ActiServ: Activity Recognition Service for Mobile Phones

Martin Berchtold
TU Braunschweig
berch@teco.edu

Matthias Budde, Dawud Gordon, Hedda R. Schmidtke, Michael Beigl
Karlsruhe Institute of Technology (KIT)
[lastname]@teco.edu

Abstract

Smart phones have become a powerful platform for wearable context recognition. We present a service-based recognition architecture which creates an evolving classification system using feedback from the user community. The approach utilizes classifiers based on fuzzy inference systems which use live annotation to personalize the classifier instance on the device. Our recognition system is designed for everyday use: it allows flexible placement of the device (no assumed or fixed position), requires only minimal personalization effort from the user (1–3 minutes per activity) and is capable of detecting a high number of activities. The components of the service are shown in an evaluation scenario, in which recognition rates up to 97% can be achieved for ten activity classes.

1. Introduction

Although smart phone devices are powerful tools, they are still passive communication enablers rather than active assistance devices from the user's point of view. The next step is to introduce *intelligence* into these platforms to allow them to proactively assist users in their everyday activities. One method of accomplishing this is by integrating situational awareness and context recognition into these devices. Smart phones represent an attractive platform for activity recognition, providing built-in sensors and powerful processing units. They are capable of detecting complex everyday activities of the user (i.e. standing, waking, biking) or the device (i.e. calling), and they are able to exchange information with other devices and systems using a large variety of data communication channels.

Several approaches to smart phone based recognition (e.g. [13, 6, 8, 12]) have been published which demonstrate the importance of research in this field. The architecture presented here builds on this research, but has several advantages over previous approaches, thus enabling context awareness under realistic conditions. Orientation Robustness: one problem in mobile context recognition is that the way in which the device is carried greatly affects the ability of conventional classifiers to recognize activities. The classifier structure presented here is robust to this through a novel training structure. Class Diversity: The ActiServ approach provides a platform for classification of many different activities (here 10 classes). Usability: The ActiServ system does not require the user to have prior knowledge or abilities in order to operate the system, and user mistakes will not worsen performance. Online Personalized Optimization: The system uses an annotation program on the mobile device to gather new training data which is used to improve the recognition algorithm *at run-time*. Crowd-Sourcing: Not only does this feedback optimize local recognition, it also is used to improve performance over the whole community of users. This also means that feedback from remote users in the community improves recognition for local algorithms, which represents a *novel break* with conventional recognition algorithms.

To use the ActiServ system, the user downloads a base recognition app for the mobile phone that already provides decent recognition of everyday activities and can be used immediately. The user can then improve recognition rates by performing a few personal training steps, providing feedback to the system. The system does not apply further restrictions, meaning a steep learning curve should be avoided and normal usage of the device should not be affected.

Related Work As early as 1999, the project *TEA* [13] proposed methods for recognizing context with low-level sensors, demonstrating the general feasibility on an extended Nokia mobile phone. The context-aware mobile phone *SenSay* [14] is designed to, among other features, automatically turning the ringer on and off and prevent inaudible ringer volume in a loud environment. However it requires an external sensor box on the user's hip as well as ambient microphones on their body. Wearable sensors are also employed in the *iLearn* system [12] for Apple's *iPhone* – in that case the *Nike+iPod Sport Kit*.

Activity recognition based on fuzzy classifiers is presented in [9] and [16]. Both systems use external sensors with a fixed body position and do not implement the recognition on an embedded device. In [10] an activity recogni-

tion module is applied to a health care monitoring system.

[1] is an often cited publication on activity recognition in pervasive computing using acceleration sensors located on four limb positions and the hip. In [11], the *eWatch* sensing platform is used to detect six activities. The *eWatch* is carried in the test subject's pocket among other body positions. [6] and [8] both do activity recognition with mobile phones, where [6] only uses sensors and resources native to the phone for sensory acquisition and classification, but just six activities are distinguished with only 80% recognition accuracy on average. In [8] a wristband in combination with a mobile phone is used to recognize six activity classes.

While today many activity recognition systems exist, few of them combine the following desirable features: The system should be able to achieve high overall recognition rates for a reasonable amount of activity classes. For that task it should only employ the internal sensors of an unmodified commodity cellphone on which the recognition is done internally. Furthermore the system should have been evaluated using a fairly high number of test subjects. Of the described systems, only one system [1] has been designed to recognize more classes than the system presented in the paper at hand. It also has been evaluated using the same high number of subjects. However, the recognition rates of said system are significantly lower than those of ours, in spite of the fact that it uses five external acceleration sensors that need to be placed in fixed positions. In terms of high recognition rates, ActiServ ties with *iLearn* [12], whereas ActiServ goes without the use of external fixed sensors. Of the systems that were surveyed, only ours and two others [11, 6] exclusively employ non-fixed sensors. Again, the recognition rates lie well below the ones our system achieves. One of these systems [11] also does the classification on the device, as well as ActiServ and three others [8, 12, 7]. However, the latter three employ fixed sensors for the activity recognition. ActiServ is the only system combining the desired features described above. In addition, ActiServ has the unique features of being service-oriented and personalizable to the individual as well as the community as a whole.

2. The Architecture

The problem of personalized activity recognition was solved with a service based approach. The service consists of several steps of activity classifier training, optimal classifier selection, classifier personalization, and user accuracy feedback. Each step is realized by different components either located on a server or the user's phone. All steps and components of ActiServ are displayed in Fig. 1 and described below. Subsequently, the workflow of the service is presented.

Activity Classification Module Set (ACMS): The ACMS is the collection of classification modules which are running

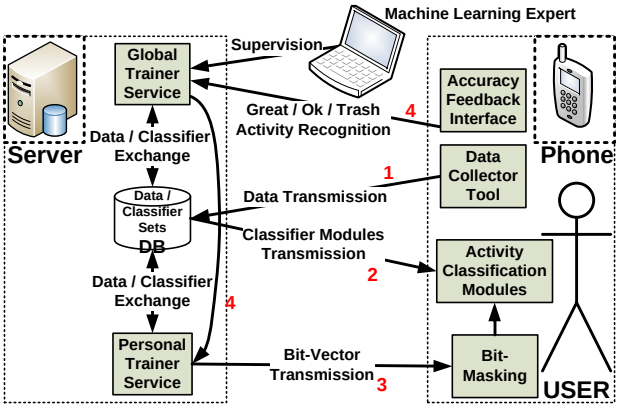

Figure 1: ActiServ service architecture.

on the user's mobile phone at a given time.

Global Trainer Service (GTS): The GTS is the key component which trains new Activity Classification Module Sets (ACMS). The GTS frequently checks the database for new user activity data sets or for combinations of user data which have not been used. When data is found, the GTS creates a new ACMS using the new data combination. Through the GTS the database is constantly filled with new ACMSs, where badly performing ACMSs are replaced to ensure that the database always consists of the best performing ACMSs.

Personal Trainer Service (PTS): The PTS selects which activity classification modules are delivered to the user device. The decision is made based on the performance of the classifiers over the annotated data collected by the user. The PTS is also responsible for personalizing the classifier modules. This process will be explained in depth further on in this paper.

Data/Classifier Set Database: This unit is the central storage for the global ACMSs and user data.

Bit-Masking: The bit-masking provides a method for personalizing an ACMS without destroying its original activity recognition capabilities.

Data Collector Tool (DCT): The DCT on the mobile phone collects annotated activity data. The user selects the activity they want to perform from an extensible set of activities in a drop-down-list, carries out the activity, and then pushes a button to stop recording.

Accuracy Feedback Interface (AFI): With the AFI, the user can give feedback to the GTS/PTS as to whether their activity recognition is working. This feedback is not used for personalization per se, as the personalization process is done using the bit-mask, but rather could be used to change the training process in general, though this is not currently implemented and outside of the scope of this paper.

ActiServ Workflow - How the Service Works First, the user downloads the ActiServ components to the phone. Second, a small amount of initial data must be collected, where

each of the activities available in the drop down list of the DCT is carried out for 1-2 minutes. When all of the activities are complete, the data is transmitted (step 1 Fig.1) to the ActiServ server. At this point, the interaction between the user and the service is finished and will only be re-initiated over the AFI if necessary. On the server the PTS selects the best performing ACMS from the database based on the initial data and transmits the set to the user device (step 2 Fig.1). This step has nearly no delay, since only a search over the pre-existing ACMSs must be done and no training is performed. The transmitted ACMS can now run on the users phone and recognize activities with an initial accuracy. Meanwhile the PTS is training the personalization of the ACMS currently running. This process takes time (depending on efficiency and complexity about 1 hour), but since an ACMS is already running on the user's phone, the delay is not directly recognizable for the user. The personalization data, which is just a bit-vector and not a complete ACMS, is transmitted to the phone in step 3 (Fig.1). On the phone the bit-masking component personalizes the ACMS, which can be done during runtime in between classifications. Now the phone should have reasonable activity recognition rates (see evaluation), but the process runs further on the ActiServ server. The GTS constantly trains new combinations of ACMSs, in which the new user's data is included as well. Over time, an ACMS is present in the database which has been trained on the data from the new user and can be transmitted to the user's phone (recognition rates can rise to above 97%). This new ACMS can be personalized again via the PTS and therefore be further improved. The user can also give feedback to the system via the AFI (step 4 Fig.1), but this is not necessary. If the ActiServ is in a faulty state due to bad user data, an expert can intervene.

3. Activity Classification Module Set

For activity recognition we use a novel Activity Classification Module Set (ACMS), through which we are able to classify a large number of classes with reduced calculation effort. The ACMS system can provide accuracy of over 97%. In this section, we first explain a simplified monolithic activity classification approach and then extend the architecture to the ACMS.

3.1. Recurrent Activity Classification

The classification consists of several steps of processing a real world value to a tuple of the class recognized and a fuzzy uncertainty value. The system (Fig. 2) is presented in the following, a detailed description can be found in [3].
1. Feature Extraction: The features for activity recognition with the used 3-axis accelerometers are variance and

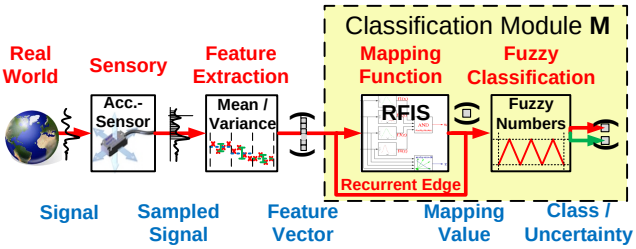

Figure 2: Classification system architecture.

mean values (two 3D acc. sensors $\rightarrow$ 12-dim. feature vector). 2. Recurrent FIS Mapping: We use a Takagi, Sugeno and Kang [15] (TSK-) with linear functional consequences as mapping function. The output of the TSK-FIS is assigned to a tuple of class and fuzziness. The outcome of the mapping at time t is fed back as input dimension n for the TSK-FIS mapping at $t+1$. Instead of 'Recurrent TSK-FIS' we use the simpler term RFIS in the remainder of this paper. For more detailed information on this process please refer to [2]. 3. Fuzzy Classification: The assignment of the RFIS mapping result to a class is done fuzzily, so the result is not only a class identifier, but also a membership, representing the reliability of the classification process. Each class c_k is interpreted by a triangularly shaped fuzzy number. The highest degree of membership to one of these numbers determines which identifier is the mapping outcome. The overall output of the classifier module $\mathbf{M}$ (which encapsulates the RFIS and the fuzzy classification) on the feature vector $\overrightarrow{v}_t$ is a tuple (c_k, μ_{c_k}) of a class identifier and the membership to it, where $c_k \in \mathcal{C}$ and $\mu_{c_k} \in [0,1]$ (cf. [3]). 4. Fuzzy Uncertainty Filter: The classifications vary strongly with respect to fuzziness and therefore in the reliability of the RFIS mapping. Since more classifications are made than needed for most applications, a filter on the fuzzy uncertainty ($\mu_{c_k} \leq \tau$) can improve reliability, but also reduces the number of classifications.

3.2. Activity Classification Module Set

Instead of using one monolithic classifier to classify on all classes $\mathcal{C}$, we use several classifier modules $\mathbf{M}_i : \mathcal{V} \rightarrow \mathcal{C}_i$ (with $i = 1, .., N$) each classifying on a small subset $\mathcal{C}_i \subseteq \mathcal{C}$ of classes. The subsets $\mathcal{C}_i$ are chosen according to the classes $c_{ij} \in \mathcal{C}_i$ semantics, therefore each subset $\mathcal{C}_i$ has its own meta semantic. We call this meta semantic 'conditional context (cc)'. To not only recognize the respective classes $c_{ij} \in \mathcal{C}_i$, but also the transition between classifiers $\mathbf{M}_i$, each module yields a complementary class $\overline{c}_i$ as well, where the complementary class represents all classes classified by other modules but not by this one. All classifier modules are chained in a dynamic queue, where the last classifier successfully classifying a class aside from the complementary class is moved to the front.

To train these queued classifier modules, we need to train

them on the respective classes $c_{ij} \in \mathcal{C}_i$ and on the complementary class $\overline{c}_i$. The training $\mathcal{V}_i^{tr}$ and check data sets $\mathcal{V}_i^{ck}$ for a classifier module $\mathbf{M}_i$ are unified with a selection of input data pairs of all other classifiers $\mathcal{V}_i^{\overline{c}} \subset \bigcup_{k \neq i} \mathcal{V}_k^{tr}$. This selection is labeled zero – which indicates the complementary class $\overline{c}_i$ in every module $\mathbf{M}_i$ – and added to the normal training and check data sets of this classifier. The actual training and check data is therefore $\mathcal{V}_i^{tr \cup \overline{c}}$ and $\mathcal{V}_i^{ck \cup \overline{c}}$, which are called $\mathcal{V}_i^{tr}$ and $\mathcal{V}_i^{ck}$ in the rest of this paper for reasons of simplicity. More analysis on the modular classifier approach can be found in [4].

4. Global Trainer Service (GTS)

A classification system which is trained on a large data set of multiple users has disadvantages. First, the training algorithms have long calculation times, since the training data set increases with every user added to it. Second, the resulting Activity Classification Module Set (ACMS) is inaccurate and complex. Since the diversity of the training data due to the different users is high, the classifier modules need a large number of rules in the RFIS mapping function to map the data. The data is partly contradictory, because different users have different patterns for certain activities. In this case, either one user's training data is preferred to solve the conflict, or both are classified with low accuracy. Instead of training the ACMS on the training data of all users in the database, we select subsets of user-specific data and train the classifier modules on them. Instead of one set of modules, we end up with various sets trained on subsets of the users. All the ACMSs are stored in a database, where only the best and fittest ACMSs remain and the worse performing are deleted. With each new user added to the database new data combinations of users become possible, so the GTS is constantly running and training new ACMSs.

4.1. Activity Classifier Module Training

To train a set of activity classifier modules, we used a machine learning algorithm [3] that is fast and requires a minimal amount of supervision from an expert. The part of each activity classification module that needs to be trained is the RFIS mapping function. A five step algorithm is used to identify the RFIS on an annotated training feature set. The algorithm is described in [3], a brief description follows:

1. Data Annotation and Separation: The training data $\mathcal{V}^{tr}$ is separated according to the class c_j to which the data pairs belong. Clustering on each subset delivers rules that can be assigned to each class. 2. Clustering: First, subtractive clustering gives an upper bound for the amount of clusters that is then used for Gath-Geva clustering. Since subtractive clustering results in more clusters than Gath-Geva clustering (multivariant cluster shapes for covariant sensor data), a genetic algorithm is used to determine the best subset of cluster centers. The output of the Gath-Geva clustering is the number of rules and the membership functions. 3. Least Squares: Linear regression identifies the parameters of the linear consequence function of the rules. Minimizing the quadratic error leads to the solution of an overdetermined linear equation. 4. Recurrent Data Set: The output of the TSK-FIS is now calculated over the training data $\mathcal{V}^{tr}$. This output is shifted by one, with a leading zero, and then added to the training data set $\mathcal{V}^{tr}$ as an additional dimension. All data pairs for time $t > 1$ have the output of the FIS mapping of $t - 1$ in the recurrent dimension n. For this data set the steps 1 to 3 are repeated. 5. Stop Criterion: There are two values qualifying for a stop criterion: the mean quadratic error and the classification accuracy. For our evaluation scenario we used the latter.

4.2. ACMS Training

The training $\mathcal{V}^{tr}$ and check data $\mathcal{V}^{ck}$ for one classifier modules set $\mathcal{M}_l = \{\mathbf{M}_{1l}, .., \mathbf{M}_{Nl}\}$ consists of data from randomly selected users ($u_h \in \{u_1, .., u_A\}$), represented by $\mathcal{V}_{u_h}$. The training data $\mathcal{V}_{\mathcal{M}_l}^{tr}$ for the classifier module set $\mathcal{M}_l$ is a subset selection of the data $\mathcal{V}_{\mathcal{M}_l}^{tr} \subseteq \mathcal{V}_{u_g} \cup .. \cup \mathcal{V}_{u_h}$ for $g \neq .. \neq h$, where each of the classes should have equal amount of training data pairs and the data from every user should contribute same amount of data sets. For the check set $\mathcal{V}_{\mathcal{M}_l}^{ck}$ we proceed accordingly. All user data is saved in a database, from which different combinations of data are randomly selected. on each selection a new set of classifier modules set $\mathcal{M}_l$ is trained. The set $\mathcal{M}_l$ is then checked with the respective check data $\mathcal{V}_{\mathcal{M}_l}^{ck}$ for classification accuracy. This process is repeated until all combinations of user specific data are trained and checked, where only a certain population of ACMSs which have the highest classification accuracy is saved. With new user data coming in through the community, the process can theoretically run indefinitely. The amount of stored classifier module sets should increase with the amount of new user data added to the database. Since storage space is limited, the amount can not be increased indefinitely.

5. Personal Trainer Service (PTS)

The classifier module set $\mathcal{M}_l$ which performs best for the current user is selected and must then be personalized to recognize activities with acceptable accuracy. This is done via a bit-vector masking, which 'activates' or 'deactivates' the input dimensions of the rules of each module's RFIS mapping function. This masking is only temporary, so the original classifier modules still persist and can be reactivated at any time. Furthermore, the bit-vector and therefore the capabilities of the masked classifier can be changed at

any time. The bit-vector is specified for the respective user according to the data $\mathcal{V}_{u_h}$ on which the module set $\mathcal{M}_l$ was selected. Since the combinations of bits in the bit-vector exclude a full search, a genetic algorithms is used as a heuristic to limit the search space.

5.1. ACMS Selection

The first step towards a user's personalized activity classification system is the selection of the best classifier module set from the database. The selection is done based on an initial data set which the user has previously collected. The data set should consist of a significant amount of data pairs for all activity classes. For every classifier module set $\mathcal{M}_l$ ($l = 1, .., A$) the classification accuracy for the new user data is calculated. The classifier module set $\mathcal{M}_l$ which has the best classification accuracy is then selected for the activity recognition on the user's device.

5.2. Bit-Vector for ACMS Masking

The adaption is done via a bit vector, which specifies the 'active' and 'inactive' dimensions of each rule for one module $\mathbf{M}_i$. Therefore the bit vector $bit_{\mathbf{M}_i}$ for module $\mathbf{M}_i$, which has n input dimensions and m_i rules, is $b_i := n \cdot m_i$ positions long. To use the bit vector, an interpretation function $I(\mathbf{M}_i(\overrightarrow{v}_t), bit_{\mathbf{M}_i})$ is defined, that 'switches' the rule's dimensions temporarily without changing the module $\mathbf{M}_i$ permanently. The interpretation function I is defined as a function mapping a module $\mathbf{M} \in \mathbb{F}(\mathbb{R}^n, \mathbb{R})$ together with a bit vector $bit_{\mathbf{M}} \in \{0, 1\}^*$ of appropriate length to a module $\mathbf{M}'$. Details on the bit vector approach can be found in [5]. A genetic algorithm is used to determine the respective classifier module's bit vector. The space that has to be searched is 2^{b_i}, a complete search would therefore have a runtime of $O(2^{b_i})$, which is impossible to calculate in a reasonable amount of time. In our experience, the genetic algorithm can find a suboptimal, but appropriate solution in a time span that is acceptable in our application.

6. Online ACMS Evaluation

For the online ACMS evaluation we used the *OpenMoko Neo Freerunner* phone based on the Samsung S3C2442B processor clocked at 266MHz using a Debian Linux operating system enabling rapid prototyping. We are also currently investigating implementations on the iPhone 3G and the Motorola Milestone running the Android operating system. The ACMS is implemented in C and the parameter setting is provided through a JASON configuration file.

Performance Evaluation To gather meaningful performance data, we must first compute whether the requirements for real-time processing can be achieved: the accelerometer sensor provides 100 samples per second and each classification requires window size of 8 samples in the feature extraction. Thus, if we want to achieve real-time performance, we must be able to perform 12.5 activity classifications per second. For the classification processes, a distinction must be made between the best and worst case: in the best case, the first classifier module of the ACMS is able to classify the data, and in the worst case, all classifiers modules need to be computed. Each process was run 1000 times to gather significant performance results. The results indicate that we only require from 1.3% (best case) to 4.9% (worst case) of the available processing power. Of that, 0.012 *PP* are used for the feature extraction and 0.093 *PP* (best case) to 0.38 *PP* (worst case) for the classification. The Neo FreeRunner's processor does not feature floating point instructions, meaning that all floating point calculations have to be simulated using the integer instruction set. With a phone providing a floating point unit (e.g. iPhone 3G) we would need approximately ten times less processing time.

Also, the processor time for the ACMS was observed for 131 minutes in a trial period in order to evaluate CPU utilization. The measurements showed a CPU usage of 3.3% for the ACMS on average.

7. Offline Activity Classification Evaluation

The ActiServ system was evaluated offline. First, the evaluation settings will be explained, followed by a demonstration of upper and lower bound approximations for the optimal case where the training data for the classifier modules are only gathered from the evaluation user. In the next step, an ACMS is selected from the database where data from the current user is not present, meaning there is no ACMS available which was directly trained on data from this user. The accuracy of the best performing ACMS under these circumstances is presented. Next, personalization using the PTS to provide bit-vectors for each module is presented where data from the current user is excluded from the database. Finally the performance of the whole system including data from the evaluation user as well as the community is presented, where different phone orientations and personalization are applied.

7.1. Evaluation Setting

As mentioned before, there is a distinction between context classes and conditional contexts (cc). The classes are the direct output of the classifier, whereas the cc is implicitly identified through the classifier module which is currently active. The classes are sorted according to semantics of the cc. Three general cc were determined, *the phone is on a table*, *the phone is in the pants pocket* and *the user has the phone in her hand*. The cc *the phone is in the pants pocket*

is split in subsequent contexts according to the amount of movement of the phone, *movement* or *no movement*. The combinations of contextual states, classes and classifiers for the acceleration data classification are shown in Tab. 1.

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users pants pocket: *no movement*	user is sitting user is standing user is lying	1 2 3	$\mathbf{M}_1$
movement	user is walking user is climbing stairs user is cycling	4 5 6	$\mathbf{M}_2$
Phone on table:	no movement	7	$\mathbf{M}_3$
Phone in users hand:	just holding talking on phone typing text message	8 9 10	$\mathbf{M}_4$

Table 1: Conditional contexts, classes and classifier modules for the acc. sensor.

We collected data from 20 users (16 male, 4 female), aged 20 to 32. All users had experience with the use of mobile phones. We tried to ensure that the collection of activities was as realistic as possible. Each test user generated 2-3 minutes of data for each of the ten activity classes during normal everyday activities, resulting in over 500 minutes of data in total.

The classifier module sets are trained on randomly selected sets of four users. 19 of the 20 users were used during the training phase, leaving the data from one user for evaluation of the ActiServ system. The male test subject used for the evaluation provided data with two different orientations for the 'pocket' activities. All the test data used in the following evaluation is randomized in slices of 16 data pairs. This is a stress test in the evaluation, since due to the recurrence in the modules, the most false-positives occur in between activity classes.

In the following tables the recognized activity classes are filtered on a threshold of $\tau = 75$. This reduces the amount of classes which are recognized by a certain amount, depending on the general detectability of the respective activity. Which percentage of the classification remain after the filtering is shown in the last row of the confusion matrices. We expect that a remainder of more than 8% of classifications to be acceptable, so on average still one classification passes through the reliability filter per second.

7.2. ACMS Trained on Evaluation User

The best results for activity recognition can be achieved when the ACMS is directly trained on the current user. This gives us an upper limit for activity recognition accuracy for a 3-minute training cycle. Data collection resulted in about 2250 feature vectors per class and 22500 in total for all ten classes. Out of this annotated data set, 600 samples per class were extracted for training data $\mathcal{V}^{tr}$ and 400 for check data $\mathcal{V}^{ck}$. For training the complementary class of each of the modules, 1200 data pairs were randomly selected from training data of the other classifier modules and added to the training data. For the check of the complementary class, 800 data pairs were added to the check data. The remaining data – ~ 1250 annotated feature vectors per class – was used to test the ACMS. The upper limit results for the trained ACMS are presented in Tab. 2.

	$\mathbf{M}_1$ (98.1%)			$\mathbf{M}_2$ (96.4%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (99.3%)		
	1	2	3	4	5	6	7	8	9	10
1	97.8	0.8	0.1	4.2	0.2	0.7	0.4	0.0	0.3	0.0
2	0.6	96.3	1.2	0.9	0.0	0.0	0.0	0.0	0.1	0.0
3	0.0	0.0	97.4	0.0	0.1	0.0	0.0	0.0	0.0	0.0
4	0.8	1.3	1.2	92.3	0.1	0.2	0.0	0.9	0.6	0.0
5	0.0	0.0	0.0	0.3	97.7	1.6	0.0	0.0	0.0	0.0
6	0.0	0.0	0.1	0.0	0.0	97.1	0.0	0.0	0.0	0.0
7	0.5	0.0	0.0	0.0	0.0	0.0	99.5	0.0	0.0	0.0
8	0.3	1.0	0.0	2.2	1.5	0.3	0.1	97.6	1.1	0.0
9	0.1	0.5	0.0	0.2	0.4	0.1	0.0	1.4	97.8	0.8
10	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	99.1
	83.9	83.2	93.5	69.4	57.5	74.5	91.7	86.3	83.2	93.5

Table 2: Upper limit classifier system, solely trained on eval. user. Threshold τ=0.75, conditional context (cc): 98.3%, overall: 97.3%.

A lower baseline limit can be expressed when the classifier was trained on one phone orientation (e.g. for the phone in the pants pocket) and the evaluation is carried out on data from the user when carrying it in a different position, which represents a major problem in activity recognition. The recognition results for the test data with a different phone orientation then the training data are shown in Tab. 3. Here the classes for the cc 'phone in user's pants pocket' have extremely low recognition rates, where the activities with only one possible orientation still have high accuracy. We

	$\mathbf{M}_1$ (64.6%)			$\mathbf{M}_2$ (51.4%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (99.7%)		
	1	2	3	4	5	6	7	8	9	10
1	20.7	44.7	0.1	30.1	10.2	20.0	0.1	0.0	0.1	0.0
2	0.2	29.3	0.7	12.4	2.5	21.8	0.0	0.0	0.1	0.0
3	0.0	0.0	98.2	1.0	0.2	0.0	0.0	0.0	0.0	0.0
4	0.0	20.9	0.5	55.2	84.1	13.5	0.0	0.2	0.5	0.0
5	0.0	0.6	0.2	0.6	0.6	0.3	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.3	0.3	99.8	0.0	0.0	0.0
8	79.2	4.1	0.3	0.7	2.0	40.6	0.1	98.5	0.7	0.1
9	0.0	0.4	0.0	0.0	0.1	3.5	0.0	1.3	98.5	0.6
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	99.3
	39.6	22.8	92.1	31.4	34.8	11.9	94.6	89.8	89.8	96.2

Table 3: Lower limit classifier system for position 2, solely trained on actual user's position 1. τ=0.75, cc: 78.9%, overall: 60.0%.

show in the evaluation of our service-based approach, that the recognition accuracy achieved is not significantly lower than the upper limit and can exceed accuracy for the lower baseline limit.

7.3. Eval. User Excluded from Training

As explained before, the upper limit classifier modules cannot be supported in a community-based approach

due to several reasons. We show that we can exceed the lower baseline limit for a different phone orientation with our service-based approach and present example results for each of our service's steps.

GTS Trained ACMS for Random Selection of User Data The Global Trainer Service (GTS) has trained 20 Activity Classification Module Sets for randomly selected sets of four users from the 19 users in the database. The training

User Combination	Accuracy (%) $\mathcal{V}^{tr}$	$\mathcal{V}^{ck}$	User Combination	Accuracy (%) $\mathcal{V}^{tr}$	$\mathcal{V}^{ck}$
17, 2, 1, 11	86.0	83.1	18, 4, 7, 15	85.6	81.4
12, 11, 1, 7	83.9	83.2	4, 7, 2, 15	85.06	83.7
6, 14, 15, 16	NaN	NaN	12, 17, 7, 16	NaN	NaN
16, 13, 12, 7	NaN	NaN	9, 13, 18, 17	82.9	82.5
3, 6, 8, 4	85.2	82.7	13, 6, 2, 16	82.7	80.2
15, 16, 12, 2	66.3	66.1	3, 17, 16 14	69.5	64.5
13, 8, 6, 1	84.5	80.2	13, 1, 16, 2	83.9	80.7
10, 7, 12, 2	84.7	81.2	10, 15, 4, 5	84.7	82.8
5, 1, 10, 1	72.5	71.4	7, 5, 17, 12	82.7	80.8
9, 5, 18, 8	82.7	69.0	5, 13, 19, 8	NaN	NaN

Table 4: GTS combinations of user data, accuracy of ACMS (training/check data). Gray rows indicate faulty ACMSs that are deleted.

data $\mathcal{V}^{tr}$ consisted of 300 randomly selected feature vectors per class and user, which makes 1200 pairs per class. Again, training data from the other modules was added to train the respective classifier module on the complementary class. In total we used 12000 data pairs for training of all four classifier modules. The check data consisted of 6000 data pairs. The accuracy for the ACMSs, trained on training data $\mathcal{V}^{tr}$ and check data $\mathcal{V}^{ck}$, is shown in Tab. 4. For some of the ACMS an error occurred during training and the accuracy on the training and check data is 'Not a Number (NaN)'. These ACMSs are deleted by the GTS from the database along with badly performing ACMSs.

Best Selection of Classifiers A new user now demands an ACMS to recognize the 10 activity classes in our evaluation setting. The PTS selects the best performing ACMS on the user data and uploads the ACMS to the user phone. The best selection is the ACMS trained on the users 10, 7, 12, and 2. A mean recognition rate of 62.4% ($\tau = 0.75$) is low, but the more user invariant activity classes were recognized with over 90% accuracy. Also the cc have an accuracy in detection of 71.2%, which is already a good recognition rate, for an ACMS that was provided without delay.

Personalization Data and Test Data Have Same Orientation The PTS now provides a bit-vector for masking the selected ACMS, and therefore personalizes the activity recognition. The resulting confusion matrix is displayed in Tab. 5. The recognition rates have improved significantly by 23.6 *PP* up to 86%. Still, classes no. 4 and no. 6 have low recognition rates. These classes are mostly misclassified onto class no. 1 of a different classifier module. Here the personalization process of the PTS can be improved, so that

	$\mathbf{M}_1$ (90.2%)			$\mathbf{M}_2$ (72.8%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (97.0%)		
	1	2	3	4	5	6	7	8	9	10
1	85.4	2.7	1.0	31.5	1.7	29.8	0.2	0.2	5.9	0.9
2	1.4	84.9	0.4	1.5	0.4	0.9	0.0	0.0	0.8	0.2
3	0.0	0.7	94.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.3	9.1	2.4	54.9	0.4	0.0	0.1	0.0	0.8	0.0
5	0.1	1.0	1.3	0.0	92.5	5.5	0.1	0.0	0.0	0.1
6	11.6	0.0	0.0	0.0	3.9	61.3	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	99.3	0.0	0.0	0.0
8	0.0	1.6	0.1	10.5	1.0	1.7	0.2	98.4	1.2	0.0
9	0.0	0.0	0.4	1.5	0.0	0.9	0.1	1.4	90.9	0.9
10	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	97.9
	19.9	24.9	42.5	23.4	18.4	11.6	41.9	22.6	25.9	40.8

Table 5: Best performing selection personalized on new user.τ=0.75,cc:89.8%,all:86.0%.

the classifier module $\mathbf{M}_1$ is not personalized to the point it interrupts the capabilities of module $\mathbf{M}_2$. But this is a trade-off, where the stop criterion of the genetic algorithm used in the PTS plays the key role. Either modules have equal average recognition rates or one is favored in the disadvantage of another one. This can also happen between classes classified through the same module.

Personalized Opposite Orientation in Training and Test Data The PTS has personalized the ACMS via a bit-vector masking according to a data set, where the phone has only one orientation in the user's pants pocket. We are investigating now what happens if the user is carrying the phone with a different orientation, which is a situation that frequently occurs in daily usage of mobile phones. The results of a test set with about 22000 data pairs for a different phone orientation are shown in Tab. 6. The accuracy is on average low at

	$\mathbf{M}_1$ (79.3%)			$\mathbf{M}_2$ (83.6%)			$\mathbf{M}_3$	$\mathbf{M}_4$ (99.4%)		
	1	2	3	4	5	6	7	8	9	10
1	54.4	87.8	0.6	1.1	0.8	9.0	0.0	0.0	1.4	0.1
2	1.8	1.5	0.6	0.2	0.1	0.1	0.0	0.0	0.2	0.0
3	0.0	0.0	91.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	14.0	5.9	2.3	85.3	77.5	73.3	0.0	0.0	0.0	0.1
5	3.5	0.0	1.0	5.8	8.7	0.1	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	99.9	0.0	0.0	0.0
8	22.8	4.2	1.0	7.4	12.3	13.6	0.0	99.4	1.2	0.0
9	0.0	0.4	3.3	0.1	0.7	3.8	0.0	0.6	96.1	0.5
10	0.0	0.2	0.1	0.0	0.0	0.1	0.0	0.0	1.1	99.3
	1.2	15.3	43.1	43.5	48.0	44.5	94.4	50.0	59.2	82.3

Table 6: Best performing selection personalized on new user, test data has different orientation. τ=0.75, cc: 90.6%, overall: 63.6%.

63.6%, but as mentioned before, we need to compare these results with the lower baseline approach. There the ACMS was directly trained on the users data with one phone orientation, which is the general approach for a user who has no knowledge of machine learning techniques. Compared to the lower baseline, we can exceed the recognition rate by 3.6 *PP*. With the recognition rate for the cc we can reach a more significant improvement by 11.7 *PP* up to 90.6%. Here user feedback through the AFI can trigger a repeated personalization via the PTS, or the GTS could have already trained a new ACMS which would perform better with the

different phone orientation.

7.4. Summary and Discussion of Results

Since our Activity Recognition Service (ActiServ) consists of many steps, we now want to summarize and discuss the results. The recognition accuracy for our example implementation compared to the upper and lower baseline is shown in Tab. 7. The numbers show, that we can reach

Service Step	Accuracy of Activity Recognition	Accuracy of Cond. Context Recognition	Delay Until Available
Upper Limit	97.3%	98.3%	days
Lower Limit	60.0%	78.9%	none
Selected ACMS	62.4%	71.2%	seconds
Pers. ACMS	86.0%	89.8%	hours
Pers. ACMS tested with diff. orientation	63.6%	90.6%	hours

Table 7: Recognition accuracy results.

reasonable recognition rates (71% for cc) and high rates ($> 90\%$) for some classes using initial data with ActiServ. This activity recognition is delivered through our service architecture with a delay of only seconds. The personalization step requires a longer computational period (a few hours), where the key factors for duration are the complexity of the ACMS and the efficiency of implementation of the PTS. After personalization we reach recognition rates of up to 90%, with the exception of two classes, where this low accuracy could be accounted for through a better tradeoff in the PTS algorithms. With the ActiServ system we also can exceed the lower baseline limit for opposite phone orientations not included in the GTS training (11.7 *PP* for cond. context). In general, with the GTS constantly training new combinations of ACMS on the user data in the database, we can reach the upper limit of over 97% accuracy with a runtime duration of a few days. Due to the random selection of training data, the delay may also only be a few minutes.

8. Conclusions

We presented an Activity Recognition Service (ActiServ), which aims to support activity recognition on mobile phones for common users. No user knowledge of machine learning is needed, nor are behavioral changes required in order to achieve results. The service provides an interface for annotating initial data, which then suffices for selection of the proper classification modules. Back-end services constantly improve recognition through personalization and simultaneously optimize recognition throughout the ActiServ user community. We presented an evaluation for each of the service's components and compared the results to upper and lower limits. The results indicate that ActiServ produces recognition rates of over 97% for the individual user, while also improving results for users carrying their mobile devices in uncommon orientations while maintining a low processing usage and power consumption profile.

ActiServ becomes especially powerful in combination with [3], where the modular extensibility of the proposed recognition concept is shown. This will maximize the flexibility of activity recognition, and makes it usable for all types of users.

Acknowledgments: This work has been (partially) supported by the NTH (Niedersaechsische Technische Hochschule) School for IT Ecosystems, the European Commission founded project "Cooperative Hybrid Objects Sensor Networks (CHOSeN)" and by the Deutsche Forschungsgemeinschaft (DFG) in the project "SenseCast".

References

[1] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. *PERVASIVE*, 2004.

[2] M. Berchtold and M. Beigl. Increased robustness in context detection and reasoning using uncertainty measures - concept and application. *Ambient Intell. (AmI'09), LNCS*, 2009.

[3] M. Berchtold, M. Budde, H. Schmidtke, and M. Beigl. An extensible modular recognition concept that makes activity recognition practical. *German Art. Int. (KI'10), LNAI*, 2010.

[4] M. Berchtold, T. Riedel, M. Beigl, and C. Decker. Awarepen - classfication probability and fuzziness in a context aware application. *Ubiquitous Intell. and Comp., LNCS*, 2008.

[5] M. Berchtold, T. Riedel, K. van Laerhoven, and C. Decker. Gath-geva specification and genetic generalization of tsk fuzzy models. *Sys., Man and Cyb. (SMC08), IEEE*, 2008.

[6] T. Brezmes, J.-L. Gorricho, and J. Cotrina. Activity recognition from accelerometer data on a mobile phone. In *Proceedings of the IWANN '09*, pages 796–799. Springer, 2009.

[7] S. Consolvo and D. W. McDonald, et al. Activity sensing in the wild: a field trial of ubifit garden. In *CHI*. ACM, 2008.

[8] N. Györbíró, A. Fábián, and G. Hományi. An activity recognition system for mobile phones. *MONET*, 2009.

[9] M. Helmi and S. AlModarresi. Human activity recognition using a fuzzy inference system. *FUZZ-IEEE*, 2009.

[10] Y.-J. Hong, I.-J. Kim, S. C. Ahn, and H.-G. Kim. Mobile health monitoring system based on activity recognition using accelerometer. *SIMPRA*, 18(4):446 – 455, 2010.

[11] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *BSN '06*. IEEE, 2006.

[12] T. S. Saponas, J. Lester, and J. E. Froehlich, et al. ilearn on the iphone: Real-time human activity classification on commodity mobile phones. *CSE Technical Report*, 2008.

[13] A. Schmidt, K. A. Aidoo, and A. Takaluoma, et al. Advanced interaction in context. In *HUC'99*, LNCS, 1999.

[14] D. Siewiorek, A. Smailagic, and J. Furukawa, et al. Sensay: A context-aware mobile phone. In *ISWC '03*. IEEE, 2003.

[15] T. Tagaki and M. Sugeno. Fuzzy identification of systems and its application to modelling and control. *SMC*, 1985.

[16] J.-Y. Yang, Y.-P. Chen, and G.-Y. Lee, et al. Activity recognition using one triaxial accelerometer: A neuro-fuzzy classifier with feature reduction. In *ICEC*. LNCS, 2007.