distributed and a weighting mechanism should be used. Experiments by Krishnan et al. used this approach yielding good results [14].

This category falls outside the scope of this research, but the segmentation techniques are still valid for the problem of learning semantic attributes.

2.4 Unsupervized learning for activity patterns

A different approach for activity recognition is the use of clustering algorithms for grouping activities with similar characteristics. This approach was used to learn a taxonomy for activities

performed in different environments[13]. Hyunh et al. used soft clustering labels as the input features for learning high-level activity patterns using topic models[11].

This approach was not used, the nodes in a context graphs are well defined concepts, thus it is difficult to use clusters generated by unsupervised learning models to build these nodes.

2.5 Attribute-Based Learning

Cheng et al. [4] studied the problem of recognizing previously unseen activities using low level attributes of the activity. This task is often referred to as the *zero-shot learning problem*. The human activity recognition system designed by Chen et al. is comprised of two layers. The first layers maps sensor data to an attribute matrix, while the second layer uses a Bayesian classifier to determine the activity being performed. Experiments were carried on activities from two domains: exercise activities and daily life activities and were based on data from on-body inertial sensors. Features used for classification included mean standard deviation of sensor data, pair-wise correlation between pairs of dimension and local slope of sensor data. The attributes identified by the system were atomic steps of the action currently being performed and were previously selected by a supervisor. The system provides an uncertainty measure and prompts the user when an unseen activity was performed.

These results are also relevant for the problem of generating context graphs. The approach we used for learning semantic attributes, which are then used to generate context graphs, is done in a similar fashion with some exceptions. In the experiments presented in this paper, the value of attributes is extended to include consequences, descriptions and side-effects of the actions. Next, we also evaluate the approach using ambient sensors in the context of smart house systems.

2.6 Activity Patterns using topic models

Another approach based on algorithms from natural language processing was studied by Huynh et al. [11]. The proposed method uses topic modeling to recognize daily routines as probabilistic combination of activity patterns. Activities performed on a daily basis can be characterized on different levels of granularity, fine-grained activities tend to be correlated to physical movement and body posture. For many types of activities, a small window of sensor data is sufficient to recognize them. However, when trying to recognize higher level activities the task becomes more difficult because these activities are not identified by physical properties measured by sensors. The complexity stems from the following causes:

1. they are composed of variable patterns of multiple activities
2. they range over large periods of time
3. often vary significantly between instances.

Probabilistic models such LDA are very effective in capturing the correlating between these activities as higher level activity pattern. The results of this work are three-fold. First, the approach was validated using labeled activity data by measuring the activation level of each high-level pattern. Second, topic modeling proved successful as a mean of inferring high-level structure from a vocabulary of labels representing relatively short-term activities. The labels were learned using supervised learning from sensor data streams. What's interesting with this approach is that the estimated topics carry an inherent meaning. The drawback lies in the amount of effort associated with the supervised learning part. Third, evaluation of

applying topic modelling over a vocabulary learned using unsupervised learning algorithms yields surprisingly good results without any activity annotation.

This approach yields high level activity patterns which are comparable to context graph representations. When compared to the results published by Hynh et al.[11], the datasets used in this work produced a considerable smaller vocabulary, which is why we considered LDA not to be an effective solution. However, it would be interesting to evaluate LDA on datasets with more complex activities, which would have a more complex associated vocabulary.

2.7 Deep learning for HAR tasks

In recent years, deep learning has become one of the biggest trends in machine learning. With both commercial and academic interest in the subject, a significant amount of work was put into developing models that learn high level abstractions from data. Furthermore, with the development of frameworks such as torch7, TensorFlow and Theano, deep learning has become considerably more accesible to both researchers and software engineers. Current research suggests that deep learning models are well suited for solving human activity recognition tasks. In particular the use of convolutional neural network proved a good choice for automating feature extraction sensor inputs[25].

One of the main advantages of neural networks is that they behave very well when using raw signals as their input. One of the key tasks in HAR is engineering good feature extractors, traditional systems often relying on heuristics or expert knowledge. Using deep learning models has the potential of overcoming this limitation, greatly facilitating the process of optimizing model parameters in a systematic manner.

We used deep learning models for learning semantic attributes from on-body sensors. For the ambient sensors dataset, the number of available features was too small and using deep learning models didn't yield any improvements over traditional machine learning models.

2.7.1 Convolutional Neural Networks

The use of Convolutional Neural Networks, or CNN, for human activity recognition was studied by a number of authors[25] [22][26]. CNN are extremely effective at identifying salient patterns from input data. The lower level layers usually learn to match simple patterns that describe basic movement, with subsequent layers learning increasingly more abstract patterns within the data. Most CNN approaches employ the use of pooling layers after convolution layers. These have the effect of multiple salient features learned from different parts of the input being jointly considered for the classification. As such, convolutional layers can act as automatic feature extractors while pooling layers are used to obtain higher level features.

An important aspect is that in HAR problems, the input consists of multiple channel signals, over which traditional convolution filters cannot be used directly. Instead, the convolution filters and pooling layers are required to operate only along the temporal dimension.

2.7.2 Recurrent Neural Networks

An issue with traditional feedforward networks lies in the assumption that inputs are independent variables. To successfully model temporal dynamics in data, the input should include temporal information. The solution to this limitation is using recurrent neural networks (RNNs). In recurrent neural networks, the output of a node is fed back to itself with a delay using a

recurrent connection. This connection to the past activation allows RNN units to model temporal sequences. While in principle the RNN is a simple and powerful model, in practice, it is difficult to train properly. The main reason for this is the vanishing gradient problem [10].

2.7.3 Long short term memory

Long short term memory(LSTM) is a solution to the vanishing gradient problem encountered with RNNs. LSTM uses memory cells instead of recurrent units that store information using a gating mechanism and are capable of learning long-term dependencies with hundreds of steps. Each unit keeps track of its memory using 3 gates: output gate, input gate and forget gate. Figure 2.1 shows the gating mechanism used for a single LSTM unit.

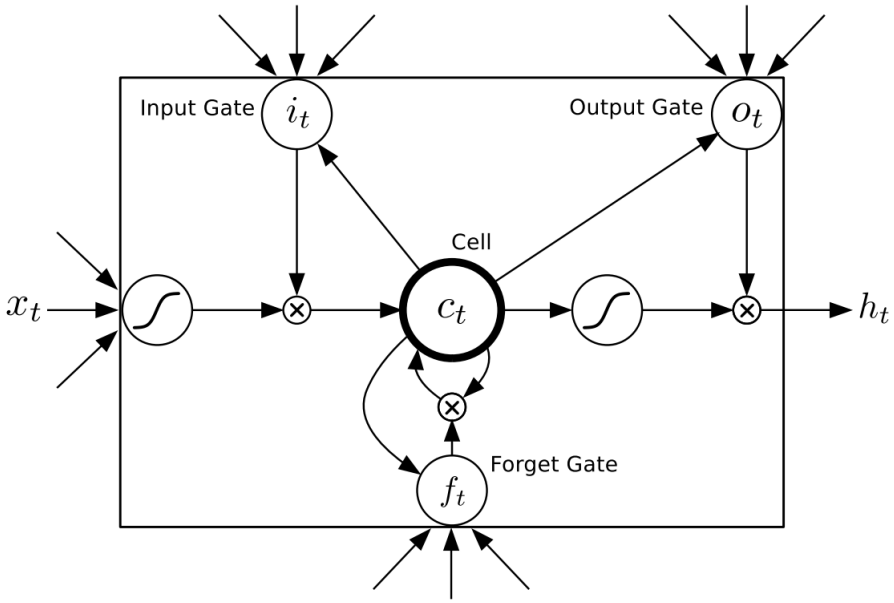

Figure 2.1: Unit inside an LSTM layer

Chapter 3

Generating context graphs using semantic attributes

In this chapter we present the steps for generating context graphs using the semantic attribute approach. We discuss the steps taken for training the classifiers that produce the list of semantic attributes and describe an example of how a rule-based system can transform these attributes into transactions on the context graph model.

3.1 Overview

As mentioned in Chapter 1, the first step is to select a list of semantic attributes that are relevant in a given AmI application's domain. While it is possible to build attributes sets that cover multiple situations that fall in the same domain, such as the domain of daily life activities, we consider this step to be specific to individual AmI applications and, as such, for the experiments presented in this work, the attributes are preselected.

With a given list of semantic attributes, the next step is training classifiers that recognize these attributes. We applied general machine learning techniques that are often used in human activity recognition tasks. There are some differences between the two types of sensor data sources: ambient sensors and on-body sensors. In the following sections we describe the steps for preprocessing, feature extraction and training of the classifiers. Finally, we describe how a rule based system can use these semantic attributes to generate a context graph.

3.2 Preprocessing

The preprocessing steps are slightly different for each of the two datasets. The ambient sensors dataset, WSU CASAS, contained mostly binary sensors which would generate a sample only when their output switched in value. For the continuous sensors (e.g. water-tap sensor) we performed feature scaling. The time between two sensor samples' timestamps is in the order of seconds. With the exception of the proximity sensors, which were connected wirelessly using bluetooth, all the sensors were hard-wired and didn't exhibit missing values. Because of this, the dataset doesn't include samples with empty values, but rather just includes small gaps with no events.

The on-body sensors dataset included only continuous value data which exhibited numerous missing values. We performed per-channel unity-based normalization, and filled in the missing

values using linear interpolation. The sample rate that was used in the experiments on the Opportunity dataset was 30Hz.

3.3 Feature extraction

For both types of experiments we applied a windowing technique. This means that features for a sample at time t is actually a sequence with data from samples from time $t-K$ to time t. The main difference between how feature extraction was applied on the two datasets stems from the samples' structure. For the WSU CASAS dataset, a sample would be a tuple containing the following items:

- timestamp.
- type of sensor.
- sensor value.

As noted in the previous section, the time difference between timestamps could vary significantly. The Opportunity dataset, on the other hand, contained vectors of raw values which were measured at a fixed sample rate. The Opportunity dataset contained also considerably more samples. As such the features extracted from the WSU CASAS dataset include:

- sequence of (type of sensor, sensor value)
- time offsets between the first sample in the sequence and all subsequent samples.

For the Opportunity the features used were comprised of a matrix of all sensor readings in a 800ms time frame.

3.4 Machine learning models

Depending on the dataset, we experimented with two types of classifiers. For WU CASAS we used linear regression and support vector machines. For the Opportunity dataset we used deep learning models with convolutional layers and long short term memory units. The deciding factor in this case was the amount of available data. For the first dataset, the size of the feature size was $[windowsize, 4]$ with 16000 entries, while for the second one the feature size was $[windowsize, 113]$ with 46000 entries. Because of this, using deep learning models on the WU CASAS didn't yield any improvements.

3.5 Training

Given an activity-attribute matrix, we train detectors which can infer the presence or absence of a given attribute from features extracted from the sensor data.

Training the classifiers poses a number of issues. First, the number of attributes can be quite large and collecting separate training datasets for each attribute is not practical. Next, low level attributes can be descriptions, consequences or just side effects of the activity a subject is performing. Finally, a lot of attributes are common to different types of activities. The solution is to reuse a dataset for high level activities and provide both negative and positive samples for each attribute classifier. Positive samples are obtained by merging labelled data of all activities associated with an attribute, while the negative samples are assembled using all the high level activities which are not associated with the attribute.

3.6 Graph generation

With a list of attributes provided by the trained classifiers, the final step is to feed this list into a tATAmI agent which translates it to changes in a context graph using a set of rules. As mentioned in Chapter 1 this step is solved on a case by case basis and is considered a task left for the AmI agent's developer. In this section, using a subset of the semantic attributes extracted from the Opportunity dataset, we provide an examples of how the set of rules can be defined. Table 3.1 lists the set of attributes used for this example.

Table 3.1: Example Attributes for generating context graphs

Attribute	Value range
posture	stand, walk, sit, lie
general_activity	relaxing, coffee time, early morning, cleanup, sandwich time.
holding_object_in_left_arm	bottle, salami, bread, sugar, dishwasher, switch, milk, drawer3, spoon.
holding_object_in_right_arm	knife cheese, drawer2, table, glass, cheese, chair, door1, door2, plate.

We consider the initial context graph to be a single node graph with the label *user*. Inspecting the attribute list, we notice that we can map each attribute name to an outward edge from the root node connected to a new node with label from the attribute's range of values. Next, the user cannot be performing two general activities at the same time, hold multiple objects in the same hand or have different postures at the same time. From this observation, we can infer an update rule for the graph for each attribute:

Given a context graph G and a list of $(attribute, value)$ pairs:
For each attribute, value pair in the input vector:
Let P = pattern constructed from the attribute's label, the root node and a generic node: $user - attribute- >?$
Let M = matching subgraph when comparing pattern P to the graph G.
If M is not *null* then:
Remove M from the context graph.
Let N = subgraph constructed from the root node, attribute's label and the attribute's value: $user - attribute- > value$
Add N to G.

In figure 3.1 we can view the results of applying the rule set on the following input:
posture=stand, *general_activity*=sandwich_time,
holding_object_in_left_arm=knife, *holding_object_in_right_arm*=cheese. Next, if the value of *holding_object_in_left_arm*, *holding_object_in_right_arm* attributes changed to *null* and *drawer_handle* the changes applied to the graph would be removing the edge associated with *holding_object_in_left_arm* and changing the label of the node that describes the object held in the right hand. The resulting graph is displayed in figure 3.2.

The *holding_ object_ in_ left* and *holding_ object_ in_ right_ arm* attributes could be interpreted differently. For example, they could be used together to build a single branch in the context graph for *using_ item*. We believe that there are multiple approaches on how the attributes can be translated into changes in the context graph, but this depends on the specifics of the AmI application in which they are used.

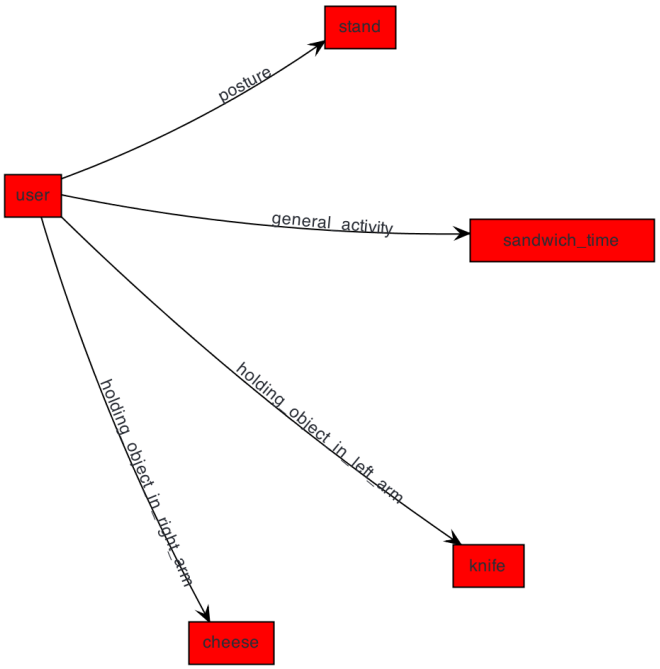

Figure 3.1: Example resulting context graph after applying the ruleset.

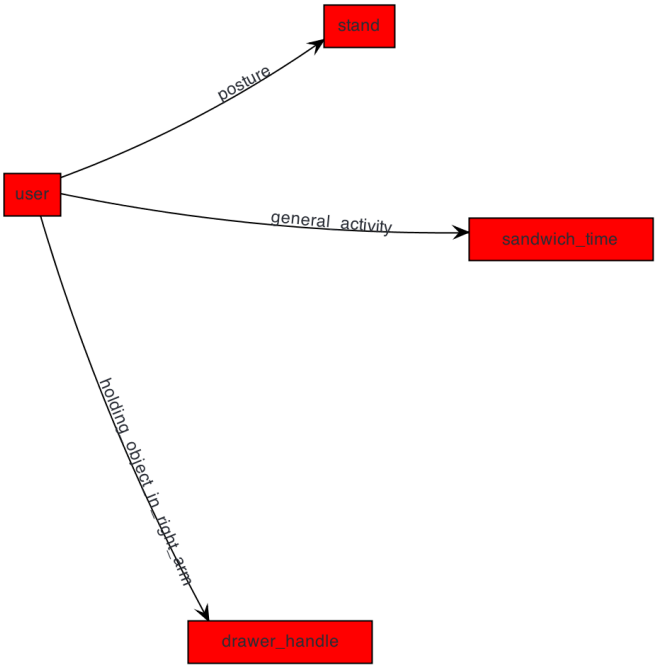

Figure 3.2: Example resulting context graph when the value of the $holding\_object\_in\_left\_arm$ attribute changed to $null$.

Chapter 4

Learning semantic attributes from ambient sensors

In this chapter we discuss the experiments carried for the task of extracting semantic attributes from ambient sensor data resulted from recording test subjects performing daily life activities. As mentioned in Chapter 3, in order to produce semantic attributes from data, we select a list of semantic attributes that can be mapped in a context graph and train classifiers for each of these attributes. In the following sections we describe the WSU CASAS[6] dataset, discuss the selection of attributes and detail how the learning was performed.

4.1 The WSU CASAS Dataset

The experiment was carried on a dataset from the WSU CASAS smart home project[6]. The dataset consists of sensor events for 8 annotated activities from the domain of daily life activities. Participants perform each activity separately and then are asked to perform the entire set of 8 activities in any order they prefer.

The following is a list of the 8 activities the participants were asked to perform:

1. **Fill medication dispenser**. The participant moves to the kitchen, retrieves a pill dispenser and bottle of pills, and follows directions to fill the dispenser.
2. **Watch DVD**. The participant moves to the living room, puts a DVD in the player and watches a news clip on TV.
3. **Water plants**. The participant retrieves a watering can from the kitchen supply closet and waters three plants.
4. **Answer the phone**. The phone rings and the participant answers it. The participant converses over the phone with the experimenter to answer some questions about the news clip they watched.
5. **Prepare birthday card**. The participant fills out a birthday card with a check to a friend and addresses the envelope.
6. **Prepare soup**. The participant moves to the kitchen and prepares a cup of noodle soup in the microwave, following the directions on the package. The participant brings the soup and a glass of water to the dining room table.

7. **Clean**. The participant sweeps the kitchen floor and dusts the living room and dining room using supplies retrieved from the kitchen supply closet.
8. **Choose outfit**. The participant selects an outfit from the clothes closet that their friend will wear for a job interview.

The dataset includes approximately 16000 events from sensors installed around a smart house environment. Most of the sensors types have binary output. Table 4.1 shows a complete list of the sensor types used. Figure 4.1 shows the positioning of the sensors inside the smart apartment, also the path of participant performing task 7 can be observed.

Table 4.1: Sensor types

Sensor	Output type	Notes
motion sensors	binary	
item sensors	binary	for oatmeal, raisins, brown sugar, bowl, measuring spoon
medicine container sensor	binary	
pot sensor	binary	
phone book sensor	binary	
cabinet sensor	binary	
water sensor	continuous	
water sensor	continuous	
burner sensor	continuous	
phone sensor	binary	
temperature sensors	continuous	

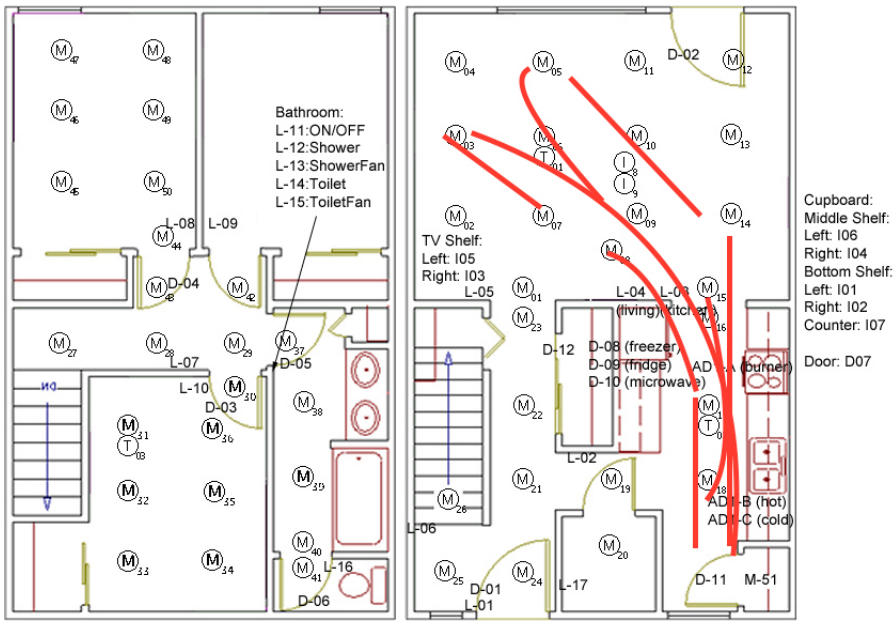

Figure 4.1: WSU CASAS smart apartment layout and path of the subject performing task 7.

4.2 Learning semantic attributes

Based on the activity descriptions mentioned above, we encoded three types of semantic attributes which are listed in table 4.2. Inspecting the table of attributes, we can observe that they map directly to nodes and edges in the context graph using a rule set similar to the one presented in Chapter 3.For example, the activity of cleaning the kitchen is characterized by $location-is-kitchen = true$, $activity-is-cleaning = true$ and $using-cleaning-supplies = true$, with all other attributes being false. The choice of location as a semantic attribute might be put into question, because location correlated directly to raw sensor inputs (since their position in the house is known). However, our goal is to validate the usefulness of semantic attributes for context graph generation and not to produce the very best output from the given dataset. The mapping between each attribute and the set of 8 activities can be defined as a [N, 8] matrix.

For each pair of (attribute, value), we trained a binary classifier on the annotated single-activity datasets and measured the performance on the interwoven datasets. We tested two different classifiers: support vector machines (SVM) and logistic regression. Early tests revealed similar performance between the two and we decided to use latter for the final evaluation. We used the logistic regression and SVM implementation from the python package sklearn[1].

Table 4.2: Attributes for the WSU CASAS ambient sensors dataset

Attribute	Value range
location	kitchen, living-room, hallway
using	pill-dispenser, dvd, tv, can-of-water, sink, phone, phone-agenda, microwave, pot, cleaning-supplies, closet
action	filling-medication-dispenser, watching-dvd, watering-flowers answering-phone, prepare-birthday-card, prepare-soup, clean-kitchen, clean-living-room, choose-outfit.

4.2.1 Feature extraction

The features used in training the classifiers are extracted from sequences of $K = 15$ sensor events. This value was chosen empirically after some initial tests. The extracted features include: type of the sensor, sensor value, position in the sequence, time offsets between the first event and all subsequent events in the sequence.

4.2.2 Training

Because the dataset contains annotations only for the general activity being performed, we lacked training sets for each of the binary classifier. To overcome this we used the dataset to provide both negative and positive samples for each of the training targets. By merging the samples of all activities associated with an attribute we obtained a positive training set. The set of negative samples was assembled in a similar fashion, by merging the samples of activities not associated with those attributes.

[1] http://scikit-learn.org/stable/

Chapter 5

Learning semantic attributes from on-body sensors

In this chapter we discuss the task of learning semantic attributes from on-body sensors. We use the Opportunity dataset to train and evaluate the performance of deep neural network classifiers for 7 types of attributes.

One of the main advantages of neural networks is that they behave very well when using raw signals as their input. As mentioned in Chapter 3, the task of learning attributes is very similar to HAR problems. One of the key tasks in HAR is engineering good feature extractions, traditional systems often relying on heuristics or expert knowledge. Using deep learning models has the potential of overcoming this limitation, greatly facilitating the process of optimizing model parameters in a systematic manner.

5.1 The Opportunity challenge dataset

The OPPORTUNITY dataset[1][3] consists of a set of daily user activities performed in an environment measured through a high number of sensors. The data was recorded using four subjects in a daily living scenario and amounts to 6 hours of recorded material. Each subject performed 5 sessions with daily life activities plus an extra drill session. When performing the activities, subjects followed a loose description of the task with no restrictions. Examples of activities include: 1. preparing and drinking coffee; 2. preparing and eating lunch; 3. cleaning the table; The drill sessions consisted of twenty repetitions of predefined sorted lists of activities. Activities performed during the drill sessions display a considerable amount of overlap, the subjects being give a lot of leeway in the manner in which they performed the activities. Figure 5.1 shows a view of the room in which the activities were recorded.

Regarding sensor setup, the set of on-body sensors used includes 5 RS485-networked XSense inertial measurement units attached to a special jacket worn by the subjects, 2 commercial InertiaCube3 inertial sensors located on the subjects' feet and 12 bluetooth acceleration sensors on the limbs. Each measurement unit is composed of a 3D accelerometer, a 3D gyroscope and a 3D magnetic sensor. Figure 5.2 offers a representation of the sensors' position. The dataset was used in an open challenge for human activity recognition. As part of the challenge, each sensor axis is treated as an individual channel. Therefore, the input space consists of 113 different channels with a sample rate of 30 Hz[3].

The Opportunity dataset includes annotations for several different challenges:

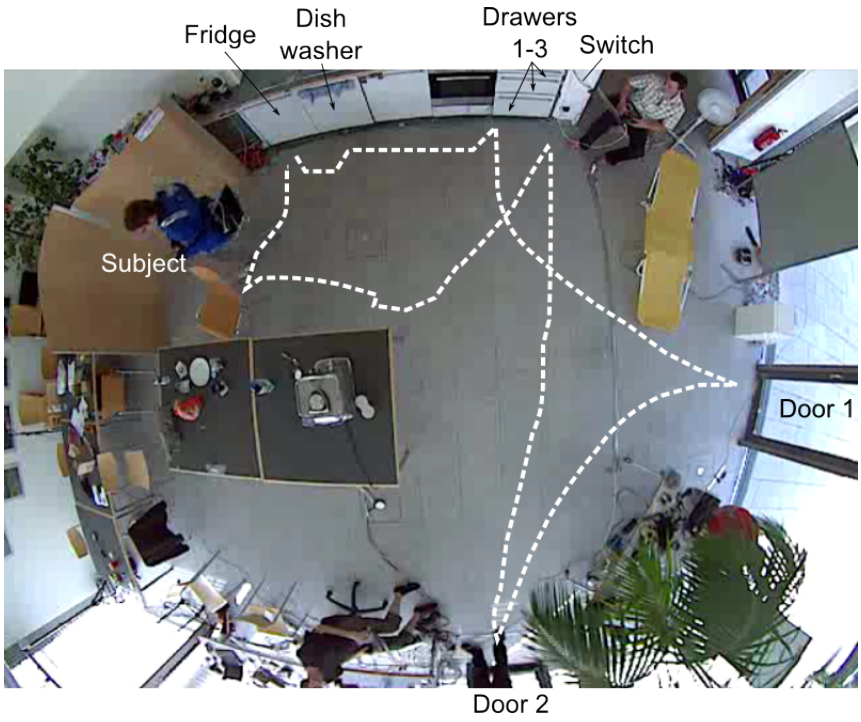

Figure 5.1: View of the room in which the activities were recorded. Dashed lines mark the subjects' trajectory within the room. Picture from Chavarriaga, R., et al.[3]

1. recognition of modes of locomotion and postures. 5-class segmentation and classification problem.
2. hand gesture recognition. This consists of different right-arm actions. 18-class segmentation and classification problem.

These challenges received numerous submission for the human activity recognition tasks. While our target was to learn semantic attributes, we leveraged existing results to design the architecture of a deep neural network that delivers state of the art results on the gesture recognition tasks. We then used this architecture for the semantic attributes classifiers.

The current state of the art model for the gesture recognition model was reported by Hammerla et. al.[9] with F1 score of 0.92 on the Opportunity gesture recognition task. The model uses 3 bidirectional LSTM layers applied on the normalized sensor data using a sliding window mechanism. The window size used was 1 second, as opposed to a window size 800ms used in our experiments.

5.2 Semantic attributes from the Opportunity dataset

As per the steps presented in Chapter 3 we extracted a list of semantic attributes for which we trained individual classifiers. A full list of these attributes can be viewed in table 6.4. Unlike the WSU CASAS dataset, the Opportunity dataset has rich annotations for the actions performed, thus we didn't need to generate positive and negative samples for each classifier.

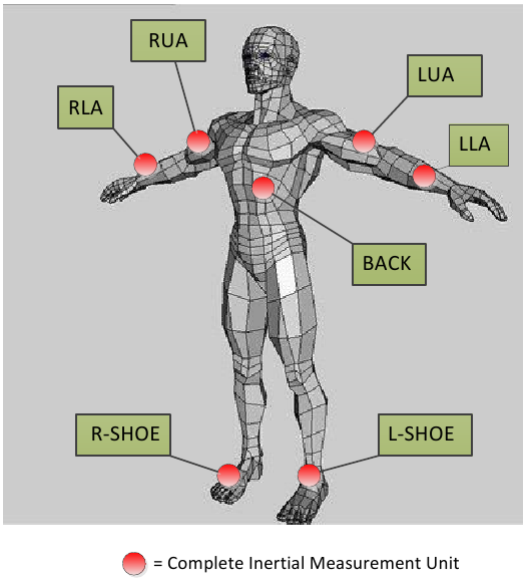

Figure 5.2: On-body sensor position used for activity recognition data in the Opportunity dataset. Picture from Chavarriaga, R., et al.[3]

Table 5.1: Attributes extracted from the Opportunity dataset

Predicate	Value range
posture	stand, walk, sit, lie
general_activity	relaxing, coffee time, early morning, cleanup, sandwich time
left_arm_action	unlock, stir, lock, close, reach, open, sip, clean, bite, cut, spread, release, move
holding_object_in_left_arm	bottle, salami, bread, sugar, dishwasher, switch, milk, drawer3, spoon, knife cheese, drawer2, table, glass, cheese, chair, door1, door2, plate, drawer1, fridge, cup, knife salami, lazychair
right_arm_action	unlock, stir, lock, close, reach, open, sip, clean, bite, cut, spread, release, move
holding_object_in_right_arm	bottle, salami, bread, sugar, dishwasher, switch, milk, drawer3, spoon, knife cheese, drawer2, table, glass, cheese, chair, door1, door2, plate, drawer1, fridge, cup, knife salami, lazychair
performing_activity	open door 1, open door 2, close door 1, cloose door 2, open fridge, close fridge, open dishwasher, close dishwasher, open drawer 1, close drawer 1, open drawer 2, close drawer 2, open drawer 3.

5.3 Deep Neural Network Architecture for Learning Semantic Attributes

In this section we introduce a deep learning model for learning semantic attributes which uses a combination of convolutional layers and LSTM layers referred to as CNN_LSTM in the following sections.

The purpose of the convolutional layers is to automatically extract higher level features from the sampled data, while the recurrent layers attempt to model the temporal dependency within the time series. For comparison, we also present a baseline model that only uses convolutional layers(baseline CNN) and a second baseline model that only uses recurrent layers(baseline LSTM). All three models have the same configuration on layers of the same type and use a fully connected (dense) layer with 18 units as the output layer. Figure 5.3 offers a rough representation of the CNN_LSTM network.

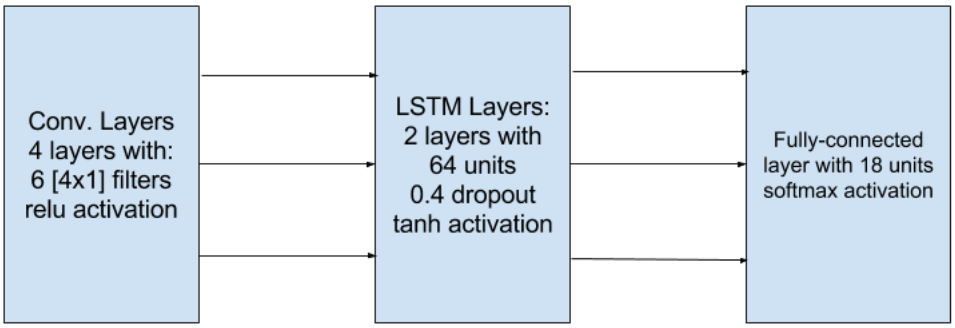

Figure 5.3: Architecture of the deep neural network for learning semantic attributes

As mentioned in the beginning of this chapter, we designed the CNN_LSTM network by iterating over layer configurations that were evaluated on the gesture recognition task from the Opportunity challenge.

5.3.1 Network input and preprocessing

The input of the network is bidimensional vector of size [24 x 113] corresponding to a 800ms time frame with 113 sensor channels. As mentioned in the previous section, the input data was segmented using a sliding window method with 50% overlap. As it is often the case in real world scenarios, sensor channels suffer from noise or missing output. This was an issue with the bluetooth sensors in particular. To fill missing entries in the dataset we used linear interpolation and performed (per channel) input normalization to the [0,1] interval.

5.3.2 Convolutional layers

Our network uses 4 convolutional layers. Each layers applies 6 [4x1] filters. Note that the first dimension represents time, thus the filters will be applied for each sensor channel individually. The activation function used for these layer is rectified linear unit(ReLU) $relu(x) = max(0, x)$.

We experimented with adding pooling layers between the convolutional layers. However, no performance increased was observed. This can be explained by the presence of the LSTM layers, with temporal convolutional filters performing the same task as the LSTM layer, but on considerable smaller time intervals. The network's output after the convolutional layer consists of 5 feature maps of size [24, 113].

5.3.3 Recurrent layers

On top of the convolutional layers our network stacks 2 LSTM layers with 64 units each. The activation function between the recurrent layer cells is tanh. When training the network for more than 45 epochs, it starts to manifest some degree of overfitting. To counter this issue, we experimented with adding dropout to each recurrent layer's input. The results reported in this paper were obtained using a dropout rate of 0.4 on the inputs of the recurrent layers.

5.3.4 Training

The network output is computed using the Softmax activation function from the top level fully connected (dense) layer. Training was performed in a supervised fashion, minimizing the output of the categorical cross entropy loss function. We used batch gradient descent with rmsprop[7]. All models were trained using batches of 100 samples.

After performing segmentation on the training data, we obtained 46000 training samples. This training set is considered small for deep learning models. Experimenting with deeper neural networks lead to overfitting problems. We followed the general guidelines from existing deep learning research[24] and applied dropout factors greater than 0.5 for the lower layers and between 0.4 and 0.5 for higher layers. Other methods of regularization reported in existing research include max norm regularization[9] and L1 regularization[20].

5.3.5 Implementation

To implement the CNN_LSTM network and the two other baseline networks we used two popular libraries: Keras[5] with the TensorFlow[2] backend. TensorFlow is an open source library for numerical computation using data flow graphs. Keras is machine learning library built on top of TensorFlow and Numpy which facilitates prototyping with deep learning tasks. Training and classification were executed using libcudnn on an Amazon Web Service instance which provided a K520 series GPU with 1500 cuda cores running at 850MHz and 4GB video ram.

Chapter 6

Results evaluation

In this chapter we present the experimental results for the task of learning semantic attributes using human activity recognition models on the two datasets we described in Chapter 4 and Chapter 5. Furthermore, we compare the deep learning model that was trained on the Opportunity dataset with existing models from research published for the open challenge associated with the dataset. As previously mentioned, this comparison was the method used to validate the neural networks' architecture before applying it on the semantic attribute learning task. Before presenting the results we provide some observation on the metrics used for performance measuring.

6.1 Null class and class imbalance

A common issue with human activity datasets is that they are often highly unbalanced. Class imbalance means that a small set of classes are represented by a large number of samples, while others are underrepresented[12]. For activity recognition problems, the most widely encountered source of imbalance is the Null class; the Null class is used to label samples in which the subject is not performing any particular activity. Because the semantic attributes we extracted from the datasets are directly correlated with activities being performed, this problem also applies to our experiments. For the Opportunity dataset, the Null class represents more than 75% of the recorded samples. The exact percentages for the first 3 subjects are: 76%, 82% and 76%. As such, using classification accuracy is a poor choice for measuring performance. Using a simple classifier that just returns the Null class for any input will score a very high accuracy. The suggested method for measuring classifiers' performance in this case is the F_1 score. The F_1 score combines precision and recall, measurements that are often used in information retrieval.

$$F_1 = \sum 2 \cdot w_i \frac{precision_i \cdot recall_i}{precision_i + recall_i}$$

where i is the class number and w_i is the weight of class i within the dataset; $precision_i$ and $recall_i$ are the measurements for class i and are defined as follows:

$$precision = \frac{true\_positives}{true\_positives + false\_positives}$$
$$recall = \frac{true\_positives}{true\_positives + false\_negatives}$$

6.2 Results: learning semantic attributes from ambient sensors using the WSU CASAS dataset

To measure the performance of the classifiers we computed the precision, accuracy, recall and F1 score for attributes predicted by the SVM classifiers. The results are displayed in figure 6.1. All classifiers achieved a high accuracy score, however this is mostly due to the large number of true negative samples for each classifier. As mentioned in the previous section, the relevant metric is the F1 score. All classifiers achieved a precision greater than 55% which improve over the baseline random classifier which achieves a precision of 32% on average. Recall ranged from 0.29 to to 0.96, depending on how often the attribute was represented in the generated training samples. As mentioned in the previous section, the relevant measure is the F1 score. The complete results, including the F1 scores are presented in table 6.1.

The best results were for location attribute, this due to the fact that all activities included at least one positive value for this attribute.

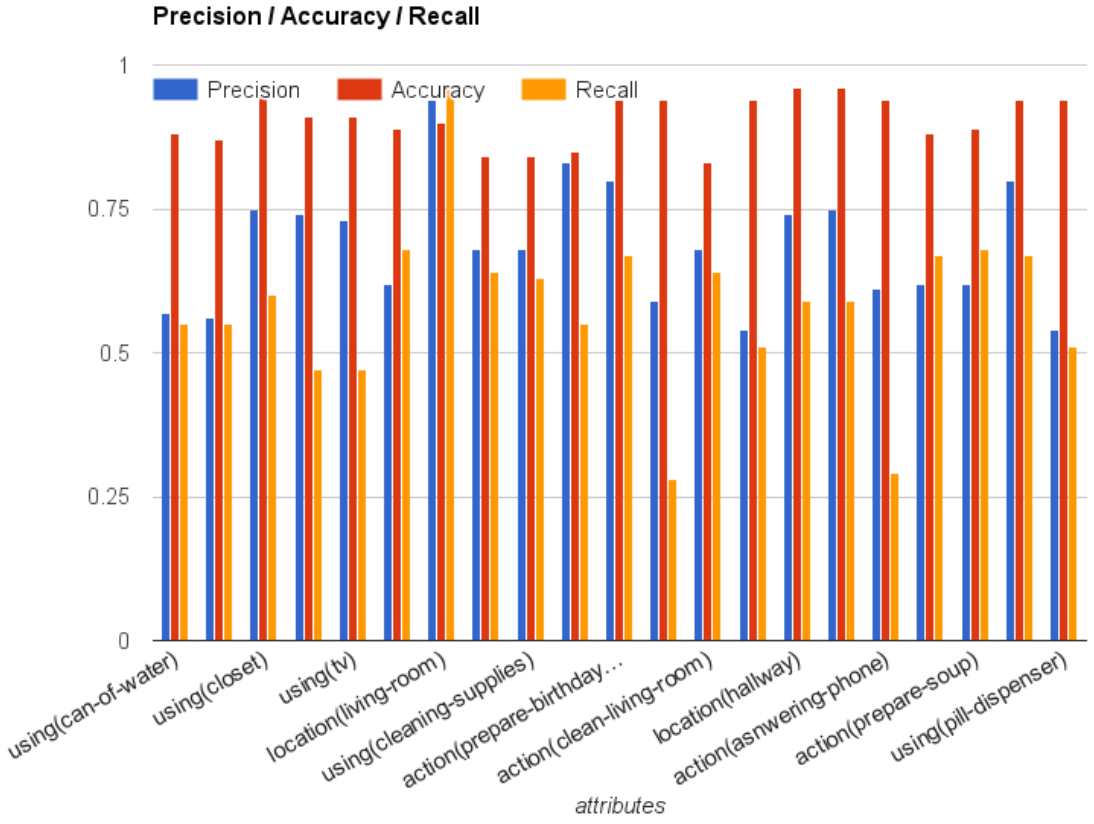

Figure 6.1: Performance measure for attribute classification

Inspecting table 6.1 we can observe that the classifiers achieved a F1 score greater 0.5 on average. This is consistent with the results published by Krishnan et al.[13] on the activity recognition task of 0.54 F1 score. However, for real world applications this is a rather poor result. We consider the main cause for these results to be attributed to the small number of features available for learning.

Table 6.1: Complete performance measurements for the WSU CASAS dataset

Attribute - Value pair	Precision	Accuracy	Recall	F1 score
using(can-of-water)	0.57	0.88	0.55	0.55
action(watering)	0.56	0.87	0.55	0.55
using(closet)	0.75	0.96	0.6	0.66
watching(dvd)	0.74	0.91	0.47	0.57
using(tv)	0.73	0.91	0.47	0.57
using(microwave)	0.62	0.89	0.68	0.64
location(living-room)	0.94	0.9	0.96	0.94
action(clean-kitchen)	0.68	0.84	0.64	0.65
using(cleaning-supplies)	0.68	0.84	0.63	0.65
location(kitchen)	0.83	0.85	0.55	0.66
action(prepare-birthday-card)	0.8	0.94	0.67	0.72
using(phone)	0.59	0.94	0.28	0.37
action(clean-living-room)	0.68	0.83	0.64	0.65
action(filling-medication-dispenser)	0.54	0.94	0.51	0.52
location(hallway)	0.74	0.96	0.59	0.656
action(choose-outfit)	0.75	0.96	0.59	0.66
action(asnwering-phone)	0.61	0.94	0.29	0.39
using(pot)	0.62	0.88	0.67	0.64
action(prepare-soup)	0.62	0.89	0.68	0.64
using(phone-agenda	0.8	0.94	0.67	0.72
using(pill-dispenser)	0.54	0.94	0.51	0.524

6.2.1 Results: CNN_LSTM deep neural network for human activity recognition on the Opportunity dataset

As mentioned in Chapter 5 we validated the architecture of the CNN_LSTM neural network on the gesture recognition task from the Opportunity dataset challenge. In this sections we present the results for this task and benchmark it against two baseline models, other submission to the challenge as well as the state of the art solutions published by different authors. Table 6.3 contains descriptions of the models used as benchmarks.

From table 6.2 we can see the CNN_LSTM model outperforms the Opportunity challenge submission and the baseline CNN and LSTM models. When compared to the best non deep learning submission, CStar, the performance increase in the F1 score is around 2%. The CNN_-LSTM also improves over the baseline models with 2-3%.

The baseline models have better performance than the non deep learning models submitted in the Opportunity challenge with one notable exception: CStar. The performance of the baseline CNN network has consistent results with the CNN model reported by Yang. et. al.[25].

The 2-layer LSTM baseline model performed slightly better than the convolutional only network. CNN_LSTM performing better than both baseline methods is evidence that recurrent neural networks are a good choice for modelling on-body sensor data.

However, CNN_LSTM falls short in comparison with the current state of the art by 0.1 on the F1 score. This can be attributed to a number of factors:

1. LSTM[9] uses 3 LSTM layers with bi-directional connections, while our model only uses forward (in time) connections for the LSTM layers.
2. LSTM[9] was trained on a slightly longer window frame.

Table 6.2: F1 score on the Opportunity gesture recognition task

Opportunity challenge submissions[3]	
Model	F1 score
LDA	0.69
QDA	0.53
NCC	0.51
1NN	0.87
3NN	0.85
UP	0.64
NStar	0.84
SStar	0.86
CStar	0.88
Deep learning approaches	
CNN[Yang et. al. 2015]	0.851
LSTM[Hammerla et. al. 2016]	**0.92**
Baseline CNN	0.876
Baseline LSTM	0.881
CNN_LSTM	**0.9096**

Table 6.3: Opportunity dataset benchmark for the gesture recognition tasks

	Opportunity challenge submissions[3]
Model	Description
LDA	Linear discriminant analysis. Gaussian classifier, assumes features are normally distributed and all classes have the same covariance matrix.
QDA	Quadratic discriminant analysis. Similar to LDA, but class covariance may differ.
NCC	Nearest centroid classifier. Uses euclidean distance.
1NN	K-nearest neighbour algorithm with k=1.
3NN	K-nearest neighbour algorithm with k=3.
UP	Submission to the Opportunity challenge from University of Parma. Uses mean, variance, maximum and minimum values for comparing patterns.
NStar	Submission from the University of Singapore. K-nearest neighbour algorithm with k=1, uses normalized data.
SStar	Submission from the University of Singapore, based on support vector machines with normalized data.
CStar	Submission from the University of Singapore. Uses both K-nearest neighbour and SVMs.
Deep learning approaches	
CNN[25]	Results reported by Yang et. al.[25].
LSTM[9]	Results reported by Hammerla et. al.[9]. Current state of the art on the Opportunity dataset.

It is unclear what is the size of LSTM layers used by Hammerla et. al. in their model. As such, we can only speculate on the difference in performance between the two network. Even so, CNN_LSTM achieves a very good F1 score on the dataset that is comparable to the current state-of-the-art.

6.2.2 Overfitting

As mentioned in Chapter 5, one of the main challenges when training the deep neural network was overfitting. When training CNN_LSTM, after 30 epochs, we can observe how the performance on the validation set plateaus at 0.90 (F1 score), while the model starts overfitting the training data. Figure 6.2 displays this issue.

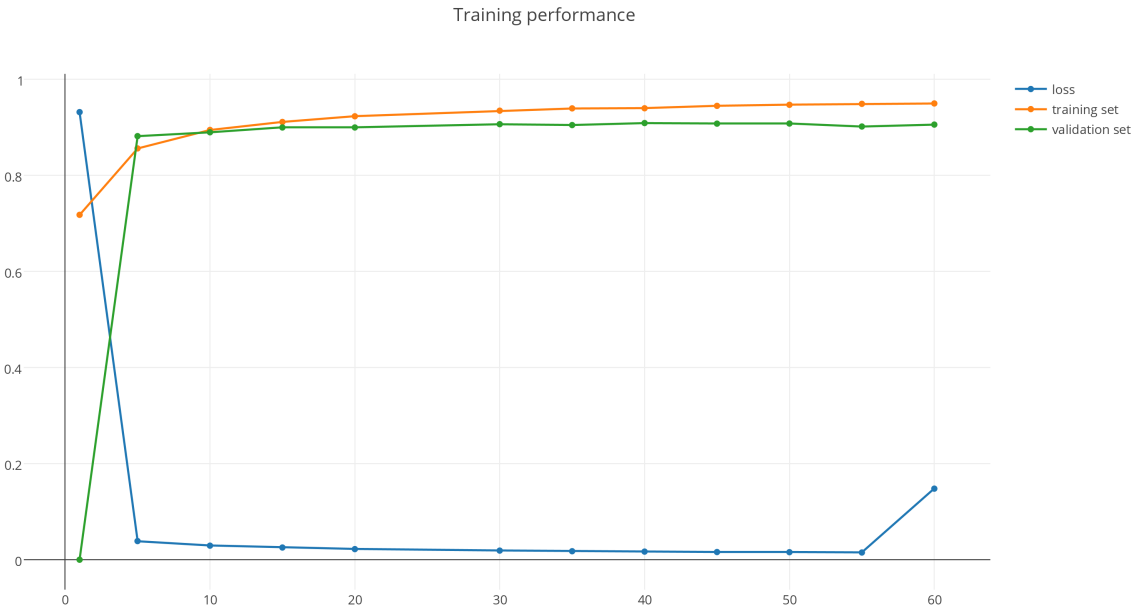

Figure 6.2: Plot with performance on the validation set during training. After 30 epochs, the performance on the validation set plateaus at 0.90 while model starts overfitting.

6.2.3 Using deeper networks

We also investigated the effect of adding more convolutional layers to the network. From our experiments, we found that after 4 layers, the performance of the network no longer increases and overfitting becomes more difficult to control. This can be observed in figure 6.3.

Another direction we experimented with was using a higher number of filters in the convolutional layers. We tested networks with filters between 10 and 32. However, we were unsuccessful in increasing the network's performance.

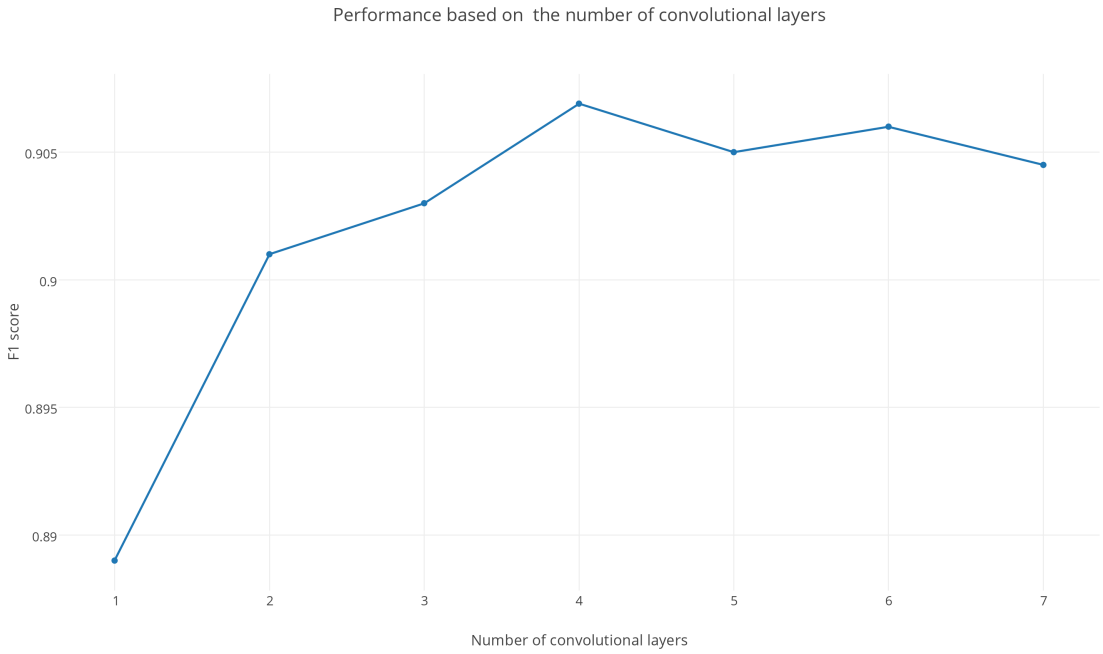

Figure 6.3: Performance of the neural network in relation to the number of convolutional layers used.

6.2.4 Results: CNN_LSTM deep neural network performance on learning semantic attributes from the Opportunity dataset

For the task of learning semantic attributes from the Opportunity dataset using the CNN_-LSTM deep neural network described in Chapter 5 we measured the accuracy and F1 score for 7 classifiers corresponding to the list of extracted attributes. As explained in the beginning of this chapter, because of the class imbalance, even trivial models achieve accuracy measures greater than 75%. In this particular case, the deep neural models achieved accuracy scores of 99%. The F1 score measures for each attribute classifiers can be viewed in table 6.4. For the *posture* and $performing\_activity$ attributes, the scores are consistent with the results on the gesture recognition task from the open challenge. For the attributes describing low level characteristics of the performed activities such as $left\_arm\_action$, $holding\_object\_in\_left\_arm$ etc., The performance is considerably lower with an average F1 score of 0.75. We attribute this to the difficulty of correlating the data with such fine characteristics of the activities. Finally, the classifier for the $general\_activity$ attribute also had lower performance with a score of 0.76. This result can be attributed to the fact that the time frame was too small to effectively identify this attribute.

We also measured the performance of the classifiers on the validation set during training. A plot of the measurements can be observed in Figure 6.4. The findings are consistent with the overfitting problem mentioned in previous sections.

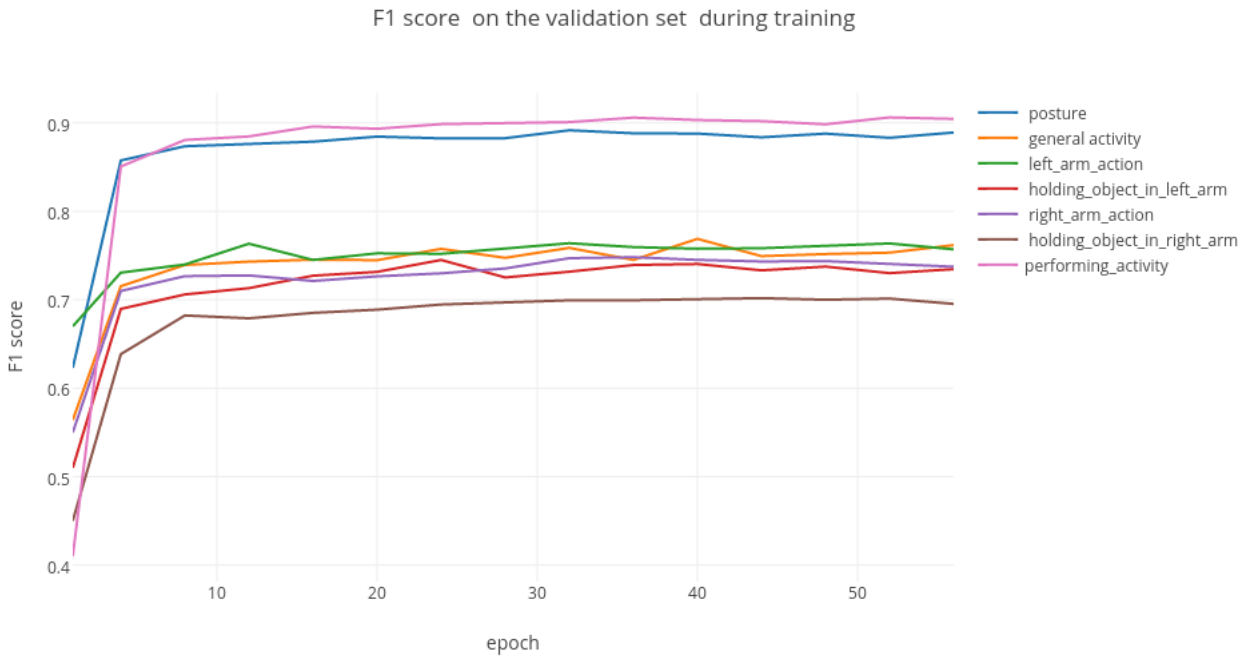

Figure 6.4: Performance of the network on the validation during training

Table 6.4: Classifiers' performance on semantic attributes extracted from the Opportunity dataset

Attribute	F1 score
posture	0.8916
general_activity	0.7688
left_arm_action	0.763
holding_object_in_left_arm	0.745
right_arm_action	0.748
holding_object_in_right_arm	0.701
performing_activity	0.905

6.3 Results interpretation

Regarding the task of extracting semantic attributes from sensor data, results from both experiments yielded promising results that were consistent with results published in human activity recognition literature on the corresponding datasets. Through these experiments we provided evidence that semantic attributes can be extracted from sensor data, which could then be used to generate context graph using the method detailed in Chapter 3. We also observe, that deep neural networks with LSTM units are well suited for capturing temporal dynamics from sensor data.

Chapter 7

Conclusion

In this paper we proposed a method for generating context graphs using semantic attributes extracted from sensor data. We discussed the architecture of machine learning models capable of identifying these attributes from both ambient and on-body sensors and evaluated their performance on two popular human activity recognition datasets: WSU CASAS and Opportunity. We presented some of the main difficulties in training these models and provided options to overcome them.

For the Opportunity dataset, we designed a deep neural network architecture that uses both convolutional layers and LSTM units. This architecture was validated on the gesture recognition task from the open challenge associated with the dataset. Our network achieved performance comparable to current state of the art submission from this challenge.

Our findings provide evidence that extracting semantic attributes from sensor data is feasible and, using the algorithm proposed in Chapter 3, that these attributes can then be used to easily generate context graph representations.

7.1 Future work

During the course of work we have identified several research directions that could be pursued in the future.

First, the experiments presented in this paper are all based on labelled data and use supervised machine learning approaches. It would be interesting to study how unsupervised learning algorithms could be used to synthesize symbolic knowledge from human activity sensor data.

Second, our datasets consisted of readings from a large number of sensors, for the on-body sensors experiment the subject had 19 accelerometers and gyroscopes attached to his body. We believe it is worth investigating setups that more closely resemble real world scenarios, for example measuring the performance of the attribute classifiers on readings from only two on-body sensor: a smart wristband and a phone.

Finally, context graphs models have the potential to cover a wider range of activity domains. It would be worth studying the performance of our approach on sensor data from other activity domains.

Bibliography

[1] Opportunity dataset.2012. available online:https://archive.ics.uci.edu/ml/datasets/opportunity +activity+recognition (accessed 25 may 2016).

[2] TensorFlow. https://www.tensorflow.org/.

[3] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R Millán, and Daniel Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013.

[4] Heng-Tze Cheng, Feng-Tso Sun, Martin Griss, Paul Davis, Jianguo Li, and Di You. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 361–374, New York, NY, USA, 2013. ACM.

[5] F Cholet. Keras. https://keras.io (accessed 24 may 2016).

[6] Diane J Cook. Learning setting-generalized activity models for smart spaces. *IEEE intelligent systems*, 2010(99):1, 2010.

[7] Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*, 2015.

[8] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.C. Burgelman. Scenarios for ambient intelligence in 2010. Technical report, Office for Official Publications of the European Communities, February 2001.

[9] Nils Y Hammerla, Shane Halloran, and Thomas Ploetz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

[10] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[11] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19. ACM, 2008.

[12] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[13] Narayanan Krishnan, Diane J Cook, and Zachary Wemlinger. Learning a taxonomy of predefined and discovered activity patterns. *Journal of ambient intelligence and smart environments*, 5(6):621–637, 2012.

[14] Narayanan C Krishnan and Diane J Cook. Activity recognition on streaming sensor data. *Pervasive and mobile computing*, 10:138–154, 2014.

[15] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *IJCAI*, volume 5, pages 766–772. Citeseer, 2005.

[16] George Wamamu Musumba and Henry O Nyongesa. Context awareness in mobile computing: A review. *International Journal of Machine Learning and Applications*, 2(1):5–pages, 2013.

[17] Andrei Olaru. Context matching for ambient intelligence applications. In Nikolaj Björner, Viorel Negru, Tetsuo Ida, Tudor Jebelean, Dana Petcu, Stephen Watt, and Daniela Zaharie, editors, *Proceedings of SYNASC 2013, 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, September 23-26, Timisoara, Romania*, pages 265–272. IEEE CPS, 2013.

[18] Andrei Olaru, Adina Magda Florea, and Amal El Fallah Seghrouchni. Graphs and patterns for context-awareness. In Paulo Novais, Davy Preuveneers, and Juan Corchado, editors, *Ambient Intelligence - Software and Applications, 2nd International Symposium on Ambient Intelligence (ISAmI 2011), University of Salamanca (Spain) 6-8th April, 2011*, volume 92 of *Advances in Intelligent and Soft Computing*, pages 165–172. Springer Berlin / Heidelberg, 2011.

[19] Andrei Olaru, Adina Magda Florea, et al. A graph-based approach to context matching. *Scalable Computing: Practice and Experience*, 11(4):393–399, 2010.

[20] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[21] Parisa Rashidi, Diane J Cook, Lawrence B Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE transactions on knowledge and data engineering*, 23(4):527–539, 2011.

[22] Charissa Ann Ronao and Sung-Bae Cho. Deep convolutional neural networks for human activity recognition with smartphone sensors. In *Neural Information Processing*, pages 46–53. Springer, 2015.

[23] Ivona-Emma Sevastian. Agentbased android application for conference participants. (unpublished bachelor thesis). 2014.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[25] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina*, pages 25–31, 2015.

[26] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Juyong Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 197–205. IEEE, 2014.