Adapting Activity Recognition to a Person with Multi-Classifier Adaptive Training

Božidara Cvetković [a,b], Boštjan Kaluža [a,b], Matjaž Gams [a,b], and Mitja Luštrek [a,b]
[a] *Department of Intelligent Systems, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana Slovenia*
E-mail: {boza.cvetkovic, bostjan.kaluza, matjaz.gams, mitja.lustrek}@ijs.si
[b] *Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia*

Abstract. Activity-recognition classifiers, which label an activity based on sensor data, have decreased classification accuracy when used in the real world with a particular person. To improve the classifier, a Multi-Classifier Adaptive-Training algorithm (MCAT) is proposed. The MCAT adapts activity recognition classifier to a particular person by using four classifiers to utilise unlabelled data. The general classifier is trained on the labelled data available before deployment and retrieved in the controlled environment. The specific classifier is trained on a limited amount of labelled data belonging to the new person in the new environment. A domain-independent meta-classifier decides whether to classify a new instance with the general or specific classifier. The final, second meta-classifier decides whether to include the new instance into the training set of the general classifier. The general classifier is periodically retrained, gradually adapting to the new person in the new environment. The adaptation results were evaluated for statistical significance. Results showed that the MCAT outperforms competing approaches and significantly increases the initial activity-recognition classifier classification accuracy.

Keywords: adaptation, semi-supervised learning, adaptation to the person, MCAT - multi classifier adaptive training, activity recognition

1. Introduction

Applications that classify highly variable data with machine learning are often faced with the problem of decreased classification accuracy when deployed in a real-world situation. For example, an activity-recognition classifier may show high classification accuracy when tested in a controlled environment, yet be significantly less accurate with an end-user. This issue could be resolved if enough person-specific data could be labelled in the real-world situation in which the classifier is deployed. However, since this is often impractical and labour-intensive, semi-supervised learning can be used to automatically label the data of specific end-user.

In semi-supervised learning, one or multiple classifiers usually label each instance. A mechanism selects the final class based on their outputs. If the confidence in this class is sufficient, it is used to label the instance, which is then added into the training set of the classifiers. This approach raises the following three challenges: (i) how to design the classifiers and what data to use for training each of them; (ii) how to choose the final class during the selection process; and (iii) how to decide if an instance will be added to the training set of the classifiers.

This paper introduces a novel algorithm for adapting an activity-recognition classifier to a person. This algorithm is referred to as Multi-Classifier Adaptive Training (MCAT). It addresses all of the above-mentioned challenges. The algorithm is based on the following key contributions: (i) it introduces two classifiers for classifying activities, a general one (trained on general data labelled in a controlled environment), and a specific one (trained on a limited number of labelled instances belonging to a specific person); (ii) the selection of the final class is handled by a meta-classifier, which uses the outputs of both the specific and general classifiers; and (iii) the decision about which instance to include in the training set of the general classifier is tackled by an additional meta-classifier. The two meta-classifiers combine the general activity-recognition model with the end-user model in the environment in which the classifiers are deployed. The final contribution of the paper is the training procedure, which takes maximum advantage of all the available data to properly train each of the four classifiers.

The MCAT algorithm was implemented and evaluated on an activity-recognition domain based on Ultra-Wideband (UWB) localisation technology. The individuals had four wearable sensors attached to their clothes (chest, waist and both ankles). The general activity-recognition domain knowledge was induced from the data of the individuals performing activities while wearing the sensors in the controlled laboratory environment. The specific data was obtained from an additional person to whom the system was adapted. The proposed approach was compared to three adaptive approaches: the initial version of the proposed approach [5], the self-learning algorithm [32], and the majority-vote algorithm [20,21].

The experimental results show that the MCAT algorithm successfully increases the classification accuracy of the initial activity-recognition classifier and significantly outperforms all three competing methods.

The rest of the paper is structured as follows. Section 2 reviews the related work on semi-supervised learning approaches and their use in adapting activity recognition. Section 3 introduces the motivating domain. Section 4 explains the MCAT algorithm. Section 5 describes the experimental setup and the results. Finally, section 6 concludes the paper.

2. Related Work

Semi-supervised learning is a technique in machine-learning that uses both labelled and unlabelled data. It is gaining popularity because the technology makes it increasingly easy to generate large datasets, whereas labelling still requires time-intensive human effort. The main idea of the semi-supervised approach is to extend classifiers either trained in supervised or unsupervised mode to label unlabelled data and include additional information into classifiers.

A similar approach is active learning, which also uses supervised learning for the initial classification. However, when the classifier is less confident in its output, a human annotator is consulted [6,27]. Human interaction is needed when high-risk data must be labelled, such as the data of patients with degenerative diseases. The focus of this paper is on low-risk data, where the cost of active learning is too high.

Four dimensions have been proposed along which semi-supervised learning can be categorised [32]: (i) single- and multiple-classifier; (ii) single- and multi-view; (iii) inductive and transductive; and (iv) classifier and database approaches. The single-classifier methods use only one classifier for the classification task, whereas multiple-classifier methods use two or more classifiers. The key characteristic of multi-view methods is multiple classifiers with different features and data for one classification problem. Single-view methods use classifiers with the same feature vector, but differentiate in the algorithm used for learning. Inductive methods first produce labels for unlabelled data and then train a classifier to use this self-labelled data. Transductive methods only produce labels and do not generate a new classifier. Classifier-based approaches start from one or more initial classifiers and enhance them iteratively. The database approaches discover an inherent geometry in the data and exploit it to find a good classifier. This paper will focus on multiple-classifier, single-view, inductive, classifier-based semi-supervised learning.

Self-training is the most common method that uses a single classifier [32]. After an unlabelled instance is classified, the classifier returns a confidence in its own prediction, or the class probability. If a class-probability threshold is reached, the instance is added to the classifier's training set, and the classifier is retrained. Bicocchi et al. [1] have successfully applied the self-training method to activity-recognition domains. Their activity-recognition classifier was initially trained on camera data and could recognise four atomic activities. The classifier was later used to label accelerometer data, which was intended to be used autonomously. Reported results are 80 percent classification accuracy for the self-trained accelerometer classifier, while the initial camera-based classifier achieves 88 percent classification accuracy. The self-training method can only be used if the initial classifier alone achieves a high classification accuracy, since misclassified instances used for retraining can decrease the classifier accuracy. The self-training has also been successfully applied to several other domains, such as handwriting recognition [10], natural language processing [13], and protein-coding gene recognition [9].

Democratic co-learning [31] is a single-view technique with multiple classifiers. All the classifiers have the same set of features that are trained on the same labelled data, but with different algorithms. When an unlabelled instance enters the system, all classifiers provide an output. The final label is based on a weighted majority vote among the classifiers. If the voting results have sufficient confidence, the instance is added in the training set of all the classifiers.

Co-training [2] is a multi-view method with two independent classifiers. To achieve independence, the features are split into two feature subspaces, one for each classifier. The classifier that surpasses a confidence threshold for a given instance can classify the instance. The instance is then added to the training set of the classifier that did not surpass the confidence threshold. A major problem of this algorithm is that the feature space of the data cannot always be divided orthogonally. If the features are randomly split, it is possible that classifiers do not satisfy the self-sufficiency requirement [8]. In other words, each classifier must achieve sufficient classification accuracy.

A modified co-training algorithm, En-Co-training [12], was used in the activity-recognition domain. The method uses information from 40 sensors. There are 20 sensors on each leg to identify the posture. The multi-view approach was changed into single-view by using all the data for training three classifiers with the same feature vector and a different learning algorithm. This is similar to democratic co-learning. The final label is chosen by majority voting among the three classifiers. The classified instance is added into the training set of all the classifiers. The reported results show an average classification improvement of 14.5 percentage points.

This paper extends our previous research on MCAT [4]. The MCAT method uses two classifiers. Both are trained with the same algorithm but on different data. A meta-classifier is used to make the final class prediction. The decision whether or not to put an instance into the training set is solved by using another meta classifier, rather than a threshold as seen in all the mentioned methods. In contrast to co-training and en-co-training, our two domain classifiers have the same feature set. Therefore, we do not have the problem of dividing the sets.

3. Motivating Domain

The MCAT algorithm is applied to activity recognition, which is a very common task in ambient intelligence. An activity-recognition classifier is usually trained using a machine-learning algorithm on the data retrieved from individuals performing predefined activities in a controlled environment, such as a research laboratory. The classifier trained in this fashion will typically report a high classification accuracy when tested in the same environment. However, it is likely that the classification accuracy will decrease when deployed in a new environment with an end-user, since each person tends to have specific characteristics and mannerisms in performing the activities. We faced this problem during the development and evaluation of the Confidence system [25,7] in different environments.

The Confidence system is developed for real-time activity monitoring and detection of abnormal-event (such as falls), or detection of long-term unusual behaviour that results from a developing disease. It is based on a six-layer architecture shown in Figure 1. The sensor layer serves raw localisation data to the next layer. The refining layer filters out the noise and interpolates the missing data. The reconstruction layer determines the location and reconstructs a person's activity. The interpretation layer interprets the state of the person and provides emergency information. The prevention layer observes the person and detects possible behavioural deviations [18]. The communication layer interacts with the user, relatives, or emergency services. Detailed description of the Confidence system can be found in [16,19]. The paper at hand focuses on the reconstruction layer.

System inputs from the sensor layer are the coordinates of Ubisense location tags [28] (based on UWB localisation technology) attached to the person's waist, chest and ankles. Since the Ubisense system is noisy, three filters are used, implemented in the refinement layer, to attenuate it. The data is first processed with a median filter. Afterwards the data is processed with an anatomic filter, which applies anatomic constraints. Finally, the data is processed with the Kalman filter. Detailed information on the filters were given in [17].

Tag positions in a specific timespan are represented as snapshots, created with 4 Hz frequency. Each snapshot is augmented with positions and various features for activity recognition and other purposes, such as detection of abnormal behaviour. For each tag the following features are computed: the z coordinate of the tag, the velocity of the tag, the velocity of the tag in the z direction. For each pair of tags the following features are computed: the distance in the z direction, the distance in the xy plane and the total distance. This results in 30 features per time point. To capture the pat-

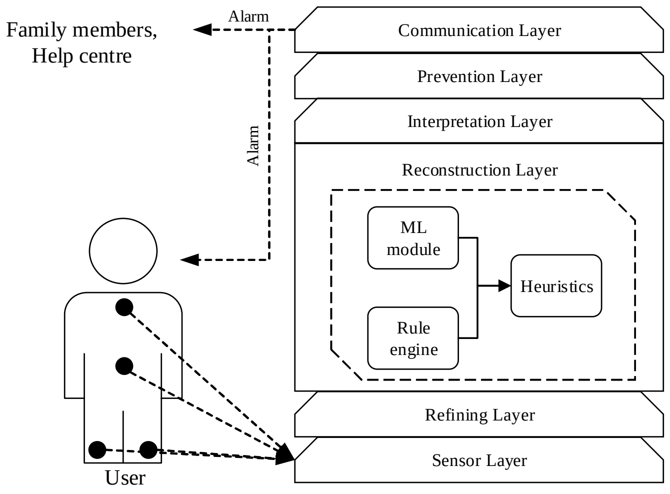

Fig. 1. The multi-layer system architecture of the Confidence system. It consists of six layers. The reconstruction layer is the focus of this paper, since it contains the activity-recognition mechanism.

tern of movement, the snapshot contains a sequence of features across 10 consecutive time points. Each feature vector thus contains 300 features, which is further in the paper referred as a snapshot. The reader is referred to [22] for more details about the features and the snapshot.

The reconstruction layer, which is responsible for activity recognition, can recognise eight atomic (elementary) activities: standing, sitting, lying, falling, on all fours, sitting on the ground, standing up, and sitting down. Activity recognition is done by combining two individual activity-recognition modules, as shown in Figure 1. The snapshots are processed by a machine-learning classifier and a rule engine working in parallel. Both return the activity that is fed into heuristic Bayesian rules, which provide the final snapshot label. Each module was evaluated in a controlled environment on data of five individuals, using the leave-one-person-out approach. The classification accuracy of the machine-learning (ML) module was 82 percent, while the accuracy of the rule-engine was 80 percent. When combined by heuristics, activity-recognition accuracy increases to 86 percent. However, if machine-learning module classification accuracy decreases, the overall accuracy will also decrease. The classification accuracy is the percentage of correctly classified instances in the dataset. The reader is referred to [23] for more details on the design and evaluation of the activity-recognition modules. The high activity-recognition classification accuracy is essential, since the entire reasoning of the system (detection of falls and abnormal behaviour) is based on it. Adapting activity recognition to each person helps improve the overall performance of the system. Accurate recognition of atomic activities also contribute to better recognition of complex activities [15]. The rest of the paper is focused on the ML module (Figure 1).

4. The Multi-Classifier Adaptive Training Method (MCAT)

This section describes the MCAT method that improves the classification accuracy of an initial activity-recognition classifier using unlabelled data and auxiliary classifiers. In addition to the general approach, a small amount of labelled data from the new real-world environment and new person is obtained. This is usually done when the system is introduced to the person for the first time.

The initial classifier is trained on activities performed by several people. When using this classifier on a new person, whose physical characteristics are different, the recognition classification accuracy can be low, since each person has a specific way of moving. The MCAT method uses a few activities performed by the new person to learn such specifics. The knowledge of the specifics is later used to adapt the initial classifier to further improve its performance.

4.1. The MCAT Algorithm

The proposed MCAT algorithm is shown in Figure 2 and presented as pseudo code in Algorithm 1. The

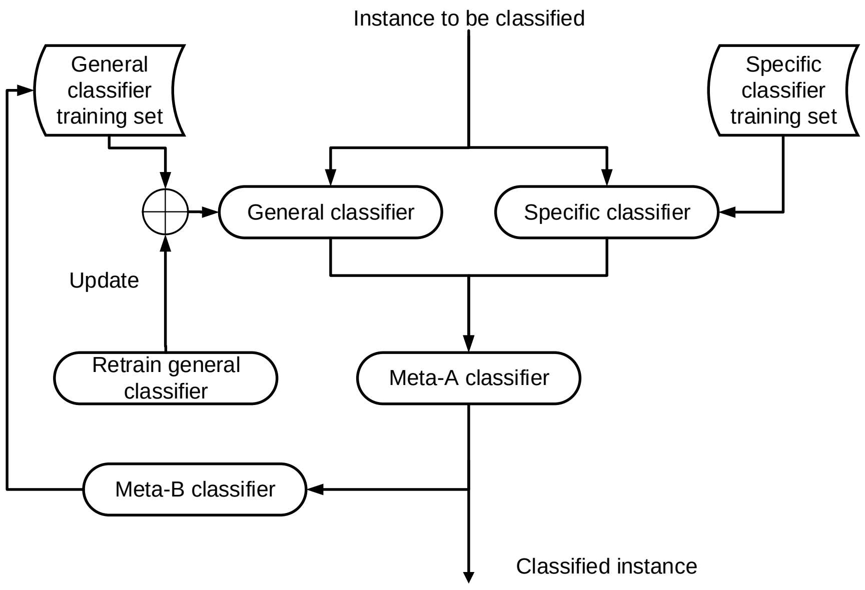
Instance to be classified
General classifier training set
Specific classifier training set
General classifier
Specific classifier
Update
Retrain general classifier
Meta-A classifier
Meta-B classifier
Classified instance

Fig. 2. The work flow of the algorithm proceeds as follows: The method contains two activity-recognition classifiers (general and specific) and two additional meta-classifiers. The meta-A classifier decides on the final class of the instance, and the meta-B classifier decides whether the instance is to be included in the training set of the general classifier.

general classifier is the initial classifier trained on the general domain data. This would be the only classifier deployed in a typical application that does not use the proposed method. In Figure 1, this would be the ML module inside the reconstruction layer. To improve the general classifier, a set of three auxiliary classifiers are proposed: (i) the *specific classifier*, which is trained on a small subset of manually labelled data specific for the person; (ii) the *meta-A classifier*, which decides which classifier (specific or general) will classify an instance; and (iii) the *meta-B classifier*, which decides whether the instance should be included in the training set of the general classifier.

The general classifier is trained on a general set of labelled data available for the activity-recognition domain in a controlled environment. The features are domain-specific. They are listed in Section 3 and denoted as AR (activity recognition) in Table 1. The machine-learning algorithm is chosen based on its performance on the domain.

The specific classifier is trained on a limited amount of labelled data that is specific to the new environment in which the classifiers are deployed (the new person). Note that this limited dataset does not necessary contain all the classes that are present in the dataset of the general classifier. This may be due to an unbalanced distribution of class labels. For example, in the activity-recognition domain, quick and short movements such as falls are rare. The classes the specific classifier knows are termed basic. The features and the machine-learning algorithm should preferably be the same as those used in constructing the general classifier as shown in the second column of Table 1.

After both classifiers return their outputs (Algorithm 1, lines 5, 6), the meta-A classifier is activated. The meta-A classifier's decision problem is to select one of the two classifiers to classify a new instance (Algorithm 1, lines 8–12). The meta-A classifier should be trained with the machine-learning algorithm performing best on the domain. The features for the meta-A classifier should describe the outputs of the general and specific classifiers as completely as possible, while remaining domain-independent. If domain features are added to the meta-features, the decision of the meta-A classifier will also be based on the specifics of the training data available prior to the deployment of the classifiers. This may be different from the specifics of the situation in which the classifiers are deployed. Our previous work [5] experimentally confirmed that domain features do not contribute to higher classifier accuracy of the classifier. Instead, they can result in overfitting to the specifics of the training data.

Although the features in meta-A classifiers, deployed in different systems need not be exactly the same, the following set of features (also shown in the third column of Table 1) are proposed:

Algorithm 1 Multi-Classifier Adaptive Training

```
1: Load classifiers $G$, $S$, $MA$, $MB$;                          ▷ Load all four classifiers
2: $timer \leftarrow n$                                            ▷ Adaptation time
3: $set \leftarrow \{\}$                                           ▷ Set of instances that will be added into the training set
4: procedure MCAT($G$, $S$, $MA$, $MB$, $inst$)
5:     $C_G \leftarrow G$.classify ($inst$)                        ▷ Classify with general classifier
6:     $C_S \leftarrow S$.classify ($inst$)                        ▷ Classify with specific classifier
7:     $ma_{attr} \leftarrow$ generate features for $MA$           ▷ Generate meta-A features
8:     if $MA$.classify($ma_{attr}$) is $G$ then                   ▷ Select which classifier will classify the instance
9:         $class \leftarrow C_G$
10:    else
11:        $class \leftarrow C_S$
12:    end if
13:    $mb_{attr} \leftarrow$ generate features for $MB$           ▷ Generate meta-B features
14:    if $MB$.classify($mb_{attr}$) is $TRUE$ then                ▷ Should the instance be included into the G dataset
15:        $set \leftarrow$ add($inst$)
16:    end if
17:    $timer \leftarrow timer - 1$
18:    if $time$ is 0 then                                         ▷ Start of adaptation when timer expires
19:        $G_{dataset} \leftarrow$ add($set$)
20:        $G$.train($G_{dataset}$)                                ▷ Adaptation of the general classifier
21:        $timer \leftarrow$ n                                    ▷ Set timer to the initial value
22:        $set \leftarrow \{\}$                                   ▷ Empty the set
23:    end if
24:    return $class$                                              ▷ Return class for the current instance
25: end procedure
```

- The class predicted by the general classifier (C_G)
- The class predicted by the specific classifier (C_S)
- The probability assigned by the general classifier to C_G ($P_G(C_G)$)
- The probability assigned by the specific classifier to C_S ($P_S(C_S)$)
- The probability assigned by the general classifier to C_S ($P_G(C_S)$)
- The probability assigned by the specific classifier to C_G ($P_S(C_G)$)
- Is C_G one of the basic classes? ($Basic(C_G)$)
- Are C_G and C_S equal? ($Equal(C_G, C_S)$)

After the selection process, the meta-B classifier then solves the problem of whether or not an instance should be included in the general classifier's training set (Algorithm 1, lines 14–16). The meta-B classifier's output should determine if the current instance contributes to a higher classification accuracy of the general classifier. This question is not trivial and there are several approaches that specifically address it [30]. We use a heuristic that answers the question: "Did the meta-A classifier select the correct class for the current instance?" The heuristic performs well and is computationally inexpensive, so the investigation of more complex approaches will be left for the future work. The features used in the meta-B classifier are the same as those in the meta-A classifier, with one addition: The

Table 1

Features per classifier. The first row represent features used for activity recognition (AR), rows from 2-10 are the features to be used for training the meta-A classifier. The feature in the last row is an additional feature for the meta-B classifier, the confidence of the meta-A classifier in its prediction.

	Classifiers			
Features	General	Specific	Meta-A	Meta-B
AR	x	x		
C_G			x	x
C_S			x	x
$P_G(C_G)$			x	x
$P_S(C_S)$			x	x
$P_G(C_S)$			x	x
$P_S(C_G)$			x	x
$Basic(C_G)$			x	x
$Equal(C_G, C_S)$			x	x
C_{meta-A}				x

confidence of the meta-A classifier in its prediction. The features of the meta-B classifier can be observed in the fourth column of Table 1. The meta-B classifier should be trained with the machine-learning algorithm performing the best on the domain.

The instances selected as eligible for inclusion into the dataset of the general classifier are stored in a set (Algorithm 1, line 15). The adaptation is performed when the predefined timer expires (Algorithm 1, lines 18–23). The stopping criteria of the algorithm can either be running out of unlabelled instances in offline adaptation or (in the case of online adaptation) a duration of adaptation in days, hours, weeks or even never.

4.2. *Training Procedure*

The training of the four classifiers requires the data to be divided into subsets so that no classifier is used to classify the data on which it was trained, while maximally utilising all the data. In particular, the meta-A classifier needs data classified by the general and specific classifiers, neither of which should be trained on these data. Furthermore, the meta-B classifier needs data classified by the meta-A classifier, but the meta-A classifier should not be trained on this data.

We propose a training procedure that divides the data into subsets for training both the general classifier and the meta-classifiers. The general classifier is simply trained on the complete dataset, while meta-classifier training is explained in the following paragraphs. Steps are labelled a.1 through a.7 for the meta-A classifier, and b.1 through b.10 for the meta-B classifier. The procedure is general, but for the purposes of this paper the activity-recognition domain is used. Specifically, the publicly available Localisation Data For Person Activity dataset from the UCI Machine Learning Repository [29] will be used. The dataset consists of recordings of five people performing a predefined scenario five times. The scenario is a continuous sequence of activities that represent typical daily activities. Each person had four Ubisense [28] tags attached to the body (neck, waist, and both ankles). Each tag returns its current position. The goal is to assign one of 8 activities to each time frame. The dataset can be divided five times by the person, and five times by the scenario repetition, or episode (steps a.1 and b.1).

Meta-A classifier training. The procedure is shown in Figure 3 and Algorithm 2.

Four people are selected for training a temporary general classifier (step a.2), which is used to classify the fifth person (step a.3). This is repeated five times, once for each person. The resulting complete dataset is classified with the temporary general classifier (step a.4). The data classified with the temporary general classifier is represented with dots in Figure 3. This data is then split five times by the episode. Since the specific classifier should be trained on a small amount of data, one episode for each person is used for training a temporary specific classifier (step a.5). The remaining four episodes are classified with the temporary specific classifier (step a.6). The data classified with the temporary specific classifier are represented with stars in Figure 3. There is a total of five temporary specific classifiers (one per person), each of which classified four episodes. This finally gives us the data in the step a.7, which represents the training set for the final meta-A classifier.

Meta-B classifier training. Figure 4 and Algorithm 3 show this procedure. The steps from b.1 to b.4 are identical to the steps from a.1 to a.4 in the meta-A classifier training procedure. Steps that are the same for meta-A and meta-B training have shaded background in Figure 4. The data classified with the temporary general classifier (dotted data in step b.4) is then divided by the episode. Four episodes are used to train a temporary meta-A classifier (step b.5), while the remaining episode is kept aside to be classified by this classifier (step b.6). The training then proceeds in essentially the same way as the training of the final meta-A classifier (steps a.5 through a.7). One episode for each person is used to train a temporary specific classifier (step b.7), and the remaining three episodes are classified with the temporary specific classifier (step b.8). This gives five specific classifiers (one per person), each of which was used to classify three episodes, yielding the data in step b.9. This data is used to train the temporary meta-A classifier. The data that was kept aside in step b.6 is now retrieved and classified with the temporary meta-A classifier (b.10). Steps b.5 through b.9 are repeated five times to classify all five episodes, yielding the training set for the final meta-B classifier (step b.10).

4.3. *Computational complexity*

The computational complexity of instance labelling introduces only a constant increase in computational load, thus only computational complexity for training procedure and retraining of the classifier is analysed. Since MCAT approach may use various algorithms as the core classifiers, only relative changes in computational complexity are analysed.

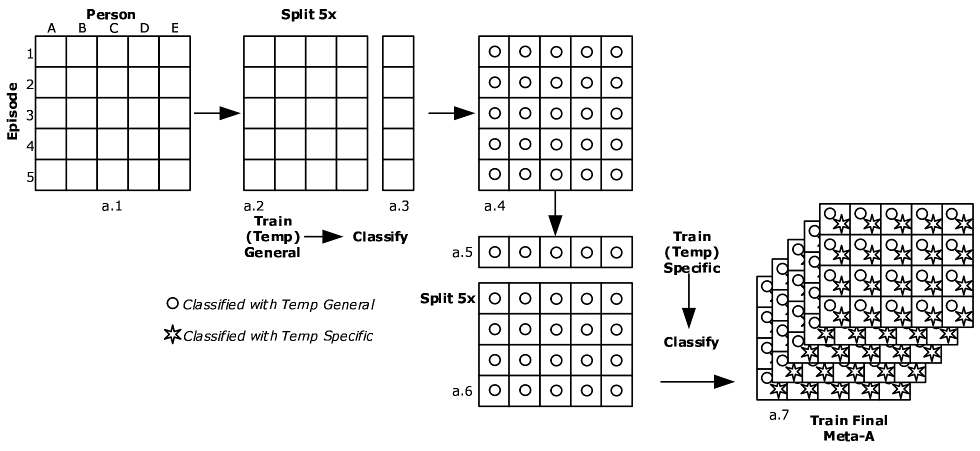

Fig. 3. Preparing the data for meta-A classifier training.

Algorithm 2 Prepare data for meta-A

```
 1: Load dataset $data$;
 2: $numberOfpersons$;
 3: $dataDot$                                         ▷ Data classified with general classifier
 4: $dataMetaA$                                       ▷ Data prepared for meta-A training
 5: procedure PREPAREDATAFORMETAA($data$)
 6:     for $i$=0 to $numberOfpersons$ do
 7:         $G \leftarrow$ train($data[not\ i]$)                       ▷ Figure 3 Step a.2
 8:         $dataDot[i] \leftarrow G$.classify($data[i]$)              ▷ Figure 3 Step a.3
 9:     end for
10:     for $i$=0 to $numberOfpersons$ do
11:         $S \leftarrow$ train($data[i][0]$)                         ▷ Figure 3 Step a.5
12:         $dataMetaA[i] \leftarrow S$.classify($dataDot[i][not\ 0]$)  ▷ Figure 3 Step a.6
13:     end for
14:     return $dataMetaA$            ▷ Result is data for training meta-A, Figure 3 Step a.7
15: end procedure
```

Let p be the number of people and let e be the number of episodes per person. Suppose a base classifier with the training time complexity O_b is chosen. The training time complexity using $p \cdot e$ data is defined by Eq. (1).

$$O_b(p \cdot e) \tag{1}$$

The meta-A classifier has three levels of training. First, it builds p temporary general classifiers using $(p-1) \cdot e$ data. Second, it builds $p \cdot e$ temporary specific classifiers using less than 1 episode unit data. And third, the final meta-A classifier is built using $p \cdot (e-1) \cdot e$ data. The total complexity of training the meta-A classifier is defined by Eq. (2).

The meta-B classifier has four levels of training. Similarly, it first builds p temporary general classifiers using $(p-1) \cdot e$ data. Second, $p \cdot e$ temporary specific classifiers are built using less than 1 episode unit data. Third, e temporary meta-A classifiers are built using $p \cdot (e-2) \cdot (e-1)$ data. Finally, the meta-B classifier is built on $p \cdot e$ data. The computational complexity is defined by Eq. (3). The upper limit for total training computational complexity is given by Eq. (4).

$$\begin{aligned} O_{metaA} &\leq p \cdot O_b((p-1) \cdot e) + p \cdot e \cdot O_b(1) \\ &\quad + O_b(p \cdot (e-1) \cdot e) \\ &\leq p \cdot O_b(p \cdot e) + p \cdot e \cdot O_b(1) \\ &\quad + O_b(p \cdot e^2) \end{aligned} \tag{2}$$

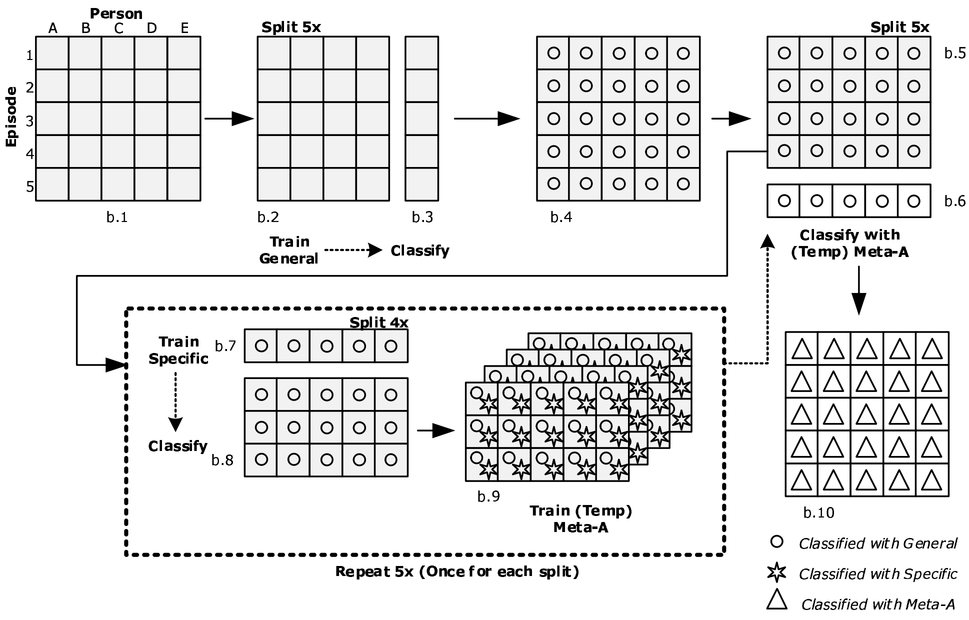

Fig. 4. Preparing the data for meta-B classifier training.

Algorithm 3 Prepare data for meta-B

```
 1: Load dataset $data$;
 2: $numberOfpersons$;
 3: $numberOfEpisodes$;
 4: $dataTemp$                                              ▷ Temporary data
 5: $dataDot$                               ▷ Data classified with general classifier
 6: $tempDataMetaA$                         ▷ Data prepared for temporary meta-A training
 7: $dataMetaB$                                   ▷ Data prepared for meta-B training
 8: procedure PREPAREDATAFORMETAB($data$)
 9:     for $i$=0 to $numberOfpersons$ do
10:         $G \leftarrow$ train($data[not\ i]$)                         ▷ Figure 4 Step b.2
11:         $dataDot[i] \leftarrow G$.classify($data[i]$)                  ▷ Figure 4 Step b.3
12:     end for                                   ▷ Result is $dataDot$ Figure 4 Step b.4
13:     for $j$=0 to $numberOfEpisodes$ do
14:         $dataTemp \leftarrow dataDot[not\ j]$                        ▷ Figure 4 Step b.5
15:         for $i$=0 to $numberOfpersons$ do
16:             $S \leftarrow$ train($dataTemp[i][0]$)                  ▷ Figure 4 Step b.7
17:             $tempDataMetaA[i] \leftarrow S$.classify($dataTemp[i][not\ 0]$)   ▷ Figure 4 Step b.8
18:         end for
19:         $tempMetaA \leftarrow$ train($dataTempMetaA$)                 ▷ Figure 4 Step b.9
20:         $dataMetaB[j] \leftarrow tempMetaA$.classify($dataDot[j]$)      ▷ Figure 4 Step b.6
21:     end for
22:     return $dataMetaB$              ▷ Result is data for training meta-B, Figure 4 Step b.10
23: end procedure
```

The MCAT training procedure introduces polynomial increase (of power two) in the number of trained classifiers as well as in the amount of trained data. Since instance labelling introduces only a constant increase, that is, an instance is labelled with four classifiers instead of one, the main source of complexity increase remains in the classifier training procedure. Since the training procedure is run only when the MCAT is initialised, the complexity increase does not affect real-time labelling. Occasional retraining of the general classifier, which is triggered during labelling, requires $O_b(p \cdot e + \epsilon)$ steps, where ϵ is the number of newly arrived instances.

$$\begin{aligned} O_{metaB} &\leq p \cdot O_b((p-1) \cdot e) \\ &\quad + e \cdot p \cdot (e-1) \cdot O_b(1) \\ &\quad + e \cdot O_b(p \cdot (e-2) \cdot (e-1)) \\ &\quad + O_b(p \cdot e) \\ &\leq (p+1) \cdot O_b(p \cdot e) \\ &\quad + p \cdot e \cdot (e-1) \cdot O_b(1) \\ &\quad + e \cdot O_b(p \cdot e^2) \end{aligned} \tag{3}$$

$$\begin{aligned} O_{upper} &\leq [p \cdot O_b(p \cdot e) + p \cdot e \cdot O_b(1) + O_b(p \cdot e^2)] \\ &\quad + [(p+1) \cdot O_b(p \cdot e) \\ &\quad + p \cdot e \cdot (e-1) \cdot O_b(1) + e \cdot O_b(p \cdot e^2)] \\ &\quad + [O_b(p \cdot e)] \\ &\leq (2 \cdot p + 2) \cdot O_b(p \cdot e) + p \cdot e^2 \cdot O_b(1) \\ &\quad + (e+1) \cdot O_b(p \cdot e^2) \end{aligned} \tag{4}$$

5. Experimental Evaluation

The experimental evaluation focuses on the activity recognition discussed in the previous sections. The main reason why the general classifier will likely not perform well in a real-world environment on a particular person is that each person has their own specifics, such as the height and movement characteristics. The general classifier, which is trained on several people, can then recognise the activities of a "general person". Obtaining enough training data for a particular end-user is difficult, so the MCAT method is well-suited for solving this problem.

5.1. Experimental Setup

The experimental data was collected using the Confidence system described in Section 3. Two different sets of data were collected, one for the classifier training (*the training dataset*) and one for testing the semi-supervised adaptation to the individuals (*the test dataset*).

The training dataset was retrieved from the UCI Machine Learning Repository [29]. The data was divided into segments as discussed in Section 4.2 and used for training the classifiers.

The test dataset consists of the recordings of 10 people performing typical daily activities. All individuals were different than in the training dataset. Each person repeated the scenario five times, which provided an average 2.8 hours of data per person, and 28.5 hours altogether. The scenario was designed to reflect the distribution of the activities during one day of an average person. The scenario contains eight activities: standing, lying, sitting, sitting down, standing up, falling, on all fours, and sitting on the ground. Compared to the training dataset, the scenario in the test dataset contains an additional episode of sitting on the ground, with the respective transitions.

The training dataset contains more labels than the test dataset, so the transition activities were merged as follows. The activities lying down and sitting down were merged into the going down activity. The standing up from lying, standing up from sitting, and standing up from sitting on the ground activities were merged into the standing up activity. The walking activity was merged into the standing activity.

The test dataset was divided as follows in order to create datasets needed for the proposed method: (i) the basic activity dataset had 10 people performing the basic activities (30 seconds per activity), which were lying, standing, and sitting; it was used to train specific classifier for each person and (ii) the unlabelled test dataset had 10 people performing the scenario five times (125 minutes per person on average, data for the basic activity dataset is excluded); it was used for the semi-supervised adaptation, which is the core of MCAT.

Demographic information on the people used for MCAT adaptation is shown in Table 2. Additional information about the datasets can be seen in Tables 3 and 4, which show the number of instances, duration and distribution of classes per person of the unlabelled test dataset and the basic activity dataset, respectively, in comparison to the training dataset.

The classifier training used the training dataset, as described in Section 4.2. The meta-A classifier was trained with the Support Vector Machines algorithm [24] and the meta-B classifier used the C4.5 ma-

Table 2
Information about the people used in the test dataset

Person	A	B	C	D	E	F	G	H	I	J
Gender	M	M	F	M	M	M	M	F	F	F
Age	27	26	28	28	24	27	25	34	29	2 7
Height (cm)	194	181	160	184	188	178	187	160	172	168

Table 3
Number of instance, duration in hours and distribution of classes per person of the unlabelled test dataset in comparison to training dataset used to train the general classifier.

	Training	Person									
	dataset	A	B	C	D	E	F	G	H	I	J
# Instances (x10000)	4.1	2.5	2.6	3.0	3.1	3.2	3.4	3.2	3.2	3.5	3.2
Duration (hours)	3.80	2.28	2.36	2.76	2.84	2.93	3.18	2.97	2.96	3.23	2.96
Standing (%)	19.56	30.27	31.56	29.40	31.87	28.84	29.02	24.98	29.27	28.40	31.03
Sitting (%)	16.67	15.82	16.10	16.11	15.82	16.06	14.69	16.19	14.96	14.67	14.96
Lying (%)	33.14	31.57	31.18	30.52	30.62	31.22	30.25	32.40	33.43	33.38	31.43
Sitting on the ground (%)	7.20	15.26	15.69	15.46	15.36	16.65	14.69	19.00	16.18	17.31	15.59
On all fours (%)	3.06	0.49	0.54	0.71	0.57	0.67	0.72	0.66	0.69	0.44	0.65
Falling (%)	1.82	1.23	0.92	0.88	1.17	0.64	0.84	0.87	0.62	0.62	0.79
Going down (%)	4.80	1.42	0.96	1.08	1.25	1.46	1.75	1.37	1.12	1.01	1.20
Standing up (%)	13.75	3.94	3.05	5.85	3.34	4.46	8.04	4.52	3.74	4.17	4.34

Table 4
Number of instance, duration in minutes and distribution of classes per person of the basic dataset used to train the specific classifier in comparison to training dataset used to train the general classifier. The classes that are not present in the basic dataset are omitted from the table.

	Training	Person									
	dataset	A	B	C	D	E	F	G	H	I	J
# Instances	40968	304	303	303	307	305	304	302	305	306	307
Duration (minutes)	228	1.69	1.68	1.68	1.71	1.69	1.69	1.68	1.69	1.70	1.71
Standing (%)	19.56	33.88	33.66	33.66	33.22	33.44	34.21	34.44	32.79	34.31	33.22
Sitting (%)	16.67	32.57	32.34	33.33	33.55	34.10	32.24	32.78	33.44	32.35	33.55
Lying (%)	33.14	33.55	33.99	33.00	33.22	32.46	33.55	32.78	33.77	33.33	33.22

chine learning algorithm [26]. The classifiers could be trained with other machine-learning algorithms, but these two algorithms proved satisfying in experiments. Both were used with the default parameter values as implemented in the Weka suite [14], and the features presented in Section 4.2 and in Table 1. The general classifier was trained with the Random Forest algorithm [3] and the features presented in Section 3 and in Table 1.This algorithm was selected according to the results from previous research [11]. The classifier is 82 percent accurate (as seen in Section 3) when evaluated using the leave-one-person out approach. This implies that one can expect the same classification accuracy when the classifier is deployed in the new environment. However, this is not the case, as Table 5 shows in the column labelled with G (initial general classifier). There, the average classification accuracy is 70.03 percent. The specifics of each person when performing the activities explain this decreased classification accuracy.

The specific classifier was trained with the Random Forest algorithm on the basic activity dataset for each person in the test dataset. The unlabelled test dataset of that person was processed with the MCAT method. The general classifier was retrained after each episode of the test scenario (five episodes per person), to take advantage of the instances from the unlabelled dataset that were included in its training dataset.

The MCAT method was compared to five competing methods: two transductive approaches that only labelled the instance and did not perform any adaptation; and three inductive semi-supervised learning approaches. The transductive approaches are: (i) the baseline approach; and (ii) the MCAT without meta-B approach. The inductive approaches are: (i) self-training; (ii) majority vote; and (iii) threshold-based MCAT.

The baseline approach merges the training data of the general and the specific classifier into one training set and then trains a new general classifier.

MCAT without meta-B is MCAT without the general classifier adaptation. It builds both the general and specific classifier and uses the meta-A classifier to decide which one to trust.

Self-training is a well-known method for semi-supervised learning [10]. It uses the general classifier for classification. The instances in the unlabelled test dataset with 100 percent classification confidence are added to its training set. A 100 percent confidence is most commonly used as a self-training threshold in related work. Self-training was not specifically adapted to be used with our dataset.

The majority vote is slightly modified Democratic co-learning [31]. The modification is done on the level of decision, since the original paper does not contain enough information for effective reimplementation. The majority vote uses three classifiers trained on the same training set with different machine-learning algorithms. The algorithms that achieved the highest classification accuracy in the evaluation of general classifier with the leave-one-person out approach were used. These were: Support Vector Machines, Random Forest, and C4.5. The instances in the unlabelled test dataset with 100 percent classification confidence were included in the training set of all classifiers as seen in Democratic co-learning.

The threshold-based MCAT is the previous version of the current MCAT algorithm [5]. It uses a threshold rule of 100 percent instead of the meta-B classifier to select the instances to be included in the training set of the general classifier. It was included in the comparison to show the benefits of the meta-B classifier.

The experiment was done in two steps: The classifier training and then the MCAT as a semi-supervised adaptation of the initial general classifier.

The data for each person was processed two times by all the methods. In the first run, the instances selected for inclusion in the training set were assigned a weight of 2 to accelerate the adaptation and in the second run the weight was decreased to 1 to avoid overfitting. If an instance already existed in the training set and was selected for the inclusion again, it was discarded.

5.2. Results

Table 5 shows the absolute difference in classification accuracies between the initial general classifier and the general classifier after adaptation. The left side of Table 5 shows the classification accuracy of the initial general classifier G and the specific classifier S. The right side of Table 5 shows the gain/loss in classification accuracy of the MCAT and the competing methods, compared to the initial general classifier.

The results for the general classifier show a decrease in classification accuracy when used in a different environment than the controlled environment, which had a classification accuracy of 82 percent (cross-validated on the training set). The average classification accuracy on 10 people was 70.03 percent (Table 5, column Initial G). The results of the specific classifier (Table 5, column Specific S) show that it achieved a slightly higher classification accuracy than the initial general classifier in several individual cases, even though it was able to predict only three basic classes (lying, standing, sitting).

The worst results were with the baseline method, where the merged datasets (general and specific) were used for training. The classification accuracy for four people decreased compared to the general classifier. This is because some instances from the specific and general classifier were similar but differed in the label. The results of using the meta-A classifier to select the final class showed that this method outperformed the general classifier by 6.14 percentage points (pp). The higher classification accuracy is due to the knowledge of the basic activities representing 76.67 percent of the dataset. This increase in the classification accuracy reveals that using a classifier for class selection is one reason for the success of the proposed approach. Another reason is the semi-supervised adaptation.

The results of the self-training show that in a few cases, where the general classifier has a low initial classification accuracy, the method performs poorly and further weakens the classifier. The average classification accuracy after adaptation was 71.91 percent. The results of the majority-vote method show that introducing extra classifiers contributes to a gain in classification accuracy compared to the initial classifier.

Table 5

Classifier classification accuracies and comparison of the MCAT method to the transductive approaches: baseline (G and S merged G&S) and MCAT without adaptation (only meta-A); and semi-supervised approaches: self-training (ST) and majority vote (MV); and our: threshold-based MCAT (TB) and full MCAT.

	Initial classifier (classification accuracy %)		Method Comparison (gain/loss in pp against G)					
Person	Initial G	Specific S	G&S	Meta-A	ST	MV	TB	MCAT
A	75.28	76.82	-10.57	+3.62	+2.36	+0.38	+3.82	+9.18
B	76.28	60.06	+1.17	+0.58	+0.28	+0.64	+0.70	+7.38
C	62.87	70.85	+8.52	+12.82	-1.18	+3.07	+13.46	+20.58
D	69.55	76.17	-0.21	+8.99	+2.23	+2.10	+9.77	+12.57
E	68.13	74.23	+0.20	+5.78	-1.81	+6.73	+9.76	+16.22
F	73.57	68.18	-4.42	+2.62	+6.07	+3.67	+8.08	+8.86
G	65.42	67.72	5.97	+6.99	-0.21	+8.57	+9.76	+12.60
H	73.45	67.46	-8.02	+0.01	+4.31	+0.41	+4.51	+10.53
I	62.09	68.08	+7.75	+16.81	-0.18	+11.03	+16.98	+17.08
J	73.67	74.17	+1.18	+3.15	+6.87	+5.98	+8.73	+11.65
Average	70.03	70.37	+0.16	+6.14	+1.87	+4.26	+8.56	+12.66

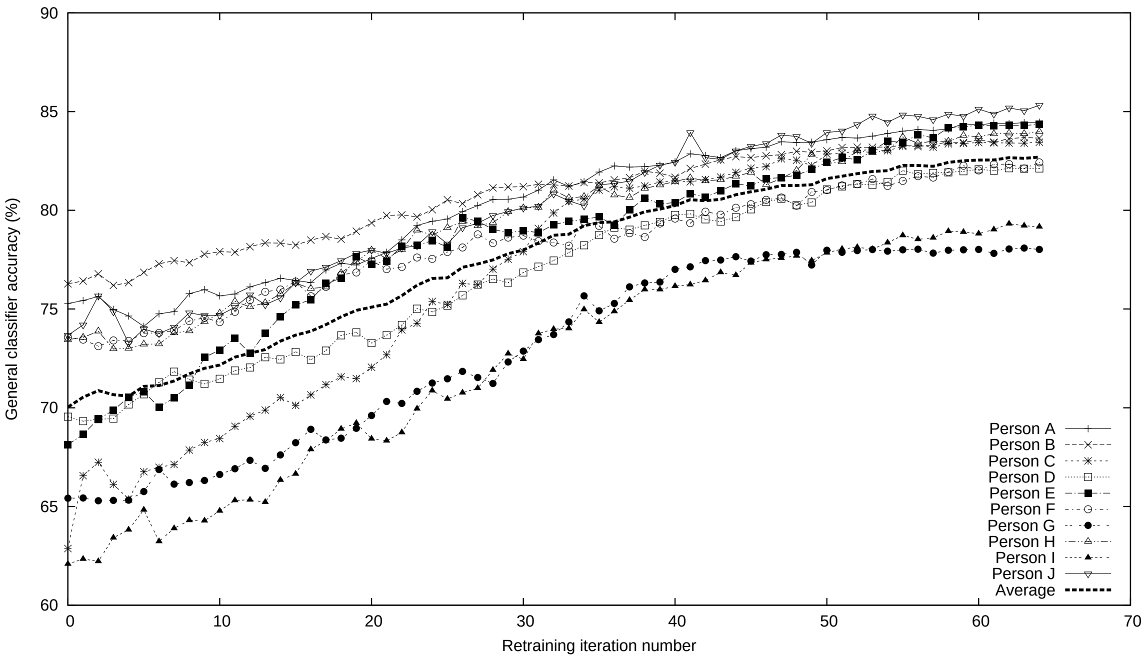

Fig. 5. Adaptation process. Increase in classification accuracy per person and on average.

The average classification accuracy after the adaptation with the majority-vote method was 74.29 percent. The average classification accuracy of the general classifier after adaptation using the previous version of MCAT was 78.59 percent. The average gain in the classification accuracy was 8.56 percentage points.

The results for the MCAT method show the average gain in the classification accuracy of 12.66 percentage points, and the average classification accuracy of the classifier was 82.70 percent. In the best case (Person C), it achieves an increase in the classification accuracy of 20.58 percentage points. The worst case (Person B) had an increase in the classification accuracy of only 7.38 percentage points. Person J achieved the highest classification accuracy of 85.32 percent. The MCAT method outperformed the self-training method (by 10.79 percentage points), the majority vote (by 8.41 percentage points), and the previous version of MCAT (by 4.11 percentage points).

Figure 5 shows the trend of changing the classification accuracy with the adaptation time. The dashed lines represent the general-classifier classification accuracy for each person, and the solid line is the average general-classifier classification accuracy line for the entire process. We observed that the average line increased rather monotonously. The reported final classification accuracy was measured after all the data was used. We speculate that if more data was fed to the MCAT method, we could gain more classification accuracy before achieving convergence.

To further evaluate the results of the MCAT method, a paired two-tailed t-test for statistical significance of the algorithm was performed. The t-test was performed for initial classifiers (general and specific) and for each algorithm the MCAT method was compared to. The results of the test returned a probability of $6 * 10^{-4} < p < 4.8 * 10^{-6}$. The conclusion is that MCAT statistically outperformed all other methods.

6. Conclusion and Discussion

The paper focuses on the problem of improving an initial activity-recognition classifier using unlabelled data. The two main contributions of this paper are: (i) a novel approach for specialising the activity-recognition classifier by semi-supervised learning referred to as MCAT; and (ii) a procedure for training multiple classifiers from a limited amount of labelled data that fully utilises the data. The MCAT method for the semi-supervised learning uses auxiliary classifiers to improve the classification accuracy of the initial general classifier. It uses a specific classifier trained on a small amount of labelled data specific to a particular person, in addition to the general-knowledge classifier. The two additional meta-classifiers decide whether to trust the general or the specific classifier on a given instance, and whether or not the instance should be added to the training set of the general classifier.

The MCAT method was compared to two transductive approaches and three inductive semi-supervised approaches. The MCAT method significantly outperformed the baseline approach (by an average 12.51 percentage points) and the MCAT without meta-B (by an average 6.53 percentage points). The MCAT method also significantly outperformed inductive approaches. It outperformed the self-training (by an average 10.79 percentage points), the majority vote (by an average 8.41 percentage points), and the threshold-based MCAT (by an average 4.11 percentage points). On average, the classification accuracy of the initial classifier improved by 12.66 percentage points. The classification accuracy trend analysis suggest that if there was more data to feed to the MCAT algorithm, a higher classification accuracy could be achieved before convergence.

The outcomes of the experiment were tested for statistical significance with a paired two-tailed t-test. The results show that MCAT statistically significantly ($p << 0.01$) outperformed all other methods and the initial classifiers.

To verify if MCAT can significantly contribute to further development of the ambient intelligence applications, many more tests will have to be performed. It is not clear yet if the method is successful only for recognising an individual's activities or if it can be used for general learning tasks. Real-time systems produce large amount of unlabelled data that can be used for adapting the modules to a specific environment or person. Each person has specific mannerisms in performing activities. This can be used to achieve higher classification accuracy on activity recognition. MCAT offers several potential benefits, including accurate recognition of atomic activities, which can also contribute to more reliable recognition of the complex activities.

In addition to more concrete analysis of the algorithms performance, overfitting of the classifiers and drift, the algorithm will have to be evaluated on several other domains. Moreover, the future work includes further investigation of the MCAT algorithm to work on regression domains, for example, estimation of human energy expenditure.

Acknowledgments.

The research leading to these results was partly supported by the ARRS under the Research Programme P2–0209, partly by the European Community's Framework Programme FP7/2007–2013 under grant agreement N 214986.

References

[1] N. Bicocchi, N. Mamei, A. Prati, R. Cucchiara, F. Zambonelli, *Pervasive Self-Learning with Multi-modal Distributed Sensors*, in: 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, IEEE Press, Washington, 2008, pp. 61–66.

[2] A. Blum, T. Mitchell, *Combining labeled and unlabeled data with co-training*, in: 11th annual conference on Computational learning theory, Morgan Kaufmann press, 1998, pp. 92–100.

[3] L. Breiman, Random Forests, *Machine Learning* **45** (2001), 5–32.

[4] B. Cvetković, B. Kaluža, M. Luštrek, M. Gams, *Multi-Classifier Adaptive Training: Specialising an Activity Recognition Classifier Using Semi-supervised Learning*, in: AmI 2012, F. Paterno, B. Ruyter, P. Markopoulos, C. Santoro, E. Loenen, K. Luyten eds., LNCS, vol. 7683, Spriner, Heidelberg, 2012, pp. 193–207.

[5] B. Cvetković, B. Kaluža, M. Luštrek, M. Gams, *Semi-supervised Learning for Adaptation of Human Activity Recognition Classifier to the User*, in: IJCAI Workshop on Space, Time and Ambient Intelligence, Barcelona, 2011, pp. 24–29.

[6] S. Dasgupta, Two faces of active learning, *Theoretical Computer Science* **412** (2011), 1767–1781.

[7] E. Dovgan, M. Luštrek, B. Pogorelc, A. Gradišek, H. Burger, M. Gams, Intelligent elderly-care prototype for fall and disease detection from sensor data, *Zdravniski Vestnik - Slovenian Medical Journal* **80** (2011), 824–831.

[8] J. Du, C.X. Ling, Z. H. Zhou, When Does Cotraining Work in Real Data?, *IEEE Transactions on Knowledge and Data Engineering* **23** (2010), 788–799.

[9] G. Feng-Biao, Z. Chun-Ting, ZCURVEV: a new self-training system for recognizing protein-coding genes in viral and phage genomes, *BMC Bioinformatics* **7**, (2006).

[10] V. Frinken, H. Bunke, *Self-training Strategies for Handwriting Word Recognition*, in: Advances in Data Mining. Applications and Theoretical Aspects, Perner P. ed., Springer, Heidelberg, LNCS vol. 5633, 2009, pp. 291–300.

[11] H. Gjoreski, A*daptive human activity/posture recognition and fall detection using wearable sensors*, Master thesis, Jožef Stefan International Postgraduate School, 2011.

[12] D. Guan, W. Yuan, Y.K. Lee, A. Gavrilov, S. Lee, *Activity Recognition Based on Semi-supervised Learning*, in: IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, IEEE press, Washington, 2007, pp. 469–475.

[13] R. Guzmán-Cabrera, M. Montes-Y-Gómez, P. Rosso, L. Villaseñor-Pineda, *A Web-Based Self-training Approach for Authorship Attribution*, in: 6th international conference on Advances in Natural Language Processing, Springer-Verlag press, 2008, pp. 160–168,

[14] M. Hall, E. Frank, G. Holems, B. Pfahringer, P. Reutemann, I. H. Witten, The WEKA Data Mining Software: An Update, *SIGKDD Explorations* **11** 2009, 10–18.

[15] L. Kalra, X. Zhao, A. J. Soto, E. Milios. Detection of daily living activities using a two-stage Markov model, *Journal of Ambient Intelligence and Smart Environments*, **3(5)** (2013), 273–285.

[16] B. Kaluža, B. Cvetković, E. Dovgan, H. Gjoreski, M. Gams, and M. Luštrek. Multiagent Care System to Support Independent Living, *International Journal on Artificial Intelligence Tools*, Accepted for publication (2013).

[17] B. Kaluža, E. Dovgan, *Glajenje trajektorij gibanja cloveškega telesa zajetih z radijsko tehnologijo*, in: Proceedings of the 12th International Multiconference Information Society - IS, Ljubljana, Slovenia, 2009, pp. 97–100.

[18] B. Kaluža and M. Gams. Analysis of Daily-Living Dynamics, *Journal of Ambient Intelligence and Smart Environments*, **4(5)** (2012), 403–413.

[19] B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, M. Gams, *An Agent-Based Approach to Care in Independent Living*, in: AmI 2010, B. Wichert, R. Keyson, D. Markopoulos, P. Streitz, N. Divitini, M. Georgantas, N. M. Gomez, A. De Reuter eds., LNCS, vol. 6439, Spriner, Heidelberg, 2010, pp. 177–186.

[20] L. I. Kuncheva, C. J. Whitaker, R. P. W. Duin, Limits on the majority vote accuracy in classifier fusion, *Pattern Analysis and Applications* **8** (2003), 22–31.

[21] B. Longstaff, S. Reddy, D. Estrin, *Improving activity classification for health applications on mobile devices using active and semi-supervised learning*, in: 4th International Conference on Pervasive Computing Technologies for Healthcare, IEEE press 2010, pp. 1–7.

[22] M. Luštrek, B. Kaluža, Fall detection and activity recognition with machine learning, *Informatica* **33** (2009), 197–204.

[23] V. Mirchevska, M. Luštrek, i. Velez, N. G. Vega, *Classifying Posture Based on Location of Radio Tags*, in: AMIF, P. Cech, V. Bures, L. Nerudova eds., Ambient Intelligence and Smart Environments, vol. 5, IOS Press, 2009, pp. 85–92.

[24] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*, MIT Press, Cambridge, MA, 1999.

[25] Project Confidence FP7, `http://www.confidence-eu.org/`, 2013.

[26] Q. J. Ross, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco,1992.

[27] B. Settles, *Active Learning Literature Survey*, CS Technical Report, University of Wisconsin-Madison, 2009.

[28] Ubisense, `http://www.ubisense.net`, 2013.

[29] UCI ML Repository, `http://archive.ics.uci.edu/ml/`, 2013.

[30] C. Yanhua, R. Manjeet, D. Ming, H. Jing, Non-negative matrix factorization for semi-supervised data clustering, *Knowledge and Information Systems* **17** (2009), 355-379.

[31] Y. Zhou, S. Goldman, *Democratic Co-Learning*, in: 16th IEEE International Conference on Tools with Artificial Intelligence, IEEE press, Washington, 2004, pp. 594–602.

[32] X. Zhu, *Semi-Supervised Learning Literature Survey*, CS Technical Report, University of Wisconsin-Madison, 2005.