APPROVAL SHEET

Title of Dissertation: Active Activity Recognition with Context-Aware Activity Recognition

Name of Candidate: H M Sajjad Hossain
Doctor of Philosophy, 2019

Dissertation and Abstract Approved: nimalya

Nimalya Roy
Associate Professor
Department of Information Systems

Date Approved: 04/29/2019

NOTE: *The Approval Sheet with the original signature must accompany the thesis or dissertation. No terminal punctuation is to be used.

ABSTRACT

Title of dissertation: ACTIVE ACTIVITY RECOGNITION WITH CONTEXT-AWARE ANNOTATOR SELECTION

H M Sajjad Hossain, Doctor of Philosophy, 2019

Dissertation directed by: Associate Professor, Nirmalya Roy
Department of Information Systems

The proliferation of wearable and pervasive devices have revolutionized many application domains ranging from health care, sports, entertainment etc. By exploiting the sensor rich wearable and Internet-of-Things (IoT) devices in smart environments, assessment and inference of Activities of Daily Living (ADL) have been a major research proposition over the past decade. A multitude of activity recognition models based on supervised, semi-supervised and unsupervised approaches with significant performance improvements have been posited by the researchers. However, collecting robust label information is fundamental for interpreting the underlying activity data distributions distinctively, and training of the supervised and semi-supervised learning models adequately. Moreover, in practical setting, the limited availability of ground truth information and the variabilities in activities make the activity recognition models impractical for real-world deployment. This ground truth annotation is objectively manual and tedious as it needs considerable amount of human interventions. With the advent of Active Learning with multiple annotators, the burden can be somewhat mitigated by actively acquiring labels of most

informative data instances. However, multiple annotators with varying degrees of expertise poses new set of challenges in terms of quality of the label received and availability of the annotator.

In this thesis, we investigate how this obligation of collecting ground truth information can be mitigated by acquiring labels of most informative data instances using *Active Learning* in activity recognition domain. We propose several active learning enabled activity recognition models which help collect activity labels from human annotators online and reduce the training time warranted while achieving reasonably similar accuracy compared to traditional supervised models. We also propose an active learning combined deep model which updates its network parameters based on the optimization of a joint loss function. In addition, it is both difficult and annoying for an user to provide his own activity information continuously while employing active learning. Introducing multiple annotators can alleviate this adversity, but which annotator can provide reliable label is a fundamental research question. We propose to model the interactions between the users and annotators as relationships spanning across *spatial* and *temporal* space of activity domain. We introduce a novel approach to quantify the strength of relations using the *spatial* and *temporal* information of interactions, type of the relationships, and activities. Our proposed model leverages model-free deep reinforcement learning in a partially observable environment setting to capture the action-reward interaction among multiple annotators. Our experiments in real-world settings exhibit that our active deep model converges to optimal accuracy with fewer labeled instances and achieves $\approx 8\%$ improvement in accuracy in fewer iterations.

ACTIVE ACTIVITY RECOGNITION WITH CONTEXT-AWARE ANNOTATOR SELECTION

by

H M Sajjad Hossain

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland Baltimore County in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Dr. Nirmalya Roy, Chair/Advisor
Dr. Aryya Gangopadhyay
Dr. Shimei Pan
Dr. Jiaqi Gong
Dr. James Foulds
Dr. Hamed Pirsiavash

Dedication

I am dedicating this thesis to my beloved wife Sumaya who kept on encouraging me during the struggles and sacrificed so many things without any hesitation to be always on my side. Thank you very much for your practical and emotional support through out the whole time. I would like to dedicate this to my parents as well for their active support, encouragement and always guiding me to achieve greater heights. I thank my friends and colleagues who have been a source of encouragement and inspiration throughout my life. I would also like to dedicate this work to all the mentors and teachers in my life who had prepared me for what i have achieved today and made me a better person in life.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I would like to thank my advisor, Dr. Nirmalya Roy for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past five years. He has always made himself available for help and advice and there has never been an occasion when I have knocked on his door and he has not given me time. It has been a pleasure to work with and learn from such an extraordinary individual.

My colleagues at the mobile, pervasive & sensor computing lab have enriched my graduate life in many ways and deserve a special mention. My interactions with Md Abdullah Al Hafiz Khan, Nilavra Pathak, Abu Zaher Faridee, Sreenivasan, Neha, Naima, Avijoy and Bipen has been very fruitful.

I am grateful to all of those with whom i have had the pleasure to work during this work. Each of the members of my Dissertation Committee has provided me extensive personal and professional guidance and taught me a great deal about scientific research and life in general.

Lastly, thank you all and thank God!

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Activity Recognition

Recently we have been experiencing rapid development in using ambient technologies and smart devices which has been triggered by the adaptable sensor technologies. These advancements in technologies have broadened the functional domain of mobile and sensor computing. As a result Internet of Things (IoTs) where multimodal sensor setup is combined to perform context aware actions, is becoming an integral component. Wide variety of context aware applications like occupancy detection [2], HVAC control [3], localization, health monitoring [4] etc. , have been introduced. Combining with human activity recognition model, an IoT inspired domain "*Smart Home Technologies*" have been gaining more attention for health monitoring, independent living for senior citizens and making our life comfortable. Smartthings, Vera, Microsoft Lab of Things, openHAB, Ninjablocks, Twine, CASAS:Smart Home in a Box [5] and other automation systems are already in the market for the users. Researchers are extending the capabilities of these automation systems by augmenting different functionalities. One major driving component for these context-aware smart home technologies is to reliably *learn* the human activities using the observable states of ambient and wearable sensor data.

1.2 Activity Labeling

For *learning* and *recognizing* activities of daily living (ADLs) machine learning or rule based algorithms are being exploited extensively. Activity recognition using wearable and ambient sensors in smart home domain is a well studied problem in literature [6] [7] [8]. These trained models are then employed to make decisions according to the domain of the application even in unknown situations. Traditional passive learning approaches are only consistent with the existing class labels. The training for passive learning is carried out in specific experimental setup which is not ideal for real life applications due to a large variety of human activities and underpinning uncertainty with the data capturing. In order to build a robust and stable model, we have to provide vast amount of labeled data- *ground truth information* which is cumbersome and not always feasible. Building adaptive and personalized model for individual users has become a crucial obligation because of the diversity of a same activity across different individuals. For example, the differences in speed of walking, gestures, sleep habits etc., are ambiguous to a general passive learning model. *Active learning* [9] helps us to determine the most informative data points which is a semi-supervised approach. We can actively query the users for labeling informative data points and mitigate the necessity for acquiring a large amount of labeled data points by applying active learning. In activity recognition tasks using ambient or wearable sensors, we have abundant unlabeled data instances which makes active learning as an ideal solution for building an efficient classification model. Although crowdsourcing provides the platform for labeling large amount of

data, the fundamental question is how much of these labeled data will be statistically significant, reliable, noise free and just-in-time. Also from a requester perspective increase demand in data labeling in turn increases the cost of crowdsourcing. Therefore, a cost-effective crowdsourcing model needs to be designed. In this case, active learning can potentially help us to identify potential important data instances that can balance out the trade-off between cost and model performance.

In general, a subset of ADLs are dealt with and most of the proposed activity recognition models are trained with a handful number of activities. On the other hand, it is quite impossible to train the classifier with as many activities as possible. Thus we need an activity recognition system which can dynamically discover unseen activities. Recognition model which can discover unseen activities has been investigated in some of the previous works. [10] proposed a semantic attribute based activity recognition model by creating an activity-attribute matrix for learning unseen activities while [11] proposed a metric learning based approach to reject unseen activities. An activity recognition model using finite state machines which can differentiate unimportant instances of activities for learning a generalized model has been presented in [12]. The use of active learning for activity recognition systems has been investigated by very few researchers. The authors of [13] have proposed a bayesian active learning approach for labeling data in smart homes. Uncertainty based active learning for activity recognition has been employed in [14] [15]. The authors of [16] used entropy based measure to calculate the informativeness of activity data instances. An activity recognition framework called Legion:AR [17] uses active learning and seeks for labels from crowd on demand. AALO [18], a single

habitant activity recognition model for smart homes uses active learning for labeling overlapped activities. In this study, we investigate an active learning enabled activity recognition model which can help identify latent informative and as well as unseen data instances and validate the annotators in the labeling process.

1.3 Quality of Labels

Acquiring labeled data instances is an important task for training supervised and semi supervised machine learning models. In most of the problem domains, both domain knowledge and label information for a learning algorithm are compiled by the human annotators. As a result human intervention is indispensable for collecting ground truths and labels. Therefore the labeling process is manual and the domain experts have to put in hours of effort. As the amount of labeled data influences the validity of the trained model [19], so traditionally we want to label as much data as possible. Labeling large amount of data suggests engaging more domain experts or extending the time for the labeling process. Adapting either of these approaches is a daunting task as it is difficult to find abundant domain experts who can relentlessly provide labels. Consider building an Activities of Daily Living (ADLs) classifier using accelerometry data. If the sampling frequency is 30 Hz and we collect data from a single user for a single day, we end up with approximately 2.5 million data instances. Moreover, the reliability, availability of domain experts, and the incurring costs associated with data annotation process make it a painstaking step while building a machine learning model. It is possible to reduce the complexity

of data annotation by dissecting the problem domain and identifying the relevancy of data with appropriate activity. For example, in case of ADLs we can select a handful number of activities or emphasize more on a specific period of the day instead of considering all of the data. From a machine learning perspective, we can view this as to look for most important data instances which can have significant impact on our classifier. For example, the data instances which are in proximity to the decision boundary, or a representative instance from a dense region of the instance space [20]. By utilizing Active Learning [21], we can therefore select the most informative data instances and pose the label queries to the annotators. There are alternative methods to reduce annotation effort other than Active Learning like utilizing *Crowdsourcing* platforms [22] [23] [24] or training the learning model using unlabeled data indeed [25] [26] [27]. However these approaches can invoke negative impact, for example, annotators in Crowdsourcing platforms are mostly not domain experts and can introduce noisy labels in the model. In addition, learning model trained with only unlabeled data may not be able to portrait the underlying distribution of the data accurately.

Even though active learning can be effective in acquiring labels, but its foundation is built on impractical assumptions - *an annotator who is always available to provide the correct labels to every queries without incurring any cost and the active learner can query as many instances as possible* [28]. In practical, a single annotator may or may not respond to all the queries. Therefore, exploiting multiple annotators seems more practical [29] [30], nevertheless their expertise level may differ drastically. Moreover, the labels received from these imperfect annotators are not always

reliable, so if we pose an important query to a wrong annotator all the efforts will be pointless. Thus based on the informativeness of the selected data instance, its is always desirable to pose the query to the right annotator.

1.4 Annotator Selection

Existing activity recognition methods endure limitation in terms of data scarcity and scalability. The sensors produce an immense volume of data due to high sampling frequency in order to capture fine-grained information without any loss. In order to collect ground truth information, existing works have relied on the video feed heavily where each video frame is mapped with the timestamp [31], and labeled accordingly which is a painstakingly laborious process. In this work, we propose an online annotator selection model while exploiting active learning in smart home activity recognition domain. Every day we perform variety of ADLs, some we do it on our own and some has correspondences. For example, we might cook alone but have lunch or dinner in presence of someone else. While living in a society people are connected to others in the means of relationships, they can be also related in terms of social norms, work places, locations, events etc. Knowledge about certain activities are also associated with these underlying semantic human relationships. For example, if a person has a roommate, the roommate has better insights regarding his daily activities while being at home whereas a person's office colleagues may have better understanding of his office related activities. Note that activities and humans are also tightly connected by two important dimensions - *temporal (time)*

and *spatial (location)*. In most of the cases every human being tends to follow similar activity patterns with respect to these dimensions. For instance, a person if at work may have his lunch at 12:00-1:00 PM everyday but if at home he is likely to have lunch at certain times in a day. Consequently the people who are connected with each other have better concepts about their routines, habits and choices. In addition, how well do others correspond to a person also helps to regulate the level of concepts. While employing *active learning*, a very naive assumption is that we can pose a query to an annotator as many times as needed, which is impractical in most of the real life problem domains. For example, in our context the best annotator will be always the person who actually performed the activity. The second best choice can be a very close friend or a family member. From a greedy perspective we always want to pose our query to these annotators. As a result a set of annotators will be always initiated whereas others will remain idle. We propose to associate a cost to each annotator with respect to their efficiency. Efficient annotators will incur higher cost, as a result given a budget constraint we cannot always invoke highly efficient annotators.

1.5 Contribution

Most of the proposed active learning model focused on finding the most informative point using uncertainty measurement or maximizing error reduction. For some classifiers a basic intuition is followed - *any data instance closest to the decision boundary is considered important and thus queried*. Some approaches are strictly for

SVM classifiers where the intention is to identify the instances from feature space which will maximize the hyperplane margin. These approaches can yield good results and improve the overall performance, however they tend to ignore the prior distribution of the feature space. Prior distribution can be useful for active learning and ignoring them can create sampling bias in the overall system. Deep learning algorithms have recently become popular because of their intrinsic nature of learning good feature representations given input data with complex patterns [32] [33]. Nevertheless, deep models still require large labeled dataset to tune their parameters and adapt to the instances with distinctive scenarios.

In this study, we investigate and propose different active learning techniques in activity recognition domain to mitigate the manual effort needed for ground truth information collection. we propose a novel activity recognition algorithm which harnesses the underlying efficient feature learning capability of deep models and the competency of active learning to collect ground truth information. In most of the existing works [34] [35] informative instance selection process using active learning is administered independently from the principal inference model. Depending on the methodology of the active sampling algorithm, the selection process can become over confident and vulnerable to outliers over time. Thus it is necessary to propagate the learned knowledge of active learning to the base model. In our model, we propose to blend active learning with the deep model by jointly optimizing their respective loss function. By doing this our model can also take advantage of the unlabeled data instances during the training phase. Our model can then endure outliers and adapt to diverse data distributions of the same activity.

Our model applies active learning in a multiple annotator setting to ensure effective use of active learning. The potential annotators for a certain user are accumulated based on the similarity of their context information. We model the similarity between users using the spatial and temporal properties of the user context. The key insight here is that the potential annotators are those with whom we interact with mostly in our day to day life. At first we demonstrate a contextual multi armed bandit model with budget constraints where we take actions based on our similarity metrics. We then further improve the model by employing model-free reinforcement learning.

Chapter 2: Related Work

Activity recognition has been one of the core research areas in ubiquitous computing field for many years [36] [37] [38]. This rapid surge and advancement in learning activity pattern have also assisted a plethora of application domains ranging from sports [39] [40] [41] to health analytics [42]. Activity recognition research have been addressed from two pespectives using *computer vision* [43] [44] [45] and *sensor modalities* [46] [47] [31] [48] [49]. A variety of sensor modalities like accelerometer, ambient sensors, RFID tags, Radar etc [50] [51] have been used in sensor based approach. The major upsurge of mobile and wearable technologies have also accelerated the activity recognition research [52]. Bao and Intille [36] used five biaxial accelerometers which are placed in different parts of the body and detected 20 activities. Hafiz et al. [53] used an array of micro-doppler radars to detect different human body movements. In many of these works, shallow machine learning models like Decision Trees, HMM, SVM etc have been harnessed to find meaningful relationship between the features learned from the sensor data and the performed activity [50]. The widely used features in activity recognition domain include basis transform coding (e.g.signals with wavelet transform and Fourier transform) [54], statistical parameters extracted from raw sensor signals [55] and

symbolic representation [56]. Although these features are widely used in many time series problems, they are heuristic and not task dependent. Activity recognition models also have to address challenges like intraclass variability, interclass similarity, the NULL-class dominance, complexness and diversity of physical activities [55]. Various machine learning models including both shallow learning [57] [58] [59] and deep learning [60] [6] [61] [62] [63] [64] algorithm have been exploited in existing activity recognition literature over the years. Several works have exploited ontology framework to model the semantic relationships between objects and entities and extracted activity reasoning out of the ontology [65] [66] [67] [68]. Activity recognition models exploiting supervised and semi-supervised learning algorithms have to heavily rely on the number of labeled data instances. Some literature have proposed models using unsupervised learning algorithms [25] [69] [70] [71] but if the distribution of the data is not clearly inherent, unsupervised algorithms fail to find the pattern in the data.

2.1 Deep Learning

In most of the cases activity recognition models based on shallow classifiers deal with a set of handcrafted features. These features are then fed to the model, but many of these features are not assured to be relevant and eventually this expedites difficulty in inferring complex activities. Some times statistical features fail to capture substantial facets of human body movements. Due to recent advancements in high performance computing and implementation of deep architec-

tures, researchers are now proposing activity recognition models using deep learning [72] [73] [74] [75] [76] [77]. The authors of [78] proposed a hybrid approach using HMM and deep learning to infer activities from triaxial accelerometer data. Francisco et al [79] demonstrated a model based on deep convolution and LSTM recurrent network which is suitable for multimodal wearable sensors. Simple RBM based model can outperform other activity recognition models has been demonstrated in [64]. The authors also prove that resource usage of traditional RBM in smartwatches is manageable. A deep model using an extension of Convolutional Neural Netowrk and recurrent neural network has been proposed in [80]. *DeepEar* [81] a deep learning enabled mobile audio sensing framework addresses the issue of audio data classification. Li et al. [82] proposed an activity recognition system based on convolutional neural network using passive RFID data. By connecting different Convolutional Neural Networks (CNN) through fusion methods, an ensemble model is built to infer the kitchen related activities in [83]. The authors of [84] investigated the effect of transferring kernel in the convolutional layers in mobile activity recognition domain. Their work considered transfer between users, application domains, different sensors and locations. They validated that kernel transfer can reduce the training time by 17%. Guan et al. [6] proposed a Long Short Term LSTM based ensemble model and tackled the problems of having imbalanaced and problematic data for wearable devices. A hierarchical deep multi-task learning model to detect simple and complex activities , AROMA [85] utilizes convolutional neural network(CNN) to extract features and then applies LSTM to learn the temporal properties of the activities. Various researchers [70] [32] [86] [32] have

also addressed the problem of sensor fusion and heterogeneity in presence of multimodal sensors using deep learning methods. Guan et al. proposed an ensemble model based on LSTM to address the problem of imbalanced dataset and data quality [6]. The authors of [87] proposed a novel model which can infer activities given a sequence of past activities and durations. Several works have proposed to tackle the problem of data insufficiency by applying transfer learning algorithms for deep models [88] [66] [89]. Khan et al. have demonstrated a CNN inspired transductive transfer learning model, HDCNN [88] which can attain high accuracy in absence of any labeled information in the target domain. The authors of [90] proposed a framework called STL which transfers intra-class knowledge iteratively to transform both target and source domains into the same subspace. A personalized inference model which reuses the lower layers of CNN network from the source domain in the target domain and trains the upper layer of the target domain has been proposed in [91].

2.2 Active Learning

Active learning algorithms aim to ease the learning complexity and cost by sequentially selecting optimal number of informative unlabeled data instances to query for their label in order to minimize the prediction error of the classification model. Studies have shown that active learning can help reduce the labeling effort in different domains [92] [93]. The most important step for active learning is to define the informativeness measurement for data instances. In a smart home setting

a wearable sensor reading may belong to *watching television*, a high level activity class and *sitting* micro activity class. In such cases the learner can consolidate both feature and label space (feature-label pair) instead of relying on the distinct features for evaluating uncertainty. Evaluating instance uncertainty is a common approach for measuring the informativeness of a data instance. The uncertainty is measured with respect to the feature space. In this approach the learner can focus more on the data instances which are confusing based on the uncertainty score. Least Confident, Margin Sampling and Entropy learning are the most popular informativeness measures [94]. Another uncertainty resolving approach is searching through hypothesis space where the associated classifier maintains a set of candidate hypothesis space known as version space. The goal of the active learning algorithm is to minimize the cardinality of the version space which depicts maximal change to the current classification model. One popular and significant contribution in this approach is disagreement based active learning [95]. Another approach *Variance Reduction* chooses the instances that minimize the square loss of a learner [96].

Some active learning works have proposed to augment instance correlations where an utility metric from a sample or sample-label space has been defined as a combination of an uncertainty function and a correlation function. In feature based correlation, a similarity measurement [97] or a correlation matrix [98] on features has been utilized to compare pairwise similarities of instances, so the informativeness of an instance is weighted by average similarity on its neighbors. *Label correlation* is widely used for multi label learning. Conventional active learning algorithms considers an oracle to provide the correct label for each query always which is not

ideal. Many of the existing active learning strategies are prone to introduce noise on learning models, as the process of finding an optimal boundary between two classes involves label queries that have lower proximity to the decision boundary and usually these labels induce large noise. The authors of [99] have proposed an algorithm A^2 Learning which works in presence of arbitrary forms of noise. [100] uses a randomized query mechanism and includes importance weights in the calculation of empirical error rates, to compensate for the bias in the sample so that it is possible to obtain rough estimates of the excess empirical error rates. we define the label complexity of A^2 learning [99] by using the disagreement coefficient has been described in [95]. The authors of [101] proposed a bayesian based active learning algorithm where queries are selected sequentially to reduce uncertainty. The authors demonstrate that instead of focusing on minimizing uncertainty, the aim is to drive uncertainty into a single decision region as quickly as possible. Next we discuss the usage of active learning for practical applications such as smart home activity recognition. In [102] the author used a hierarchical clustering for unlabeled instances and then a data instance is picked randomly from the cluster. [93] tries to reduce the classification variance by employing Query By Committee (QBC) method.

2.3 Active Learning & Activity Recognition

The use of active learning for attaining ground truth at low cost in activity recognition systems has been addressed in recent years [13] [103]. The authors of [14] [15] used uncertainty based active learning for activity recognition. The au-

thors of [16] used entropy based measure to calculate the informativeness of activity data instances. Legion:AR [17], an activity recognition framework uses active learning and inquires for labels from crowd on demand. AALO [18] uses active learning for labeling overlapped activities. The authors of [104] validates that by using active learning the actual annotation cost can be reduced by 30-75%. [16] employed an entropy based uncertainty measure to select the most informative instances and validated that only 20% of the training data ensures convergence to the same accuracy while using the whole training set. A variety of research where active learning has been augmented with deep learning have also been proposed [105] [106] [107]. Zhu et al. [108] proposed an active learning algorithm GAAL using Generative Adversarial Network (GAN). GAAL generates new uncertain instances and then requests for annotation to the human annotator. These labeled instances are then added back to the training data set. A cost effective active learning algorithm in conjunction with CNN (CEAL) to discover the large amount of high confidence samples from the unlabeled set for feature learning has been proposed in [109].

Chapter 3: Active learning validation

Our primary goal in this work is to exploit active learning to improve our activity recognition model. First we want the proof of concept that active learning is effective in our scenario. We employ active learning a single activity domain first and then apply it in a more generalized settings with multiple activities. In the first part of this chapter we investigate discretized sleep activity using a wrist worn wearable device.

3.1 Sleep Monitoring

Sleep is the most important aspect of healthy and active living. Right amount of sleep at the right time helps an individual to protect his physical, mental, cognitive health and maintain his quality of life. The most durative of the Activities of Daily Living (ADL), sleep, has a major synergic influence on a person's functional, behavioral and cognitive health. A deep understanding of sleep behavior and its relationship with its physiological signals, and contexts (such as eye or body movements) is necessary to design and develop a robust intelligent sleep monitoring system. In this chapter, we propose an intelligent algorithm to detect the microscopic states of the *sleep*, which fundamentally constitute the components of a good and bad

sleeping behavior and thus help shaping the formative assessment of sleep quality. Our initial analysis includes the investigation of several classification techniques to identify and correlate the relationship of microscopic sleep states with the overall sleep behavior. Subsequently, we also propose an online algorithm based on change point detection to process and classify the microscopic sleep states. We also develop a lightweight version of the proposed algorithm for real-time sleep monitoring, recognition and assessment at scale. For a larger deployment of our proposed model across a community of individuals, we propose an active learning based methodology to reduce the effort of ground truth data collection and labeling. Finally, we evaluate the performance of our proposed algorithms on real data traces, and demonstrate the efficacy of our models for detecting and assessing the fine-grained sleep states beyond an individual. Our focus here is whether active learning can improve the process of sleep monitoring. We first conduct experiment with elemental active learning methodologies for sleep activity and establish the ground for generic activity recognition.

3.2 Motivation

Sleep monitoring has been in the limelight of research due to the growing need of good quality sleep in person's day to day lives. As well as sleep has a bi-directional relationship with the well-being of a person [110]. Sleep disorders such as sleep apnea, COPD (Chronic Obstructive Pulmonary Disease), Chronic renal diseases and various other medical conditions can manifest itself as disruptions in sleep patterns

and thus affect sleep quality [111]. [112] has defined sleep quality as one's satisfaction of the sleep experience, integrating aspects of sleep initiation, sleep maintenance, sleep quantity, and refreshment upon awakening. Sleep quality is highly correlated to exhaustion, discomfort, depression and lack of concentration during the day. The quality of sleep can affect the intuitive symptoms caused by the underlying disease. Clinical studies have suggested that if the sleep quality is improved, the underlying symptoms of a patient might improve too. Monitoring sleep could help a physician to diagnose the underlying condition. Clinically, Polysomnography (PSG) is used for sleep monitoring, which is also the "gold standard" for sleep monitoring and diagnosis of sleep related disorders. PSG captures multiple physiological parameters such as Electroencephalogram(EEG), Electrocardiogram (ECG), Electromyogram (EMG), Electrooculogram (EOG) simultaneously which makes it the best solution to diagnose sleep disorders. PSG provides general sleep measures, such as total sleep time (TST) and sleep efficiency (SE), and also detects specific sleep stages. Conducting PSG requires specialized environment like sleep center or clinic which makes it un-suitable for day to day sleep monitoring. Recent advances in the field of sensor technology and wearables have led to the advent of various sleep monitoring wearable devices such as [113], Actiwatch [114], BASIS watch [115], Misfit Shine, Withings Pulse O2, etc. These devices are commercially available in the market except for the BASIS watch [115]. These devices record accelerometer data, heart rate to monitor sleep quality. Research indicates that Actiwatch and Fitbit captures the TST and the SE well. In other words, it focuses on calculating the duration of sleep which gives a very good insight on someone's sleep hygiene. How-

ever, the BASIS B1 band [115] classifies sleep stages (REM, light sleep, deep sleep) and provides the means to identify patterns and triggers which are causing sleep disturbances [116]. The use of wearable devices has also been proved effective for in-home sleep measurements and evaluations [117].

To evaluate a person's sleep quality, it is important to detect the stages and micro stages of sleep. Researchers view this problem as a classification problem and have been using machine learning algorithms on the data extracted from the wearable devices. Existing sleep monitoring studies have been using supervised learning algorithms where they collect and label a set of training data with pre-defined classes which the system aims to detect. The higher the labeled training dataset, the better is the classification. One of the major challenges faced by researchers while addressing this problem is the collection of the ground truth information. Collecting ground truth without violating privacy (using cameras) of the individuals is an extremely difficult task. It is also a herculean task for the test subjects and other human annotators to annotate the data manually. Leveraging unsupervised algorithm can help to eliminate the requirement for labeled data. However drawing boundaries between similar instances (with respect to properties) but belonging to different classes are difficult using unsupervised learning. The microscopic sleep states are very hard to differentiate based on the accelerometer data as the signal patterns are quite similar for different states. Feedbacks from the users can help us to differentiate between these microscopic states. By employing Active learning (AL) [118], we can acquire feedbacks from the users for important data instances. AL can not only mitigate the manual effort needed for collecting ground truth information but also reduce the

training time. AL retrieves the most informative data instances from a pool of unlabeled data instances and pose them as queries to the annotators. As a consequence we only have to label a handful number of instances. Several researches showed the effectiveness of AL in activity recognition domain [119] [31] [120]. Using AL we can improve our model incrementally based on the feedback provided by the annotators. AL works in an online manner, where we receive a stream of instances and then form a pool from which we select the single most informative instance. In order to get bulk label information Crowdsourcing [121] has been exploited in different problem domains. It has been used in activity recognition domain as well [24] [122]. We can collect ground truth information of unlabeled samples in a bulk which will help to improve our model. In addition, identifying and classifying only pre-defined sleep stages is not sufficient for medical diagnosis. Even though the current state of the art wearable technology cannot replace the PSG in clinical evaluations, obtaining accurate sleep stages has been the aim of every researcher who works on this problem domain. Further, we have identified certain weaknesses in the literature such as patients suffering from nightmare disorder, muscle contractions have not been considered in any study. In this work, we propose a sleep monitoring model using accelerometer mounted wearable device to classify previously unseen various sleep states, that further improves patients sleep hygiene by being able to pinpoint the causes of sleep disturbances. We train our model using supervised and unsupervised learning algorithms and identify the basic sleep states - Rapid Eye Movement (REM), non-REM (NREM) sleep, awake, movement, getting up from bed, getting up and sitting. We also introduce a crowdsourcing model to collect large volume of

labels and also introduce a sleep scoring module.

Wearable devices have become common and accessible to people these days and researchers are making them evolve by incorporating various sensors as attested by the new release of smart watches such as Google Android Wear [123], etc. The world is moving more towards wearable technology which has expedited a plethora of application domains ranging from health care applications [124] to sports [40]. In this study, we have used two different wearable devices - EZ430-Chronos [125] and wActiSleep BT [114] worn on the waist. After collecting the data, we apply a variant of gradient descent algorithm to build a classification model. Further, we apply importance weighted active learning to label the uncertain data points and also incorporate previously unseen sleep states. Active learning improves the annotation effort greatly and improves the performance of classification model. We also exploit crowdsourcing to collect bulk ground truth information in offline. In order to discover abrupt changes on the data streams, increase the classification accuracy, remove noises and provide greater support for informativeness in active learning, we propose an online change point detection algorithm. Finally, we show the results of our proposed algorithm using publicly available benchmark dataset [126] which provides the sleep phases determined by clinical polysomnography where the data was collected using wrist worn device.

3.3 Sleep Studies

In medical studies Polysomnography (PSG) is the major sleep study to diagnose a patient's sleep quality [127]. Polysomnography records biophysiological changes that occur during the sleep. Apart from PSG, some other sleep studies are Multiple Sleep Latency Test (MSLT) and Maintenance of Wakefulness Test (MWT) [128]. These diagnoses are cumbersome and need a lot of prior setup, for example, in case of PSG it requires 12 channels requiring almost 22 wire attachments to a patient. Obviously this imposes a great level of discomfort to the patients and its users. Early literatures [129] involving wearable devices to replicate the Polysomnography results validated the applicability of actigraphy in sleep monitoring. [130] provided further justification for the use of actigraphy in sleep research. Van et al. proposed a model to support the feasibility of a continuous home-monitoring of sleeping trends using wearable devices [131]. Recently the authors of [117] proposed a wearable actigraphy device with low sampling rate for in-home sleep assessment. Several other literatures [132] [133] [134] [135] [136] also ascertain the strength and simplicity of wearable devices in sleep monitoring. The authors of [137] developed a wearable neck cuff system for monitoring physiological signals in real-time. The authors of [138] have developed a light-weight and inexpensive in-ear wearable sensing system that can capture electrical activities of the brain, eye and facial muscles. [138] have used a supervised non negative matrix factorization algorithm to adaptively analyze the signals. A sleep monitoring model using image analysis has been proposed in [139], but it has proved inefficient in

case of low light condition at night. [140] used near-infrared cameras to overcome this challenge but the images still created non-uniformity. A novel sleep monitoring framework- LullaBy to capture and monitor the sleeping environment using microphone, light sensor and motion sensor has been proposed in [141]. Yanzhi et al. [142] put emphasis on the importance of breathing pattern while sleeping and the proposed model captures the breathing sound using signal envelop detection on the acoustic data. The proposed model can detect snore, cough, turn over and get up using the acoustic features. The authors of [143] proposed a real-time system to monitor the sleep conditions where pulse oximeter is exploited to monitor user's pulse oxygen saturation (SPO2) during the sleep process.

Pressure bed sensors have been used to supervise the postures and movements of the users in sleep [144] [145]. Though these methods are unobtrusive and do not create discomfort to the users, but still it has not been streamlined due to its cost and deployment issues. [146] used fine-grained body positions from accelerometer data using WISP tags attached to the sides of a bed. A novel framework for pressure image analysis to monitor sleep postures including a set of geometrical features for sleep posture characterization and three sparse classifiers for posture recognition has been proposed in [147]. The authors of [148] have proposed a sleep monitoring framework comprising of an accelerometer and a pressure sensor. Features pertaining to body motion, respiration, body activity and heart rate were extracted and the proposed framework fuses information from various features and detects the stages of sleep. [149] have proposed a combined framework for fall and sleep monitoring of elderly people by hypothesizing that the acceleration calculated from the accelerom-

eter data will be in the range $0 - 1.5m/s^2$. The authors of [150] have used a custom made accelerometer chip which streams data to an Arduino board. In addition to the accelerometer, a ECG sensor is also used. Features such as Heart Rate and RR interval were extracted and Kushida's algorithm-derived equation was used to differentiate the sleep stages.

Sleep related research are gaining attention due to the recent proliferation of low-cost easy-to-deploy technologies based on mobile and ambient sensors and its large penetration in the market. Commercial wearbale devices, such as Fitbit [113], Zeo [151], Actigraph [114], Jawbone, Sleep Tracker etc have been used extensively these days for monitoring sleep and activities of daily living (ADLs). iSleep [152] uses the built in microphone sensor of smartphone to detect the events which are closely related to sleep quality like body movements, coughing, snoring etc. The authors of [153] used the accelerometer sensor of the smartphone to track the sleep duration and user movement patterns. [154] proposed a passive approach to track some stationary features, such as user silence, ambient light, phone usage and charging etc for monitoring sleep habits and developed a mobile application BeWell [155] for unified health monitoring. [156] used the daily context information of an user to define the sleep quality. Sleep Hunter [157] used the accelerometer and microphone sensors of the smartphone a fine-grained detection of sleep stage transition for sleep quality monitoring and intelligent wake-up call. Mimo Baby Monitor is a bodysuit for infants aged 0 - 12 and incorporates respiratory sensor, accelerometer sensor, and temperature sensor to measure the physiological signals, body movements, and temperature respectively. These signals are transmitted via bluetooth to an online

data cloud and to the caretaker's mobile [158]. Using several android apps like *Sleep As Android*, *Sleep Time Smart Alarm Clock* [159], *Sleep cycle*, *SleepBot*, etc. it is possible to monitor the quality of sleep. Other commercial unobtrusive technologies like Beddit [160], Hello Sense [161] can also monitor sleep. Hello Sense also tracks the quality of the sleeping environment. [162] proposed to use change-point segmentation on PSG data to differentiate macrostructural organization of sleep. A point process based novel model for the assessment of heart rate variability and respiratory sinus arrhythmia based on PSG data has been proposed in [163].

3.4 Overview of SleepWell Framework

Sleep is not just a dormant part of our lives, we remain very active and pass through several important stages of sleep. Interference or disturbance in these states can cause impatience, drowsiness and lack of concentration during the regular activities of daily living. Therefore for maintaining a good sleep hygiene we have to sleep a certain amount of time in each of those sleep states. There are two main types of sleep states:

- Non-Rapid Eye Movement (NREM) (also known as *quiet sleep*). NREM consists of three states (stage-1, stage-2, stage-3).
- Rapid Eye Movement (REM) (also known as *active sleep*).

A complete and healthy cycle of sleep consists of a progression from states 1 to 3 before reaching REM state, then the cycle starts over again. If the REM sleep is disrupted and the person wakes up, the person's circadian cycle is disrupted.

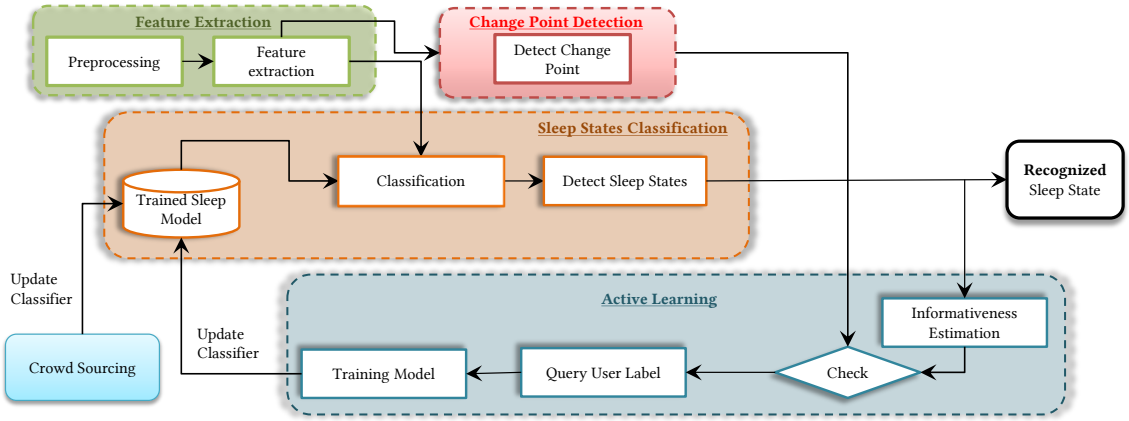


Figure 3.1: An architectural overview of Sleep Well [1] Framework

In order to complete the cycle the person will move to REM state directly next time. Thus it is very important to sleep a good amount of time each day and maintain a good sleep cycle. REM sleep is considered as active sleep because in this state people dream. If a person is having a nightmare disorder too often it is possible that he/she is having problems to complete the sleep cycle. In this work, we first focus on properly classifying the sleep cycle into these finer states. We also propose to integrate some other broader intermediate sleep states such as *movement*, *getting up from bed* and *getting up and sitting*. These other states would help to identify the casual and formal causes of sleep disturbance and sleep latency and provide meaningful insights on designing scalable sleep monitoring technologies and automated assessment methodologies.

3.4.1 Sleep Well Architecture

In figure 3.1 we demonstrate our proposed framework. Our proposed framework consists of the following logical components.

- **Feature Extraction**: After collecting raw sensor data, this component pre-processes and extracts the low level signal features as shown in Table 3.1 from the processed raw sensor data. (Details in 3.5.1).

- **Change Point Detection**: After extracting features and analyzing the sleep data, we noticed that change point occurs in sleep transitions (transition from one stage to another, for example unconscious movement during sleeping, waking up, being restless in bed etc.). The importance of these change points has proven to be very effective as it help removing noises in the data and detect the exact point of the sleep transition. For example, when a person gets up from the bed and starts walking, the accelerometer readings other than sleep classes become noisy. Therefore by identifying change points we can partition the data and have more fine-grained information for easing the training effort. (Details in 3.6).

- **Classification**: At this stage we train our model using the features from processed raw sensor data and build up our classification model to recognize the several intermediate sleep states. We investigated an online gradient descent [164] as our classification algorithm. This is different from traditional gradient descent by dealing with importance weights to collaborate with active learning and learning reductions. To calculate the average loss during the classification process we propose to use squared loss function (Details in 3.6.1).

- **Active Learning**: After feeding the test data into our classification model

and getting the prediction, active learning helps to calculate the informativeness of each data points. If any data point falls within an uncertain space and while predicting it is found to be the most informative, then if the actual label of the point is provided it would have more significant impact on the classification model. This component then initiate prompt for "*query user label*" and get the ground truth from the user. Subsequently the labels are then used for re-training and updating the model. This component helps to ensure better classification accuracy with minimal user feedback. This also helps to scale the sleep monitoring model across multiple individuals. The input from change point detection method strengthens the active learning query selection by asking the user to label the appropriate sleep state transitioning step. (Details in 3.7).

- **Crowdsourcing**: For large scale deployment we apply crowd sourcing for reducing the ground truth collection effort. The problem with existing crowd sourcing platforms is that there is no standard to evaluate the quality of the workers. In our model we calculate two parameters for each worker - *reliability* and *awareness*. Using these two parameters we rank the user and get the most out of our crowdsourcing platform (Details in 3.8).

3.5 Sleep Well Framework Design

In this section we describe in details the design of our Sleep Well framework. We first discuss about several micro-states of sleep and feature extraction process.

Table 3.1: Features used for Sleep Micro-States Classification

	Name	Definition
Time Domain Features	Mean	AVG ($\sum x_i$), AVG ($\sum y_i$), AVG ($\sum z_i$)
	Mean-magnitude	AVG $\sqrt{x_i^2 + y_i^2 + z_i^2}$
	Magnitude-Mean	$\sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2}$
	Variance	VAR ($\sum x_i$), VAR ($\sum y_i$), VAR ($\sum z_i$)
	Co-Variance (Two-axis correlation)	$cov(xy)$; $cov(yz)$; $cov(xz)$
	Standard Deviation	$\sigma_x = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$; $\sigma_x = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$; $\sigma_x = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$
Frequency Domain Features	FFT-Magnitude	$m_j^{(x)} = \lvert a_j + b_j i \rvert$; $m_j^{(y)} = \lvert a_j + b_j i \rvert$; $m_j^{(z)} = \lvert a_j + b_j i \rvert$
	FFT-Energy	$\frac{\sum_{j=1}^{N}(m_j^2)}{N}$ for x, y, z respectively

Next we discuss an online change point detection algorithm to have a better handle on the microscopic sleep state classification problem.

3.5.1 Sleep Event Detection and Feature Extraction

We extract low-level features using each of the three components of the triaxial accelerometer signal to capture the aspects of movements while sleeping. We use both time and frequency domain features in our framework. As the user is not physically active while sleeping, very few number of movements are involved, so we choose a lower sampling frequency. We extract features from data using windows of 60 samples, corresponding to 1 second of accelerometer data. From each window we calculate the features mentioned in Table 3.1. Time domain features help to differentiate dynamic to static movements. The frequency domain features help

identifying patterns within acceleration data, which aids in discriminating discrete movements and their intensities.

3.5.2 Feature Selection

Our model is primarily focused on community scaling, so fewer features will scale the model computationally effective if we can achieve similar accuracy with more features. We select the subset of features, best fit for our model by applying Restricted Forward feature Selection (RFS) algorithm. It was performed in two steps. First we applied Forward feature Selection (FS) algorithm which ranks the features in decreasing order of their accuracy. The FS algorithm iterates through the feature space and measures the Leave-One-Out-Cross-Validation (LOOCV) error for each component in the feature space $\{f_1, f_2, f_3.....f_N\}$. In case of traditional FS, after the first iteration, FS calculates the best individual feature f_i. In the next iterations, FS finds the best subset consisting of two components, f_i and one other from remaining $N-1$ features. In the following iterations, FS ranks more features and evaluates the subset accordingly, so that after N iterations, the winner is the overall best feature set in these N iterations. In the second step we invoke the RFS to restrict the number of features to rank at each iteration. After the first iteration we consider only the first $N/2$ ranked features for the following iteration. After adding another feature to the winner of the first iteration at the second iteration, we consider the first $N/3$ components of the remaining ranks. RFS repeats this process until it finds the best m feature sets. The difference between conventional FS and

RFS is that RFS considers only a part of the remaining ranked features, whereas FS considers all the features. Out of 8 features, the feature selection algorithm chose 4 features (FFT-Magnitude, FFT-Energy, Mean-Magnitude and Co-Variance) which help to attain classification accuracy closer to using all the 8 features.

3.6 Change Point Detection

Change point detection helps find the abrupt variations in the sleep data stream. While some change points provide meaningful insights and some not, our motivation in this work is to find the sleep transitions by calculating the change points (abrupt signal changes) and distinguish between the important and unimportant changes. This is not only helpful to detect the sleep-related events appropriately but also help remove noisy data points from the dataset. We develop a Bayesian online changepoint detection [165] based algorithm for finer sleep related event identification and online data noise reduction.

We first partition the entire sleep dataset in different regions based on a run length [165]. Let, $x_{1:N} = \{x_1, x_2, x_3, ...x_N\}^T$ denote the N data points observed over time T which is divided into non-overlapping partitions. Consider if we find K change points then let the data set of partitioned data be $\{\rho_1, \rho_2, \rho_3, ..., \rho_k\}$ at time indices $\{t_1, t_2, t_3.....t_k\}$ where by definition $t_0 = 0$ and $t_{k+1} = N$. The discrete probability distribution over a time interval t_i to t_j is denoted by $g\ (t_i - t_j)$. Each partition ρ_t denotes a segment of the data at time t. The length of the each partition or time since the last change point occurred, is defined as "run length", r. The run

length goes back to 0 if change point occurs, otherwise it increases by 1 as follows.

$$r_n = \begin{cases} 0, & \text{if changepoint occurs at } (n-1) \\ r_{n-1} + 1, & \text{otherwise} \end{cases}$$

The conditional probability that a change point occurs on time t_k after the last change point at time t_{k-1} is

$$P(t_k|t_{k-1}) = g(t_k - t_{k-1}), \text{ where } 0 < k-1 < n \tag{3.1}$$

We assume that the predictive distribution of a change point at any time instant t only depends on the recent data. So the change points are assumed to follow markov process. Thus the prior probability of a change point at a time instant t_k is dependent on the probability distribution of the observed data over the time interval and the preceding change point.

$$P(t_k) = \sum_{i=0}^{k-1} g(t_k - t_i) P(t_{k-1}) \tag{3.2}$$

The change point detection algorithm finds the number of change points and their position by calculating the posterior probability $P(r_n|x_{1:n})$ and integrating it with the predictive distribution $P(x_{n+1}|x_n)$. We do this by calculating the joint distribu-

tion of the current run length and observed data $P(r_n, x_{1:n})$.

$$
\begin{aligned}
P(r_n, x_{1:n}) &= \sum_{r_{n-1}} P(r_n, r_{n-1}, x_{1:n}) \\
&= \sum_{r_{n-1}} P(r_n, x_n | r_{n-1}, x_{1:n-1}) P(r_{n-1}, x_{1:n-1}) \\
&= \sum_{r_{n-1}} P(r_n | r_{n-1}) P(x_n | r_{n-1}, x_{n-r:n}) P(r_{n-1}, x_{1:n-1})
\end{aligned} \tag{3.3}
$$

Where $P(r_n|r_{n-1})$ is the transition probability and $P(x_n|r_{n-1}, x_{n-r:n})$ is the data segment likelihood probability. We calculate the transition probability using equation

$$
P(r_n|r_{n-1}) = \begin{cases} h(r_{n-1}+1), & \text{if } r_n = 0 \\ 1 - h(r_{n-1}+1), & \text{if } r_n = r_{n-1}+1 \end{cases}
$$

where $h(x) = g(x)/\sum_{i=x}^{\infty} g(i)$. We calculate the posterior probability using Bayes's rule:

$$
P(r_n|x_{1:n}) = \frac{P(r_n, x_{1:n})}{\sum_{i=0}^{n-1} P(r_i, x_{1:i})} \tag{3.4}
$$

We calculate the posterior probability of the run length at that time index which corresponds to a new data sample. The pseudo code of this procedure is summarized in Algorithm 1.

3.6.1 Classification

We classify the sleep states using an online gradient descent method which leverages the importance weight on streaming data samples. To build up our classification model accurately we consider other sleep contexts such as body movements,

Algorithm 1 Change Point Detection

1: Initialize: $P(r_0) = 1$

2: **for** Each new data point x_n **do**

3: Calculate the data segment likelihood probability,

4: $P(x_n|r_{n-1}, x_{n-r\,:\,n})$

5: Calculate the transition probability, $P(r_n|r_{n-1})$

6: Calculate the joint distribution, $P(r_n, x_{1\,:\,n})$

7: Find the posterior distribution on current run length, $P(r_n|x_{1\,:\,n})$

8: Calculate the predictive distribution of x_n based on previous observation. $P(x_n|x_{n-1})$

9: **end for**

but most of the sample data points resemble stationary states during the sleeping. Online gradient descent with importance weight aware updates [164] helps to overcome this limitation of data by assigning weight to classes with lesser data points. The key principle here is: *The assignment of importance weight h to a sample that make it appears like a regular example of h times in the dataset.* We assume C is our classification model and use squared loss function for examining the consistency of C. The goal of our classification model is to minimize the loss function which reflects better accuracy. After each iteration of gradient descent, C is not altered, rather it is improved by adding an estimator h in order to optimize the loss function. We assume y is the true label and p is prediction of our model, where $l(p, y)$ is the

loss function as shown in Eqn. (3.5). At each step C is updated using Eqn. (3.6)

$$l(p, y) = \frac{1}{2}(y - p)^2 \tag{3.5}$$

$$C_{m+1} = C_m + h(x) \tag{3.6}$$

Let w be the vector of weights and the training set is a set of (x_i, y_i, h_i), $i = 1, ..., T$ where x_t is a vector of d features. For linearity we assume $p = w^T x$. Our goal is to assign w in such a way so that the model C converges to the optimized solution. Assigning weight to a data point (x, y), h times in a row have a cumulative effect with scaling factor $k(h)$ as shown in Eqn. (3.7). This scaling factor is defined by Eqn. (3.8) where η is the learning rate [164]. At each iteration this weight is updated accordingly to the loss function l. Our proposed classification algorithm for finer non-stationary sleep states detection is shown in Algorithm 2.

$$w_{i+1} = w_i - k(h)x \tag{3.7}$$

$$k(h) = \frac{p - y}{x^T x}(1 - e^{-h\eta x^T x}) \tag{3.8}$$

3.7 Active Learning Based Community Scaling

Our goal in this work is to scale the sleep monitoring model to a community of individuals. While a significant research has been done on sleep monitoring and assessment and intervention strategies, lack of novel scaling algorithms prohibit the deployment, large scale validation, and acceptance of these technologies for healthy lifestyle, smart health and independent living applications. In this section

Algorithm 2 Importance Weighted Sleep Classification

1: **Input:** Extracted feature vectors from raw data.

2: **Output:** Predicted Sleep Class y.

3: Update the importance weight of the data points.

4: *Initialize:* $\forall_i w_i \leftarrow 0$

5: Get the feature vector for data point x_i

6: Predict the class label y_i for all x_i

7: Calculate the scaling factor k(h)

8: **for** $i = 0$ to N **do**

9: Calculate the weight w_i for each x_i

10: *Update:* $w_i \leftarrow w_i - \frac{p-y}{x^T x}(1 - e^{-h\eta x^T x})x$

11: **end for**

we investigate how active learning based machine learning algorithms help build an informative model in presence of a minimal labeled datasets. We also depict how change point detection based time-series data analytics methodology help reduce the data uncertainty and guide to the selection of most informative query.

Active learning has been proved to be very effective by combining it with supervised learning when a large pool of unlabeled data is available. Though traditional passive learning takes the initiative to label the unlabeled data randomly, but most of the data points which are selected randomly does not ensure better classification. It is difficult to collect all of the sleep related ground truth information from the user though by using the accelerometer sensor it is possible to broadly monitor the user sleep behavior and the specific sleep duration. To collect more fine-grained de-

tails about the sleep we train our proposed gradient based classifier with the causes of sleep disruption (such as waking up from nightmare, muscle cramp etc). By applying active learning we propose to collect the labels of these informative data points so that our model can better classify the sleep stages and conditions and help scale this model in presence of minimal amount of ground truth. While applying active learning, one constraint is we have to assure that the whole labeling process doesn't become too intrusive. Crowd-sourcing can help us overcome this constraint by collecting a large amount of labeled data via arbitrary participants and provides aid in community scaling.

3.7.1 Query selection

In the following we briefly discuss the query selection approaches for active learning:

- **Query Synthesis**: The active learner asks the human annotator for "label membership" by using membership queries. In this approach the learner generates instances rather than sample from existing unlabeled set. But the problem with this approach is human annotator may have difficulty interpreting and labeling arbitrary instances.

- **Stream based selective sampling**: Each unlabeled instance is drawn at a time from the input source and the learner may decide instantly whether to query the instance or not. As we are using online classification algorithm and the data are processed in stream, we use this sampling strategy for our active

learner.

- **Pool based sampling**: Evaluates and ranks the entire collection of unlabeled data before selecting the best query from a pool of instances.

3.7.2 Sampling metrics

Different sampling metrics such as least confident, margin sampling or maximum entropy based sampling are common in active learning algorithms. We propose to use the importance weighted active learning approach to build our community scaled sleep monitoring model [166]. To decide which points are most informative, we first calculate the utility measurements of unlabeled data points. Whether a data point x_t will be queried or not depends on the history of labels seen so far based on our change point detection, gradient based classification and the identity of the point. If a change point is detected at data point x_t at time index t_n, and the label of x_t is inconsistent with the label of current run r_n, we invoke active learning. A probability measure p_t is maintained for each data point x_t. A coin flip, $Q_t \epsilon \{0, 1\}$ with $E[Q_t] = p_t$ determines whether the data point will be queried or not. If the data point is queried based on the past history, then we update importance weight by $\frac{1}{p_t}$.

The active learning algorithm maintains an effective hypothesis space H_t through out the process. Initially, H_t contains all of the hypotheses from global space H.

The expected loss of a hypothesis, $h \epsilon H$ at time T is defined by Eqn. (3.9).

$$L_T(h) = \frac{1}{T} \sum_{t=1}^{T} \frac{Q_t}{p_t} l(h(x_t), y_t) \tag{3.9}$$

As it progresses, H_t becomes narrower by taking a subset, and ensuring that the factual loss of H_{t+1} is not much worse than the smallest loss, L_t^* in H_t.

$$H_{t+1} = \{h \, \epsilon \, H_t : L_{t+1}(h) \leq L_t^*(h)\} \tag{3.10}$$

For each data point x_t, the active learning algorithm looks at the range of predictions and their losses by hypotheses in H_t and sets the sampling probability to the size of this range.

$$p_t = \max_{f,g \epsilon H_t} \max_{y} \; l(f(x_t), y) - l(g(x_t), y) \tag{3.11}$$

If the range is too high than the rejection threshold then the hypotheses disagree greatly with each other. This certifies that the current prediction of x_t lies in the uncertain region. The active learning algorithm then queries for the label to settle the uncertainty. Our proposed active learning algorithm for largely reducing the micro-sleep states annotation effort is shown in Algorithm 3.

Apart from using only predefined class labels, the user can introduce new unseen class along with indicative attributes with the help of active learning. While prompting for label of data point x_t, we also collect the reason for their choice of label in restricted number of words. We find specific attributes from the provided reason and associate that attribute with the data point x_t. For example, if a user labels a data point as "getting up & sitting" and specifies the reason as "woke up from nightmare", Sleep Well framework extracts the attribute "Nightmare" from the

provided reason. Subsequently we re-evaluate our classification model and apply a recursive classification to associate the provided attributes to similar data points. This help our model to achieve microscopic sleep state classification, and finer evaluation for more elaborative and accurate diagnosis of patients and eventually scale the model beyond an individual premises.

3.8 Crowdsourcing

Crowdsourcing has been proved to be an effective component for collecting labels in many machine learning applications. Large scale data processing and annotating the data with multiple annotators or experts alleviate the traditional process for gathering ground truth data which is lengthy, costly, and time consuming. By using crowdsourcing though we accumulate a large volume of labeled data, but we also increase the risk of introducing a lot of noisy and ambiguous labels into our classifier. So it is necessary to identify potential reliable annotators and limit the effect of introducing noisy labels. On the other hand, a major challenge in crowdsourcing is to verify the provided labels. To tackle this we propose to calculate the inter-annotator agreement using Fleiss' Kappa statistics and identify the proper label for the data point in concern. In our model we rank the annotators based on their reliability and awareness of the feedback. Here reliability refers to the correctness of the feedback and awareness indicates the willingness of the annotators. For each annotator we maintain a probability measurement, δ^i_{tk}:

Algorithm 3 Active Learning with Importance Weighted Sampling

Input: L = set of labeled instances $\{(x,y)^l\}_{l=1}^{L}$ U = set of unlabeled instances $\{(x)^u\}_{u=1}^{U}$ A classifier model, C_θ

Output: Updated classifier model, C_θ.

Updated importance weight of queried data points.

for every instance in U **do**

    set p_t of instance x_t using equation (3.11)

    $y_t \longleftarrow$ Prediction of C_θ for x_t

    $queried \longleftarrow False$

    if x_t falls in between successive run r_{n-1} and r_n using the posterior probability $P(r_n|x_{1:n})$ **then**

        if y_t is not same as the label of current run r_n **then**

            query label y_t; $L_t \longleftarrow L_{t-1} \cup \{x_t, y_t, \frac{1}{p_t}\}$

            $queried \longleftarrow True$

        end if

    end if

    if p_t is greater than rejection threshold and $queried = False$ **then**

        query label y_t; $L_t \longleftarrow L_{t-1} \cup \{x_t, y_t, \frac{1}{p_t}\}$

    else

        $L_t \longleftarrow L_{t-1}$

    end if

    Update the hypothesis space H_t

end for

$$\delta_{tk}^{j} = P(y^{j} = k | y^{j} = t) \tag{3.12}$$

In equation 3.12, δ_{tk}^{j} denotes the probability that the annotator j provides the class label k to an instance given that the true class label is t. Our goal is to learn the parameter δ_{tk}^{j} for each annotator. Suppose there are R annotators. If the number of data instances is N, we initialize a $N \times R$ matrix M. M_{ij} denotes the label of instance i provided by annotator j. Given that y_i^t is the true label for instance x_i, we assume that $y_i^1, y_i^2 ..... y_i^R$ are independent. Let $y = \{y_1^t, y_2^t, y_3^t ..... y_N^t\}$ be the set of true labels for the data instances. In our model δ is the reliability parameter. Our goal is to learn an optimal estimator $\hat{y}$ of y to minimize the imposed error by provided noisy data points. By taking Beta prior on the reliabilities we can formulate a maximum likelihood estimator $\hat{\delta}$ of δ using equation 3.13. By taking y as hidden variable we can estimate $\hat{\delta}$ using expectation maximization [167].

$$\hat{\delta} = \arg\max_{\delta} \log P(\delta | M) = \arg\max_{\delta} \log \sum_{y} P(\delta, y | M) \tag{3.13}$$

We calculate the awareness of a certain class c, a_j^c by taking the percentage of data of class c labeled by j^{th} annotator. We assign weight w_j^c to each annotator by taking the product of their respective reliability of a certain data instance i and awareness. Based on the weight for each class we rank the annotators with respect to each class.

$$w_j^c = \delta_{tk}^{j} a_j^c \tag{3.14}$$

To verify our estimator we calculate the inter user agreement by using Fleiss' Kappa statistics. The kappa k is defined in equation 3.15. In equation 3.15, $1 - \bar{P}_e$ gives the degree of agreement that is achievable, and, $\bar{P} - \bar{P}_e$ gives the degree of agreement actually achieved. If the annotators agree with each other then $k = 1$, if not then $k \leq 0$.

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{3.15}$$

Let n be the number of annotation per annotator, and let k be the number of classes. Then n_{ij} represent the number of annotators who assigned instance i to the j class. First calculate P_j, the proportion of all assignments which were to the class j:

$$P_j = \frac{1}{Rn} \sum_{i=1}^{R} n_{ij}, \qquad 1 = \frac{1}{n} \sum_{j=1}^{k} n_{ij} \tag{3.16}$$

Now calculate P_i, which denotes how many annotator pairs are in agreement, relative to the number of all possible annotator pairs.

$$\begin{aligned} P_i &= \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij}(n_{ij} - 1) \\ &= \frac{1}{n(n-1)} \sum_{j=1}^{k} (n_{ij}^2 - n_{ij}) \\ &= \frac{1}{n(n-1)} [(\sum_{j=1}^{k} n_{ij}^2) - (n)] \end{aligned} \tag{3.17}$$

Now we can measure $\bar{P}$, which is the mean of all P_i and $\bar{P}_e$ using P_j:

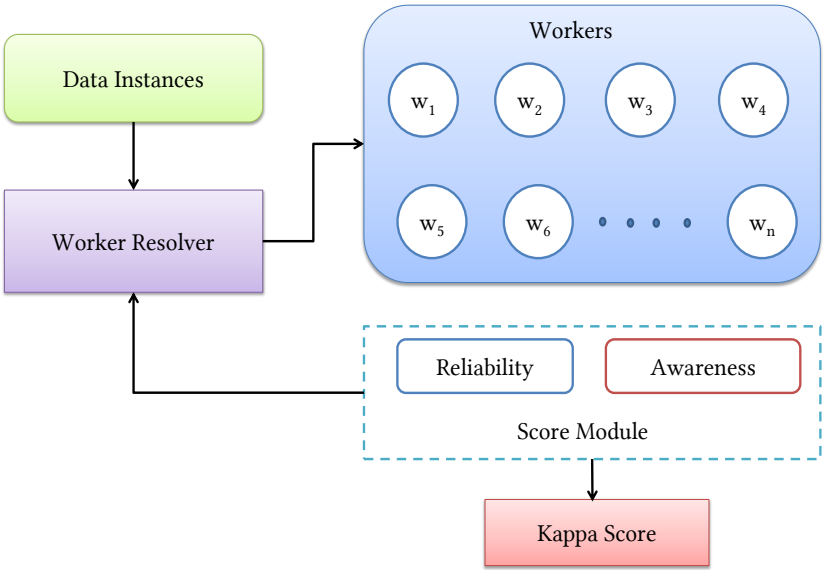


Figure 3.2: An architectural overview of Crowdsourcing

$$\begin{aligned}\bar{P} &= \frac{1}{N}\sum_{i=1}^{N} P_i \\ &= \frac{1}{Rn(n-1)}(\sum_{i=1}^{R}\sum_{j=1}^{k} n_{ij}^2 - Rn)\end{aligned} \tag{3.18}$$

$$\bar{P_e} = \sum_{j=1}^{k} p_j^2 \tag{3.19}$$

In figure 3.2 the architecture of our crowdsourcing platform is shown.

3.9 Sleep Well Framework Evaluation

To evaluate our framework we focus on the following specificities. i) The performance of different classification algorithms in comparison to our classification approach, ii) Cross-user performance by building model with a user's sleep model and testing with someone else's model, iii) Performance of our framework using

different wrist-band devices with accelerometer sensor, iv) Impact of active learning on our model, v) Precision of classifier when sleep attributes are introduced in the model by active learning.

3.9.1 System Implementation

We have implemented and tested our model by using two separate devices - wActiSleep-BT [114] and EZ430-Chronos [125], both of these devices contain 3-axis accelerometer sensor. EZ430-Chronos device also has heart monitor, pressure and temperature sensor. We collected raw accelerometer data from both of these devices using API provided by the manufacturers. We implemented our own software to extract raw data using C# programming language and then extracted the features using python numpy library. We sampled the data in 60Hz frequency. For importance weighted classification and active learning we used the machine learning tool-Vowpal Wabbit [168].

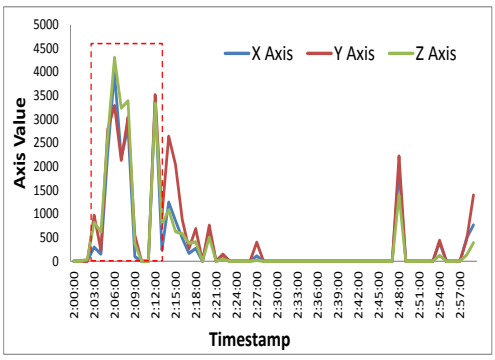


Figure 3.3: Accelerometer reading when standing

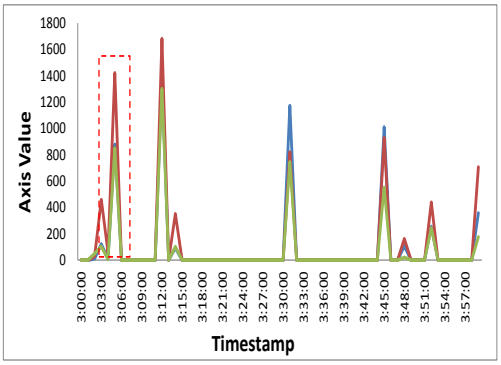


Figure 3.4: Accelerometer reading when lying

3.9.2 Ground Truth Collection

We asked the users to log their sleep habits using sleep diary to correctly label the data points. We asked the participants to note down their sleep routines (preferred sleeping postures, regular hours of sleep, light intensity and sleep latency) each day of the experiment. There were many challenges involved while collecting the ground truth from the sleep diary. For example, consider two different scenarios, 1) the user is *awake & lying* and 2) *awake & not lying*. In case of stationary states (when the user is not moving but he is either lying or just sitting in the bed) the accelerometer readings are almost identical. Also when a user gets up in the middle of the night and performs some activities (checking his phone, going to bathroom etc.), there are movements involved. It was challenging to identify which movements were during sleep and which were due to some activities. The user was unable to correctly state the reason of movements in some cases. In Fig. 3.3 and 3.4, we can see two different movements (awake and standing, awake and lying). The user went to bathroom at 2:03 AM and came back to bed at 2:12 AM. On the other hand at 3:03 AM, the user was moving while lying. Therefore to assist the ground truth collection, we investigate a posture analysis using the inclination of the accelerometer. We observe that when perpendicular to gravity, accelerometers are more sensitive to small changes in inclination, but as the inclination increases the accelerometer becomes less sensitive to it. To resolve this issue we propose to use two axis. As we are using wrist worn bands, inclination of axis y and z are used to define the posture of the user. The z axis measures the direction of the gravity

Figure 3.5: Raw accelerometer data from Dataset 3.9.3.1.

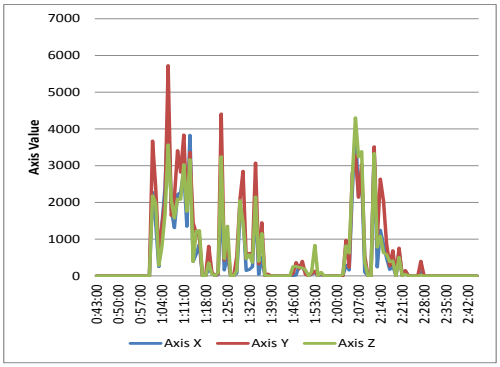


Figure 3.6: Raw accelerometer data from dataset 3.9.3.2

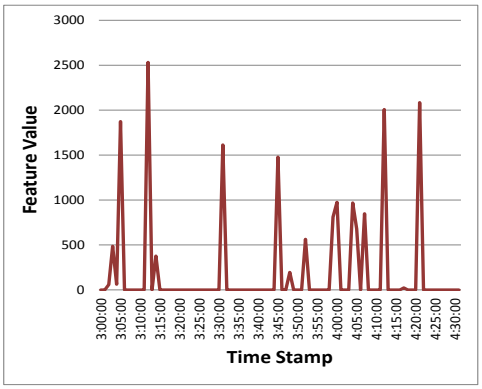


Figure 3.7: Timestamped mean-magnitude feature value.

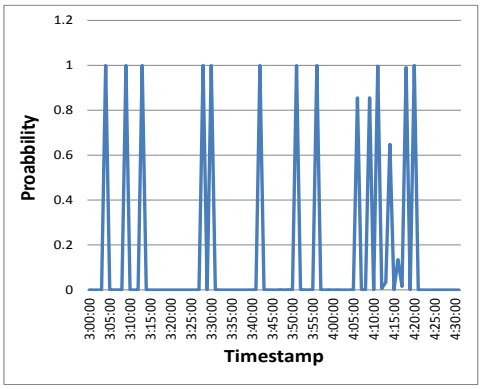


Figure 3.8: Detected change points associated to figure 3.7

in the horizontal position, so coupling with the inclination of x axis help infer the posture of the user. We calculate the inclination of the device by using Eqns. (3.20) and (3.21).

We faced a challenge to define the threshold values for these inclinations as different users have different sleeping postures. We experimented with different sleep positions (On side, Face down, On your back) and calculated the inclination of the device in those positions. We ran the J48 decision tree classifier on the postural

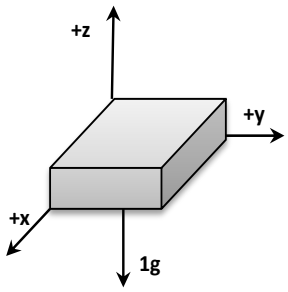


Figure 3.9: Accelerometer orientation

$$\theta_y = \tan^{-1}\left(\frac{y}{\sqrt{x^2 + z^2}}\right) \tag{3.20}$$

$$\theta_z = \tan^{-1}\left(\frac{z}{\sqrt{x^2 + y^2}}\right) \tag{3.21}$$

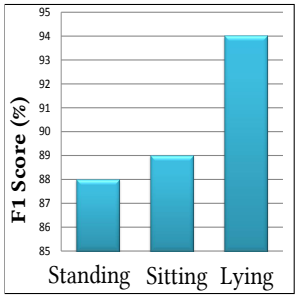


Figure 3.10: Inclination measurement accuracy

data. Based on the results of the classifier we defined the inclination threshold for different states, such as if $\theta_y < 16°$ then the user is standing, if $16° < \theta_y < 61°$ the user is considered sitting, and for $\theta_x > 61°$ the user is considered lying. Fig. 3.10 shows the results of our posture calculation using the inclination method. We also installed couple of motion sensors in the environment to strengthen our ground truth collection. We put two motion sensors (Aeotec Multisensor [169]) near both sides of the bed and another one mounted near the user's body when he/she is sleeping. The sensor mounted near the body captures the motion when the user is in the bed while the other two on the sides of the bed monitor when the user is out of the bed. While extracting information from the sensor, we assumed that consecutive two data points from the sensors mounted on the sides correspond to getting in and then getting out of the bed. These multisensors also have built in light sensor, so we can detect the light condition in the sleeping environment using these sensors. Now we are able to validate the movements of the users and calculate the overall time he/she remained out of the bed efficiently by consolidating inclination measurement, motion sensor and data from sleep diary. We were able to label most of the data

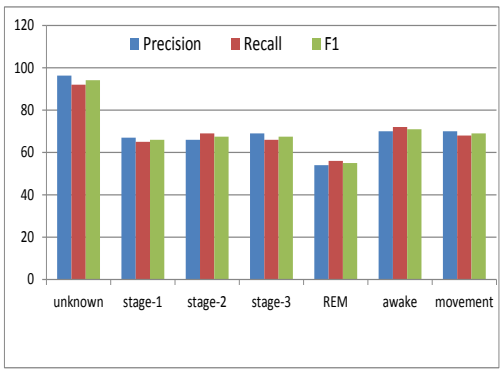


Figure 3.11: Precision, recall and F1-measurement for inter user classification (dataset 3.9.3.1)

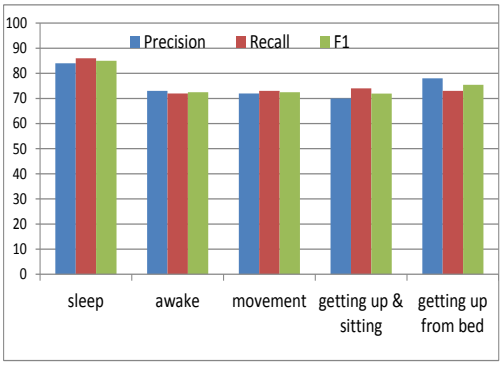


Figure 3.12: Precision, recall and F1-measurement with inter user classification (dataset 3.9.3.2)

points correctly and remove noisy data points.

3.9.3 Datasets

We use real data traces collected from ≈ 60 users to validate the performance of our Sleep Well framework. We also compare our results for data from different body position.

3.9.3.1 Dataset with Clinical Ground Truth

We evaluate our model using publicly available benchmark dataset from Technische Universität Darmstadt [126] which provides sleep phases determined by clinical polysomnography. The data set consists of timestamped raw acceleration data collected using wrist worn data logger at a sampling rate of $100Hz$ and includes the sleep stages (movements, awake, NoREM 1-3, REM, unknown) from 42 lab patients. The trend of raw accelerometer reading in this dataset is shown in Fig. 3.5. There

are seven different classes in this dataset among which majority of the data points are labeled as unknown (51%) and awake (24%) with only a few important data points which affect the classification model. After inspecting the dataset, we note that the value of different data points of different classes were very close which imposes bias in our classification model. We handle this bias by assigning less weight to abundant data points (unknown and awake) and improve the classification process and accuracy.

3.9.3.2 Actigraph and Chronos Dataset

We collected sleep data using wActiSleep-BT and EZ430-Chronos at a sampling rate of $60Hz$ from 17 participants for two weeks. Out of 17 participants (11 males and 6 females) 13 were graduate students, 3 working professionals and 1 unemployed person. We conducted a survey beforehand to know about their sleep routine. Using the survey we gathered information regarding sleep time, average sleep hour, movement frequency in a scale of 1-5 and existing sleep disorders. We then selected a set of participants with diverse sleep routine and disorders. We asked the participants to put on the sensor when they go to the bed. The participants were also instructed to maintain a log the timing of getting up and getting in bed. The participants put the sensors on their waist using a belt. We noticed that wActiSleep-BT device has better sensitivity due to slight movements rather than EZ430-Chronos which help differentiate between actual movements and sleep patterns from a user. Almost 65% data points of this dataset belong to *Sleep* class

and 22% to *Awake* class. As a consequence, our dataset is also imbalanced. Fig. 3.6 shows the raw readings from ActiSleep device.

3.9.4 Evaluation Methodology

3.9.4.1 Supervised Learning

We carried out our experiments with 17 participants (11 males and 6 females) over two weeks where each participant has provided data for 8-10 days. We validated that 17 is a staistically significant population size using *t-test*. The trend in training accuracy (intra user) with respect to size of the population is shown in figure 3.13. We see that after increasing the population size more than 17, the accuracy is not changing significantly. We proved our hypothesis that it did not happen by a chance by conducting *t-test* using dataset 3.9.3.1. We conducted *t-test* with varying population size and received p-value of 2.021 with 95% confidence interval. Although we get a bit more accuracy if we increase the population size (74%), however due to training time and resource consumption we chose 17 as the optimum population size. Out of these 17 participants, 7 wore the EZ430-Chronos device and other 10 put on wActiSleep-BT. We split each data set into two parts, one for training and other for testing. To overcome the class imbalance problem, the importance weight play a significant role. We look at the confusion matrices of classifying two classes (Sleep, Awake) in Table 3.2 & 3.3. It is evident that due to class imbalance, a lot of the Awake class instances are inferred as Sleep if importance weight is not used. We applied our classification models on the datasets mentioned in sections 3.9.3.1

and 3.9.3.2.

	Sleep	Awake
Sleep	91%	9%
Awake	14%	86%

Table 3.2: Confusion matrix of Sleep/Awake classifier with importance weight.

	Sleep	Awake
Sleep	80%	20%
Awake	51%	49%

Table 3.3: Confusion matrix of Sleep/Awake classifier without importance weight.

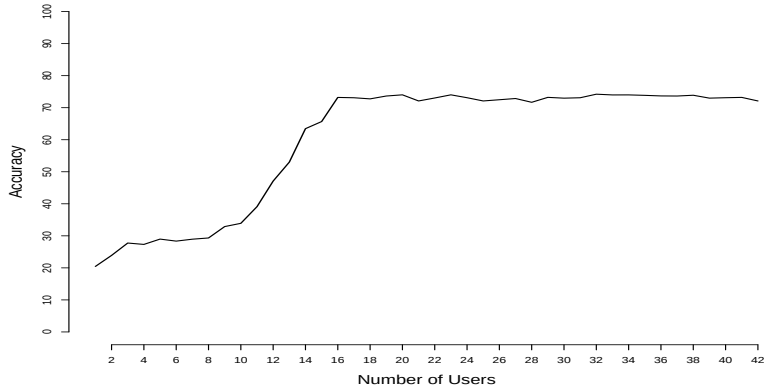


Figure 3.13: The trend in intra user classification accuracy with varying population size.

3.9.4.1.1 Intra User Classification

We tested different classification models with our proposed Online Stochastic Gradient Descent (OSGD) method - Support Vector Machine (SVM), Multilayer Perceptron (MP), LogitBoost (LB), Random Forest (RF), Logistic Regression (LR), and Decision Tree (DT) - using different user's dataset. The accuracy of different classification model using one of the subject's dataset from each datasets is shown

Table 3.4: Accuracy (Dataset : 3.9.3.1)(%) (Intra User)

	OSGD	SVM	MP	RF	LR	LB	DT
Unknown	98.76	96.50	85.80	99.01	98.93	98.37	97.95
Stage-1	69.45	44.44	60.36	58.57	47.63	61.82	70.40
Stage-2	70.29	41.02	58.54	59.09	48.18	68.46	71.87
Stage-3	68.36	39.15	63.36	48.24	49.33	60.77	63.94
REM	58.22	37.28	49.11	41.55	38.31	41.03	59.10
Awake	74.78	68.12	72.10	70.01	64.96	66.90	72.73
Movement	72.59	69.31	62.88	70.66	65.60	63.27	71.25
Average	73.20	56.54	66.13	63.87	58.99	65.80	72.46

Table 3.5: Accuracy (Dataset : 3.9.3.2)(%) (Intra User)

	OSGD	SVM	MP	RF	LR	LB	DT
Sleep	87.79	87.98	80.8	84.01	73.32	76.63	88.52
Awake	77.9	71.35	75.66	67.91	73.56	70.21	75.69
Movement	76.25	74.87	68.32	68.41	70.27	72.11	75.14
Getting up & sitting	72.11	64.58	67.39	68.11	64.85	69.15	70.02
Getting up from bed	78.21	69.89	70.36	70.1	71.19	62.39	73.39
Average	78.45	73.73	72.50	71.70	70.63	70.09	76.54

in Table 3.4 and 3.5. The average accuracy of OSGD is 73.20% for a patient from dataset 3.9.3.1 and 78.45% for dataset 3.9.3.2. This attests that consideration of inclination and sensor data and using it to correct labels in dataset 3.9.3.2 help yield better classification results. Also the results indicate that putting the device on the waist endows better accuracy. We investigated this disparity, and found that hand movements are more abrupt and arbitrary which results in more confusing data points. Also very subtle body movements are difficult to distinguish when using a wrist worn accelerometer.

The major accuracy improvement was noticed for inferring the micro sleep state. Although individual accuracy for classes - *Stage 1*, *Stage 2* and *REM* for Decision Tree (DT) classifier was better in dataset 3.9.3.1, but the average accuracy of inferring sleep states (sleep stage 1-3, REM) is 66.58% which is better than the average of DT classifier (66.32%). While for our dataset we achieved 87.79% accuracy.

3.9.4.1.2 Cross User Classification

It is important that a classification process will not only recognize the sleep states of an already seen user, but also help generalize the classification for new users. We cross validated our approach with inter user classification model. We trained our model using 20 patient's data from dataset 3.9.3.1 and tested the trained model with remaining 22 patient's data. The average accuracy was 69.79%. With data from dataset 3.9.3.2 we achieved 75.46% overall accuracy. Fig. 3.11 and 3.12 shows the results in Precision (percentage of times that a recognition result made by the

model is correct), Recall (percentage of times that a sleep state is detected) and F1-measurement (combination of both recall and precision) for both the datasets. Also in Fig. 3.14 represents the trend of loss for different datasets.

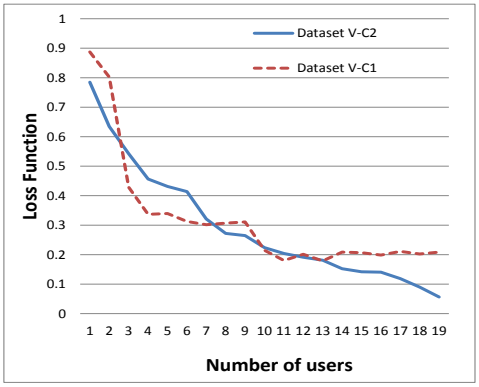


Figure 3.14: Trend of loss for inter user classification in different datasets

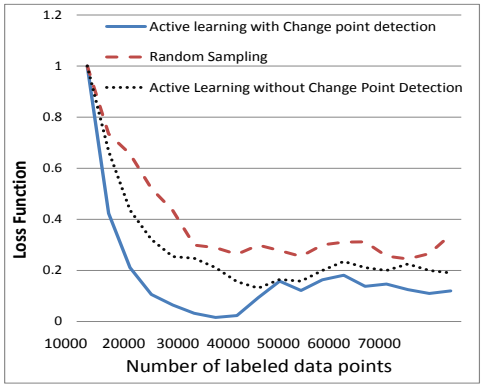


Figure 3.15: Different active learning techniques for Dataset 3.9.3.1.

3.9.5 Active Learning Experiments

In addition to supervised learning, we evaluate how we can improve the classification result using active learning with minimal user feedback. We have discussed our active learning algorithm in section 3.7. We sampled both the datasets with a window of 60 seconds on accelerometer data. Each sample is a feature vector with 16 dimensions. Initial labeled dataset L_1 consisting of 135089 samples (from dataset 3.9.3.1) and L_2 consisting of 42,000 samples (from dataset 3.9.3.2) are provided to the individual classifier C_1 and C_2 for training. Then unlabeled dataset U_1 of 510113 (dataset 3.9.3.1) and U_2 of 121,147 samples (dataset 3.9.3.2) are used to test the classifier C_1 and C_2. The samples are provided sequentially with respect to

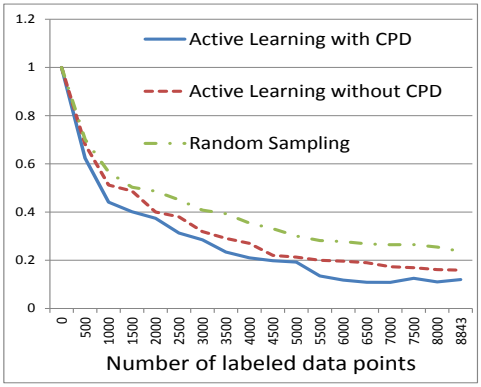


Figure 3.16: Different active learning techniques for Dataset 3.9.3.2.

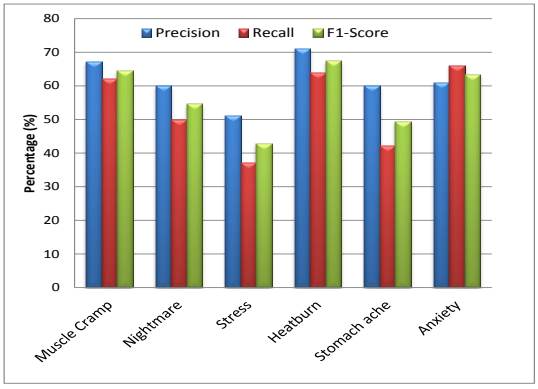


Figure 3.17: Different attributes classification result in getting up & sitting partition.

timestamp.

The uncertain data points, meaning the points which the classifier was unable to classify are queried in accordance with our active learning algorithm (3). We calculated the loss at each phase after a data point is queried and the model is retrained. We compared our result with randomly selected samples for labeling. To further assist the active learning process we validated the results with our change point detection (CPD algorithm discussed in 3.6. When a change point is detected in the dataset, we cross validated the change points with the classification result with respect to the timestamp. Figs. 3.7 and 3.8 plot the association of change points with timestamped accelerometer data points. If the label of the sample is not consistent between each of the model we imposed active learning and queried the data point. Initially with L_1 and L_2 we note the average classification accuracy as 63.8% and 70%. We applied importance weighted active learning, and see that the

model converged faster with change point detection. 86719 samples (17% of total samples) from U_1 and 8843 samples (7.3% of total samples) from U_2 were queried for the model to converge in presence of CPD which helped achieve 72% (dataset 3.9.3.1) and 76.89% (dataset 3.9.3.2) accuracy, while with randomly selected data points 68% and 73% accuracy was observed. Fig. 3.15 and 3.16 shows the change in loss with random sampling, active learning with and without CPD techniques with different datasets. We see that active learning with CPD outperforms the other strategies. In case of dataset 3.9.3.1 we notice from Fig. 3.15 that the change in loss is irregular. After analyzing the dataset 3.9.3.1 we found out that due to the presence of noisy data points the loss increased.

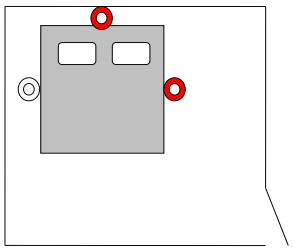


Figure 3.18: Visual illustration of sensor activation.

Class	**Kappa**	**Z-Score**	**P-Value**
Sleep	0.468	2.8	0.167
Awake	0.423	5.97	0.013
Movement	0.447	4.52	0.00678
Getting up & Sitting	0.537	2.298	0.051
Getting up from the bed	0.835	1.256	0.0006

Table 3.6: Fleiss Kappa Score (Inter user agreement).

3.9.6 Crowdsourcing Experiments

During this process we faced a challenge regarding what kinds of data to show which can reflect the sleep classes. As audio, video or image data violate the privacy of the user so we had to come up with a different methodology rather than traditional image based crowd-sourcing. We presented some semantic information from the users sleeping habit (regular hours of sleep, sleep latency, posture, average number of times the user gets up at night, how much the user moves on average in percentage and light condition) and a visual illustration of sensor activation (discussed in section 3.9.2) to the annotators. In Fig. 3.18, we show an example of visual illustration. The double circled objects represent sensors and the activation is marked by red color. In this example the sensor mounted near the head and the sensor mounted near the right side of the bed are activated as the user was getting up from the bed. 10 annotators participated in our crowdsourcing experiment with a dataset containing 10,000 data instances from Dataset 3.9.3.2. In table 3.6 the kappa coefficient, z-score and p-value for individual classes are shown. The kappa coefficients of sleep, awake and movement classes are considerably low than other two classes. This was due to the nature of these activities and sensitivity of the motion sensor. During our experiment we have seen that, movement with smaller intensities while in sleep or awake is sometimes not captured by the motion sensor. As a result most of the annotators, defined those data instances as sleep. On the other hand, movement with higher intensities while sleeping are annotated as awake, movement and getting up & sitting. Also the first three activities are related to the state lying. As a result

the posture information using inclination measurements did not help much. For getting up from bed the visual illustrations were less noisy and easier to depict, as a result the inter user agreement score is much better.

3.9.7 Introduction of New Unseen Class and Attributes

A user is able to personalize the model by introducing new unseen classes and attributes with the help of active learning. We simulated our active learning algorithm by introducing new class labels in the classification model. While collecting the query label we also asked for the reason behind choosing the label from the annotator, so that we can look for important indicators for the clinicians. We restricted the length of the reason in 5 words. For example, if a sample is queried and the annotator labels the sample as *getting up and sitting*, he can also state the reason for labeling the data such as *muscle cramp, stress or anxiety, nightmare* etc., which are microscopic events for sleep disruption. We applied a nested classification by considering these microscopic events as class labels. After classifying using our defined general class labels, we partitioned each class label data and applied our classification algorithm in separate partitions again by considering the provided attributes as labels. For example, let us assume an user states reason 'A' as the cause of sleep disruption or any kinds of changes in the pattern. Our framework then partitions the data and the number of partition is equal to the number of class labels (in our model it is 5), as a result in each partition the data points are of same class. Sleep Well framework then performs a classification on separate partitions with class label

'A'. This nested classification process ascertain the microscopic sleep events. The precision, recall and F1 score of recorded attributes (muscle cramp, heatburn, stomach ache, stress, anxiety, and nightmare) for parent class *"getting up and sitting"* is presented in Fig. 3.17.

3.10 Conclusions

In this chapter, we described the design, implementation and evaluation of Sleep Well, a sleep monitoring framework which helps classify the microscopic sleep states using wearable devices. We postulated a gradient descent-based approach which incorporates with importance weight aware updates in the microscopic sleep state detection process. We also consolidated our framework by blending change point detection and active learning in the inference pipeline. Our classification achieved 78% accuracy with the aforementioned experimental setup. The empirical results demonstrate the effectiveness of our framework in determining different sleep states. The result increased by 7% when active learning was employed. Our approach helps accelerate the faster convergence to optimal sleep states detection accuracy using minimal user feedback in presence of active learning. Besides that with the help of change point detection, we were able to validate and interpret the transitions between these sleep states. In future, we plan to investigate the combination of change point detection and classification to further improve the accuracy. Also by conforming the attributes from user provided feedback into our architecture will help provide meaningful insights for better understanding of sleeping behavior.

Chapter 4: Active Activity Recognition

4.1 Active Learning Enabled Activity Recognition

Due to a large variety in number of Activities of Daily Living (ADLs), acknowledging them in a supervised way is a non-trivial research problem. Most of the previous researches have referenced a subset of ADLs and to initialize their model, they acquire a vast amount of informative labeled training data. On the other hand to collect ground truth and differentiate ADLs, human intervention is indispensable. As a result it takes an immense effort and raises privacy concerns to collect a reasonable amount of labeled data. In the previous section we validated that active learning can help to acquire labeled information. In this section, we propose to use active learning to alleviate the labeling effort and ground truth data collection in a more generalized settings. We investigate and analyze different active learning strategies to scale activity recognition and propose a dynamic k-means clustering based active learning approach. Experimental results on real data traces from a retirement community help validate the early promise of our approach.

4.2 Contributions

Most of the proposed active learning model focused on finding the most informative point using uncertainty measurement or maximizing error reduction. For some classifiers a basic intuition is followed - *any data instance closest to the decision boundary is considered important and thus queried.* Some approaches are strictly for SVM classifiers where the intention is to identify the instances from feature space which will maximize the hyperplane margin. These approaches can yield good results and improve the overall performance, however they tend to ignore the prior distribution of the feature space. Prior distribution can be useful for active learning and ignoring them can create sampling bias in the overall system. In this work we propose a cluster based active learning model for activity recognition. Combining clustering with active learning has been proposed in [93] [102] [170]. But postulating them for practical human-in-the-loop applications are still in its infancy. The effectiveness of active learning is significantly dependent on the quality of the labels acquired from prompting queries. When multiple annotators are involved, the probability of introducing noisy inputs increases. On the other hand some of the annotators might be reluctant to provide labels to the queries. In these cases the throughput of active learning is largely hampered. It is necessary to model an active learning algorithm which can handle noisy and reluctant annotators.

In our proposed model, we first create hard clusters without explicitly differentiating the number of clusters, rather focusing the minimum number of clusters in association with the existing number of labels in the label space. We posit the

most informative instances in a cluster and, subsequently acquire label for them using our novel objective function and finally reinstate the cluster label through empirical validation. We also model an annotator selection model to capitalize the efficient annotators so that the classifier model remains stable. In summary our proposed active learning enabled activity recognition model contains following salient contributions:

- We propose a dynamic k-means clustering algorithm for creating clusters of unlabeled data.
- We propose an objective function to find the most informative data instance in the cluster.
- We propose an annotator selection model when multiple labelers with varying expertise are present.
- We propose to include an unsure option in our model to provide freedom to annotators in case of complex data instances.
- We model our active learning algorithm when no ground truth information is available.
- We validate our model using real life data traces and compare our model with other viable active learning approaches such as disagreement based approach.
- We present a data representation technique for crowdsourcing smart home data and discuss its implication on active learning assisted activity recognition.

4.3 Overall Design & Challenges

Our proposed model is comprised of two significant steps. First formulate clusters for the unlabeled data instance pool (U) using K-Means clustering. The fundamental motivation behind using clustering over other active learning strategies is *computational complexity*. Though uncertainty sampling based strategies are computationally inexpensive, they tend to be biased and become over confident. Other popular disagreement based strategies like Query By Committee (QBC) uses the hypothesis space to form the committee, and maintaining a hypothesis space demands a lot of computation. For a practical and real life system, we need an efficient strategy with low computational complexity. Clustering based strategies reduces the overall complexity of the active learning algorithm. Dasgupta et al. [102] used hierarchical clustering which becomes computationally expensive for large data sets. K-Means clustering is a widely used partition clustering method and if employed with proper heuristic, K-means can achieve linear time complexity. The clustering process follows a simple and easy way to classify a given data set through a certain number of clusters (let us assume k clusters) fixed beforehand. The idea is to identify the center of the clusters - centroids, and then associate the data instances to the closest centroid and formulate clusters. However the computational complexity becomes NP Hard. One other challenge for applying K-Means clustering is that we have to postulate the number of clusters explicitly. For building an adaptive and spontaneous activity recognition model to discover unseen activities, it is not practical to posit the number of clusters beforehand. We iteratively run the clustering

until we find a stable set of clusters with minimum clustering error. We decrease and control the computational complexity of K-Means clustering by using Elkan's heuristic [171]. We apply a pool based sampling strategy for constructing instance pool (U) as it has been used in practice [172] [173] [174]. We delve into the practical application of active learning and particularity for smart home activity recognition where a multitude of ambient and wearable sensor data streams are abundant. Thus we design a sampling approach based on a pool of unlabeled data instances and then pass it to our active learner.

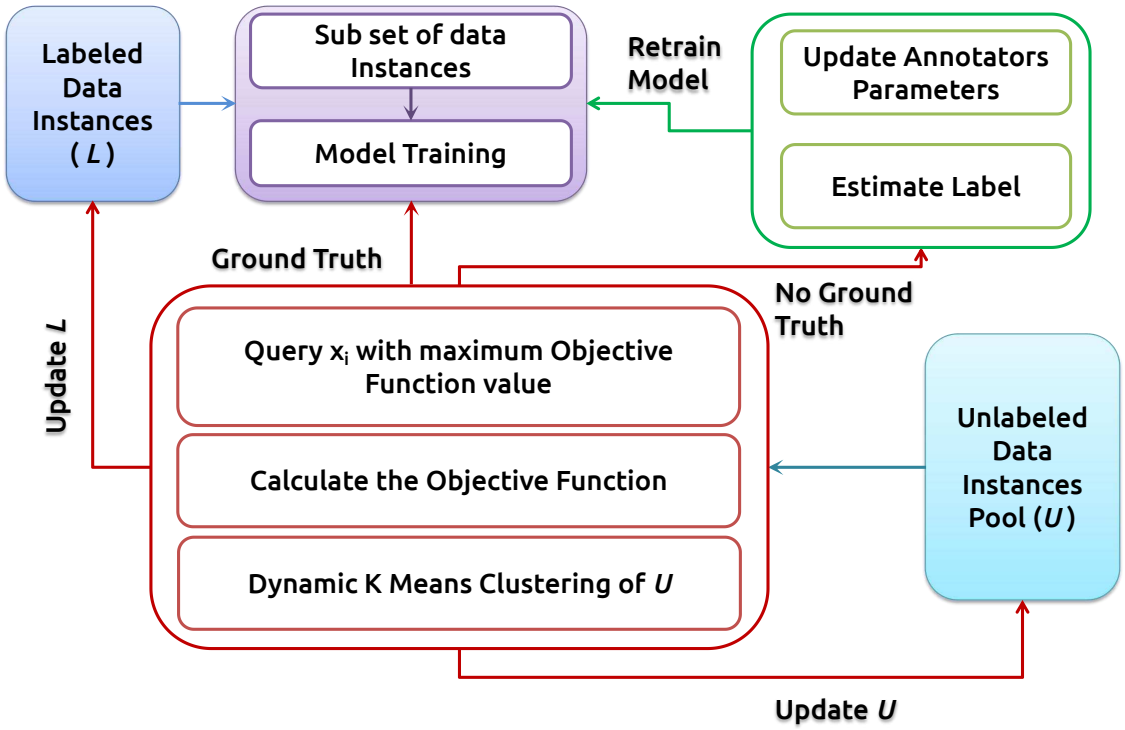


Figure 4.1: Overall framework for active learning inspired activity recognition.

After compiling the clusters out of unlabeled data instances pool, we find out the most informative data instances and query them accordingly. We also incorporate the clusters with unseen activities in this underlying process. In order to find

the most informative data instances, we formulate an objective function which is constructed using entropy measurement and a similarity coefficient. The similarity coefficient for any data instance is calculated by the distance measurement between the points in its cluster and other surrounding cluster centers. Fig. 1 depicts this entire active learning enabled activity recognition model.

4.4 Background

Given n points $\{x_1, x_2....., x_n\} \in \mathbb{R}_d$ the goal of K-means is to find K cluster centers $\{c_1, c_2...., c_m\} \in \mathbb{R}_d$ and assignment $\{q_1, \ldots, q_n\}$ of the points to the centers. K-Means clustering tries to find the position of the cluster centers and minimize the distance of the data instances in Eqn 4.1. K-means is obtained for the case p=2 (l^2 norm), because in this case the optimal centers are the means of the input vectors assigned to them.

$$E(c_1, \ldots, c_k, q_1, \ldots, q_n) = \sum_{i=1}^{n} \|x_i - c_{q_i}\| \tag{4.1}$$

Minimizing the objective E is in general a difficult combinatorial problem, so locally optimal or approximated solutions are sought instead. E is also the average reconstruction error, if the original points are approximated with the cluster centers. Thus K-means is used not only to group the input points into cluster, but also to quantize their values. The basic K-means algorithm alternate between re-estimating the centers and the assignments. Combined with a good initialization strategy and potentially, by re-running the optimization from a number of randomized starting states, this algorithm helps reduce the complexity of handling expo-

nential state-space of active learning and increase efficiency in practice. However, despite its simplicity, simple K-means is often too slow. Thus we consider Elkan's algorithm [171], which uses the triangular inequality to cut down significantly the cost of basic K-means.

4.5 Cluster Heuristics

Elkan's algorithm [171] is different than Lloyd alternate optimization algorithm (Lloyd's algorithm) which calculates the triangular inequality to mitigate many distance calculations when assigning data instances to clusters. Although this heuristic is much faster than Lloyd, but it needs storage proportional to the number of clusters. which makes it difficult to operate in case of large number of clusters. The base of this algorithm is that, if a centroid update does not move data instances much, then most of the instance to center distance computations can be avoided when the assignments are recomputed. To distinguish which distances need evaluation, the algorithm bounds the distances by lower and upper bound using triangular inequality after a center update. Elkan algorithms uses two key observations. First, one has

$$\|x_i - c_{q_i}\| \leq \|c - c_{q_i}\|/2 \quad \Rightarrow \quad \|x_i - c_{q_i}\| \leq \|x_i - c\| \tag{4.2}$$

Thus if the distance between x_i and its current center c_{q_i} is less than half the distance of the center c_{q_i} to another center c, then c can be skipped when the new assignment for x_i is searched. Checking this requires keeping track of all the inter-center distances, but centers are typically a small fraction of the training data, so

overall this can be a significant saving. In particular, if this condition is satisfied for all the centers $c \neq c_{q_i}$, the point x_i can be skipped completely. Furthermore, the condition can be tested also based on an upper bound $U(x)$ of $\|x_i - c_{q_i}\|$. Second, if a center c is updated to $\hat{c}$, then the new distance from x to $\hat{c}$ is bounded from below and above by

$$\|x - c\| - \|c - \hat{c}\| \leq \|x - \hat{c}\| \leq \|x - \hat{c}\| + \|c + \hat{c}\|. \tag{4.3}$$

This allows to maintain an upper bound on the distance of x_i to its current center c_{q_i} and a lower bound to any other center c.

$$U(x) \leftarrow U(x) + \|c_{q_i} - \hat{c}_{q_i}\| \tag{4.4}$$

$$L(x, c) \leftarrow L(x, c) - \|c - \hat{c}\|. \tag{4.5}$$

Traditional K-means clustering needs prior definition of number of clusters. But if data instances of unseen classes are present in the data pool, they will be considered as outliers in the clustering process. Having outliers in the feature space can significantly decrease the performance of the clustering algorithm if the number of clusters are not properly defined. From these motivations we choose to apply an incremental K-means clustering where at each iteration we increase K and record the overall error for our clustering using the error function in Eqn. 4.6. The minimum error depicts the best clustering for the data set and the corresponding number of k cluster.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x^{(j)} - c_j\|^2 \tag{4.6}$$

4.6 Measuring Informativeness

After clustering the unlabeled data instances, our task is to figure out the most informative data instance in different clusters. [102] proposed random sampling for labeling the data. Instead of using random sampling we propose to use uncertainty sampling to choose the most informative instance. One challenge is to differentiate between outliers and most informative data instances as outliers will have much higher uncertainty like the most informative instances. To represent whether a data instance with high uncertainty is an outlier or not, we calculate the silhouette coefficient [175], $S_c^{(x_i)}$ for each data instances. For each data instance x_i, let $a(i)$ be the average dissimilarity of x_i with all other data within the same cluster. $a(i)$ interprets how well a data instance x_i is assigned to its own cluster. Where smaller value of $a(i)$ indicates better assignment of x_i. The average dissimilarity of point x_i to a cluster c is defined as the average of the distance from x_i to points in c. Let $b(i)$ be the lowest average dissimilarity of x_i to any other cluster, of which x_i is not a member. The cluster with this lowest average dissimilarity is said to be the "neighbouring cluster" of x_i because it is the next best fit cluster for point x_i. We now define a silhouette:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{4.7}$$

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, & \text{if } a(i) > b(i) \end{cases} \tag{4.8}$$

From Eqn. 4.8 it is evident that $-1 < s(i) < 1$. For $s(i)$ to be close to 1, we require $a(i) \ll b(i)$. As $a(i)$ is a measure of how dissimilar x_i is to its own cluster, a small value means it is well matched. Furthermore, a large $b(i)$ implies that x_i is poorly matched to its neighbouring cluster. Thus an $s(i)$ close to 1 means that the instance is appropriately clustered. If $s(i)$ is close to negative one, then by the same logic we note that x_i would be more appropriate if it was clustered in its neighbouring cluster. An $s(i)$ near zero means that the instance is on the border of two natural clusters. The informativeness is measured by the entropy of the instances. The entropy of an instance is defined by the following equation:

$$\begin{aligned} e_\theta(x) &= \underset{x}{\operatorname{argmax}}\, H_\theta(y|x) \\ &= \underset{x}{\operatorname{argmax}}(-\sum_y P_\theta(y|x) \log P_\theta(y|x)) \end{aligned} \tag{4.9}$$

We combine this entropy measurement with the $S_c^{(x_i)}$ to filter out the most informative points. So the final objective function f_c for finding the most informative point becomes

$$f_c(x) = \underset{x}{\operatorname{argmax}}\{e_\theta(x)\,.\,S_c^{(x_i)}\} \tag{4.10}$$

By using Eqn. 4.10, the points which are properly clustered and has higher entropy values are chosen. The points which has $S_c^{(x_i)} = 0$ or negative values might also be important. So instead of discarding those data instances, we randomly pick instances and query them. An instance with $S_c^{(x_i)}$ closer to zero is close to the boundary of two different clusters. If we have the same label as other data instances in that cluster, we do not have to change the cluster, but if different label is received

we have to rearrange our cluster with respect to the received label. In Algorithm 4 we demonstrate the full active learner with K-means clustering.

4.7 Label Complexity

In order to identify the effectiveness of an active learning algorithm we have to define *Label Complexity*. Label Complexity denotes the number of queries required to train a approximately correct classifier while employing active learning. So we are interested in finding the upper bound on the number of queries required. The label complexity can be much smaller than the sample learning complexity of a passive learning algorithm. Let L^c be the label complexity of an active learning algorithm defined as $P(err(h_t) \leq \epsilon) \geq 1 - \delta$, where h_t is the output hypothesis after t-th iteration and err is the error rate of h_t. So label complexity L^c is the number of labeled instances required to train a classifier with error rate less than ϵ.

During each iteration in our clustering algorithm $k(k-1)/2$ pairwise distances between all centers are recomputed (Algorithm 4, line 4). Recomputing the centroids using Elkan's heuristic (Section 4.5) requires additional $O(N)$ distance calculations. To update the lower and upper bound K distances between current and new centroids must be calculated which gives us $O(N + K^2)$ distance calculations at each iteration. As the centroids calculation become close to convergence the distance calculation becomes less common. For fixed K this is our general bound, but our dynamic clustering algorithm iterate while adjusting the value of K. So for I iterations the true general bound for our clustering algorithm will be $O(N + K^2)^I$. After

Algorithm 4 Active Learner with K-Means Clustering

Input: U = A pool of unlabeled instances $\{(x)^u\}_{u=1}^{U}$, min_k = Minimum number of clusters, θ = An error threshold.

Output: Clustered classification of U and most informative data instances in each cluster.

for $K = min_k$ **do**

    while $J_{current} > \theta \,||\, K = min_k$ **do**

        for Each $x_i \in U$ **do**

            if x_i is a candidate centroid **then**

                Compute $L(x^i, c) - \|c - \hat{c}\|$

                Find the current assignment q_i and bounds $U(x^i)$ by finding the closest centers to each point

            end if

        end for

        Estimate Center and Quantize the data instances in U to associate them with a cluster c.

        Compute the Error $J_{current}$ using Eqn. 4.6.

        $K = K + 1$

    end while

end for

for every $x_i \in U$ **do**

    Calculate $S_c^{(x_i)}$ using eqn 4.8.

    Calculate the objective function $f_c(x)$ using Eqn. 4.10.

    Choose the instance with highest $f_c(x)$ and query for label.

end for

calculating our objective function $f_c(x)$, the bound becomes $\{O(N)+O(N+K^2)^I\}$. Then the true label complexity of our active learning algorithm depends on the number of cluster k as

$$O((\frac{k}{\epsilon})\ \log\ \frac{1}{\epsilon}+\frac{1}{\epsilon}\ \log\ (\frac{1}{\delta}))$$

4.8 Selecting Annotator

We filtered out the most informative data instances in the previous step, but it is quite impractical to presume in a real life environment that the annotators are going to provide true and correct labels all the time. As the search for most informative data instances is important so is getting the true label for them, otherwise unnecessary noise draws on our model. In order to build practical active learning enabled activity recognition model, choosing the right annotator is crucial and it becomes more challenging when the environment is cohabited by multiple inhabitants. As all annotators are not expert in annotating a specific activity class, we assign a sensitivity score to each annotator for each activity class. The sensitivity is defined by the number of labels correctly labeled by an individual annotator.

$$\mu_{sen}^{c} = \frac{\text{correctly labeled instances of class c}}{\text{number of points queried of class c}} \tag{4.11}$$

Initially when an annotator has not labeled any examples from class c_i, we assign a sensitivity score of $\mu_{sen}^{c_i} = 0.5$. Using the sensitivity score, we pinpoint the most efficient annotator. To further strengthen the querying process, we choose the an-

notator who has annotated a similar data instance with high sensitivity which is annotator's specificity δ_{spec}. For this we first monitor j neighboring data instances who are closer to x. For our experimental purpose we chose $j = 4$.

$$\delta_{spec_i} = \frac{\frac{1}{k} \times \sum_{i=1}^{j} \mu_{x_i}^{a_k}}{1 + \frac{1}{k} \times \sum_{i=1}^{j} |x - x_i|} \tag{4.12}$$

In Eqn. 4.12, $\mu_{x_i}^{a_k}$ is our sensitivity of annotator, a_k. $|x - x_i|$ is the euclidean distance between the neighboring data instances. Our active learner is set to select the annotator by the following Equation 4.13.

$$i = \text{argmax}(\delta_{spec_1}, \delta_{spec_2} \ldots, \delta_{spec_k}) \tag{4.13}$$

After selecting a subset of annotators we ranked accordingly based on their specificity score and query the data instance to top 4 annotators. If the confidence of the data instance is less than 80% then we keep on querying until it reaches confidence level of 80%. Here the confidence level for a data instance means the ratio of number of labels and total number of query. If x_i receives 6 labels of class c_j out of total 10 queries, then the confidence score is $100 \times \frac{6}{10} = 60\%$.

4.8.1 Unsure Option

We want to grant freedom to the annotators while they provide labels to the queries. It is possible that the provided scenario or setting may be confusing to the annotator. So we will provide an *unsure* option to tackle such scenarios. This will also help to relax the effort needed from the annotators for difficult data instances. Instead of providing random labels, the unwilling annotators are able to provide

unsure label which help noise reduction. Let us denote C as our target classifier and after i iterations it transforms into C_i. As $f_c(x)$ is our objective function, the queried instance picked in the $(i+1)$-th iteration is-

$$x_{i+1} = \underset{x \in U_i}{\operatorname{argmax}} f_c(x|L_i, C_i, S_{c_i}^{(x)}) \tag{4.14}$$

Here U_i is the current set of unlabeled instances, L_i is the current labeled data and $S_{c_i}^{(x)}$ is the silhouette coefficient of x. We have defined how specific an annotator (δ_{spec_i}) is in section 4.8. Now we define another parameter to represent the overall specificity of the all the existing annotators $\omega(x)$ which is a function of δ_{spec_i}. Now if we can rewrite the equation 4.14 as -

$$x_{i+1} = \underset{x \in U_i}{\operatorname{argmax}} f_c(x|L_i, C_i, S_{c_i}^{(x)}) \,.\, \omega(x) \tag{4.15}$$

We have to define $\omega(x)$ with respect to δ_{spec_i}. For unsure option we assign $x = 0$ and $x = 1$ otherwise. In Eqn 4.16 we define our $\omega(x)$ as a disjunction function where $x_1 \vee x_2 = 0$ if and only if $x_1 = 0$ and $x_2 = 0$ otherwise $x_1 \vee x_2 = 1$. Also we can see that $0 < \omega(x) < 1$.

$$\omega(x) = \vee_{t \in T} \, \gamma(\delta_{spec_{i(x)}}) \tag{4.16}$$

The specificity model of the annotators may not be definite in the early stages of learning, therefore the most informative instances are unlikely to be queried event if annotators who can provide correct label exists. As our primary target for our

dynamic K-means clustering is to minimize the error function J (Eqn 4.6), we rewrite Eqn 4.14 and 4.15 as -

$$x_{i+1} = \underset{x \in U_i}{\operatorname{argmin}} \left(\sum_{j=1}^{k} \sum_{i=1}^{n} \|x^{(j)} - c_j\|^2 \right) + M(1 - \omega(x)) \tag{4.17}$$

Here M is a large constant which confirm the instance x for which $\omega(x) < \theta$ would not be selected. θ is a threshold which we empirically define. We select our suitable annotator using Eqn 4.13. After getting the label for the selected instance, we re-organize our clusters and update the specificity of the annotator accordingly.

4.9 No Ground Truth

One of the major challenge in active learning is how to validate whether the annotators are providing correct labels or not when no ground truth information is available. This problem becomes more complicated as different annotators have different expertise. Although it is possible to disregard this problem by only relying on one annotator who will label his own activity data instances. The problem with single annotator model is - if the user is reluctant to provide labels to the queries, then active learning will be of no significant use. One of the other motivation for dealing with no ground truth information is when new unseen classes are introduced. New unseen classes have no prior data instances and their labels are nonexistent in the label space. As the expertise of individuals have an influence on the provided label, so it is necessary to devise a model to estimate the more probable label for data instances with no ground truth information.

4.9.1 Model Definition

Let us consider X to be our instance space with instances $\{x_1, x_2, ..., x_N\}$ and Y to be our label space that have been provided by our annotators. The existing labels in Y are observed labels whereas the true label space is Z. In ideal case $Y \equiv Z$. Let us define random variables $x \in X$ and $z \in Z$ for input data instances and true label respectively. Similarly we define random variables $y^{(t)}$ provided by labeler t over label space Y where $l = \{1, 2, 3...., L\}$. There is a possibility that Y is not fully observed as some $y_{(l)}$ will be missing because the annotator might not provide any label for the query. The probabilistic graphical model for our random variables are shown 4.2.

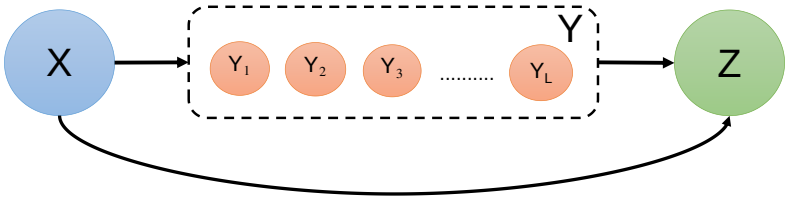


Figure 4.2: Graphical Model for input data instances X, Labels provided by L annotators Y and the true label space Z of X data instances.

We have defined our necessary random variables, now we formulate the conditional distribution as following

$$P(Y, Z|X) = \prod_i p(z_i|x_i) \prod_{l|l \in L_i} (p(y_i^{(l)}|x_i, z_i)) \tag{4.18}$$

In Eqn 4.18 L_i denotes the set of annotators who provided a label for i-th data instance. It is evident from the this equation that the label provided by the

annotator is influenced by the true label z and the observed data instance x. In order to calculate the conditional distribution we need to define $p(y_i^{(l)}|x_i, z_i)$ and $p(z_i|x_i)$. We define a Gaussian model (Eqn 4.19) to calculate these parameters as our input space X is a continuous domain in temporal and spatial domain. In our Gaussian model σ and ω are variance and the prior probability of j-th distribution respectively.

$$p(y_i^{(l)}|x_i, z_i) = \sum_{j=1}^{k} \omega_j N(y_i^{(l)}|z_i, \sigma_l(x_i)) \tag{4.19}$$

We consider that $p(y|x, z) \neq p(y|z)$ because the labels provided by the annotators not only depend on their expertise but also on the type of the presented data. Annotators will be more comfortable to label data instances which are more familiar to them. For example, some annotators are familiar with cooking activity more than others. We capture this input dependent variability by taking the variance with respect to x and specific to each annotator l. Since $y^{(l)}$ is a multinomial discrete random variable we take $\sigma_l(x_i)$ as a logistic function -

$$\sigma(x_i) = \frac{e^{\beta_0^{(c)} + x.\beta}}{\sum_c e^{\beta_0^{(c)} + x.\beta}} \tag{4.20}$$

It is possible to model our another parameter $p(z_i|x_i)$ according to any distribution. For simplicity we model $p(z_i|x_i)$ to be a multinomial logistic function as well.

4.9.2 Maximum Likelihood

We defined our model and now we need to estimate the logistic function . We can calculate the parameters of $\sigma(x)$ by maximizing the log-likelihood function. As $y^{(l)}$ can take different discrete class labels, we model $y^{(l)}$ as a poisson random variable. Then the likelihood function would be

$$p(y^{(l)}|x;\sigma(x)) = \prod_{i=1}^{n} \frac{n!}{(n-i)!i!}\, p(x_i)^{y_i}\, (1-p(x_i))^{n-y_i} \tag{4.21}$$

The log likelihood then becomes as Eqn 4.22.

$$\begin{aligned} l[\sigma(x)] &= \sum_{i=1}^{n} \{\log \frac{n!}{(n-i)!i!} \;+\; y_i \log p(x_i) \;+\; (n-y_i)\log(1-p(x_i))\} \\ &= \sum_{i=1}^{n} \{\log \frac{n!}{(n-i)!i!} \;+\; y_i (\log \frac{p(x_i)}{1-p(x_i)}) \;+\; n\log(1-p(x_i))\} \\ &= \sum_{i=1}^{n} \{\log \frac{n!}{(n-i)!i!} \;+\; y_i \sigma(x_i)(1+x_i) \;+\; n\log(1-p(x_i))\} \\ &= \sum_{i=1}^{n} \{\log \frac{n!}{(n-i)!i!} \;+\; y_i \sigma(x_i)(1+x_i) \;-\; \log(1+e^{\sigma(x_i)(1+x_i)})\} \end{aligned} \tag{4.22}$$

To maximize Eqn 4.22 we employ Expectation maximization algorithm.

4.9.2.0.1 E Step

In this step, we need to calculate the expectation of our log likelihood function. Conditional expectation of our log likelihood $logp(y^{(l)}, z|x;\sigma(x))$ is defined as

following where $\bar{\beta}_0$ and $\bar{\beta}$ are parameters for previous step.

$$
\begin{aligned}
&E[log\, p(y^{(l)}, z|x; \sigma_x)|\bar{\sigma}(x)] \\
&= \sum_y p(y^{(l)}|z, \bar{\sigma}(x))\ \log\ p(y^{(l)}, z|x; \sigma(x)) \\
&= \sum_y p_{\bar{\sigma}(x)}(y^{(l)}, z|x)\ \log\ p_{\sigma(x)}(y^{(l)}, z|x)
\end{aligned} \tag{4.23}
$$

4.9.2.0.2 M Step

In this step, we need to maximize Eqn 4.23 which is difficult to optimize because it contains the logarithm function of the sum. To solve this we take a partial derivative of 4.23 with respect to our parameter $\delta_l(x)$ and calculate the gradients $\frac{\partial x}{\partial \delta_l(x)}$. By taking the partial derivative equal to zero we solve the equation.

4.9.3 Estimating Ground Truth

We can estimate the ground truth even when new unseen class is introduced by the following probability distribution

$$p(z|y^l) = \int \prod_l p(y^l|z, x) p(z|x) dp(x) \tag{4.24}$$

As when new unseen class label is introduced in the label space we have no knowledge about the new sample's prior distribution $p(x)$. In such cases we can gain knowledge from previously seen data instances. We take a sample X from our existing input sample space $X = \{x_1, x_2....x_s\}$ and then we calculate the posterior probability by

$$p(z|y^l) \approx \frac{1}{S}\sum_{s=1}^{S} p(z|x_s) \prod_t p(y^{(l)}|z, x_s)$$

4.10 Experimental Results

In this section we validate our active learning algorithm for activity recognition and compare the outcome with other popular strategies. We set up a smart home environment with PIR motion sensors and object sensors on different household appliances. The PIR motion sensors were mounted on three different locations of a single bedroom apartment (bedroom, living room and kitchen). The object sensors were mounted on the broom, trashcan, laundry basket, dustpan and phone. The object sensors have built in compass and accelerometer which provided the usage and orientation of the objects. Two door sensors were placed on the apartment door and on the closet door. In our experiment we considered only single inhabitant environments. We had the following activities in our dataset - 1) brooming, 2) cooking, 3) doing laundry (washing), 4) taking out the trash (cleaning), 5) eating, 6) sleeping and 7) using land line telephone (talking).

We trained our passive learner with the first four activities and left the last three activities for our active learner to discover. We have used Decision Tree classifier (J48) as our passive learner. We extracted features from the ambient motion sensors which include the start and end of the sensor events, time span of the event within a k-event window, and count of events in the window. We collected data from 10 participants who reside in a retirement community. Each participant

provided around 24 hours of continuous sensor data. Average age of the participants was 85. We evaluated our cluster performance using normalized mutual information (NMI) using ground truth. Both the activity class label and clustering assignment are considered as random variables in NMI. It measures the mutual information between the two data instances, and normalizes it to a zero-to-one range. Let C be the data instance representing the cluster assignments of instances, and K be the random variable representing the class labels of the instances, the NMI is computed by the following equation:

$$NMI = \frac{2I(C;K)}{H(C) + H(K)} \tag{4.25}$$

Here $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between random variables X and Y. For our experimental validation, we focus on the following specificities for our experiments: 1) Correctly classified instances by our active learner 2) Instance selection time 3) Mean Absolute Error 4) Number of average queries per data instance for gaining confidence level of 80% 5) Annotator selection time 6) Impact for introducing new unseen activities.

Figure 4.3: Smarthome System Setup.

4.10.1 Smarthome System

We have used CloudEngines PogoPlug [176] as our central component which interfaces with the motion sensors and object sensor tags. We built a custom linux kernel for the PogoPlug and developed tools to communicate with the sensor tags. The PogoPlug works as a bridge between the Sensor Tags and a standard 802.11N network. Tests have indicated the device is stable in this capacity, and recovers from power loss and outages on the external networks without any issue. The PogoPlug is also successfully doing NAT translation in order to bridge additional Ethernet devices on to the network. We are streaming and storing the data in real time in our lab server. We have used Foscam IP camera for collecting ground truth. Due to bandwidth limitation it was difficult to stream the video remotely, so we recorded the video in SD card. Since video violates the privacy issue, each participant provided only two hours of video data. In these two hours the participants were asked to follow a script to perform several tasks related to our activity list. Rest of the 22 hours were not recorded which is our test data. Figure 4.3 demonstrates the components of our system setup.

4.10.2 Supervised Model

In figure 4.4 the precision, recall and F1 score for each activity is shown for our decision tree classifier. It is evident that the cooking activity has lower accuracy than other activities. As in our experiments, we have placed object sensors on various appliances and equipments, so it was possible to pin point an activity by tracking

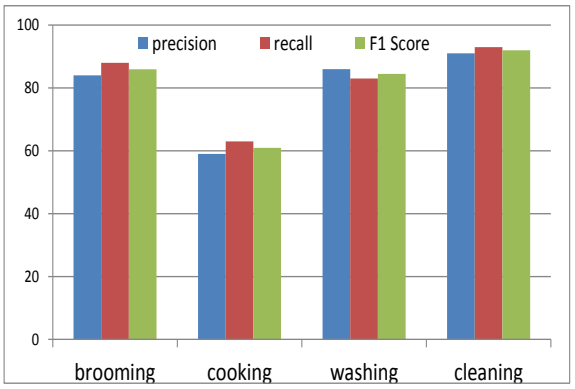


Figure 4.4: Precision, Recall and F1 Measurement of each activity for Passive Leaner.

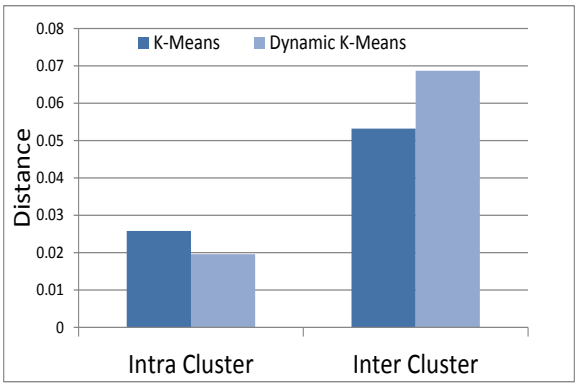


Figure 4.5: Comparison of performance using traditional k-means and dynamic k-means.

the usage of the object sensors. For example if we detected any movement for the laundry basket, we labeled the data instance of the motion sensor as cleaning. As for cooking activity there was no object sensor, so it was difficult to track down the activity properly. In some cases the objects were just moved or handled, not used for the corresponding activities which imposed noise in the dataset. For this reason we filtered the dataset by taking an assumption for the duration of each activity. If the duration of the performed activity was less than the threshold then we discard the data instance for training. The thresholds were defined in an empirical manner. Also some of the participants had pets in their apartment which introduced more noises in the dataset as the pets move around the house abruptly.

Algorithm 5 Query By Committee

1: **Input:** U = A pool of unlabeled instances $\{(x)^u\}_{u=1}^{U}$

2: L = A pool of unlabeled instances $\{(x)^l\}_{l=1}^{L}$

3: k = number of iterations

4: Repeat k times

5: Generate a Committee of Classifiers C^*

6: $\forall x^i \in U$, compute disagreement x^i_{VE} using 4.26 based on the current committee.

7: Select a subset S of instances from U that maximizes utility.

8: Query instances of S

9: Remove S from U

10: Update L by adding S

11: Return

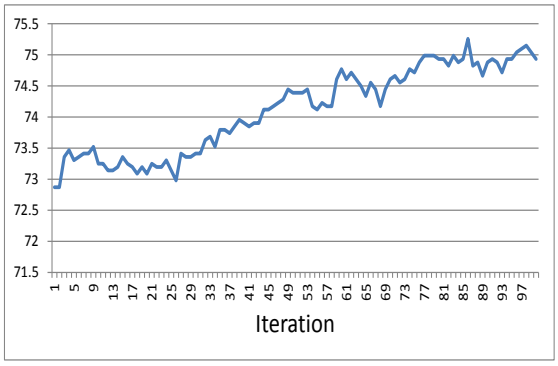


Figure 4.6: Correctly classified instances for our active learner.

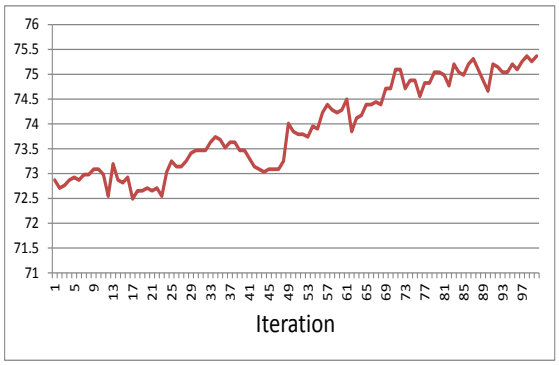


Figure 4.7: Correctly classified instances for maximum entropy sampling.

4.10.3 Active Learning Experiments

We have mentioned the criteria for our active learning experiments beforehand. We applied the active learning algorithm in a 10-fold cross validation and pool

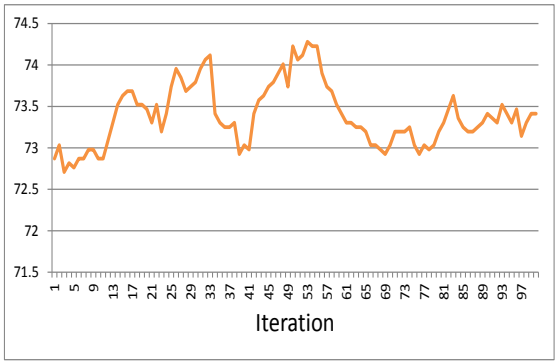


Figure 4.8: Correctly classified instances for least confidence sampling.

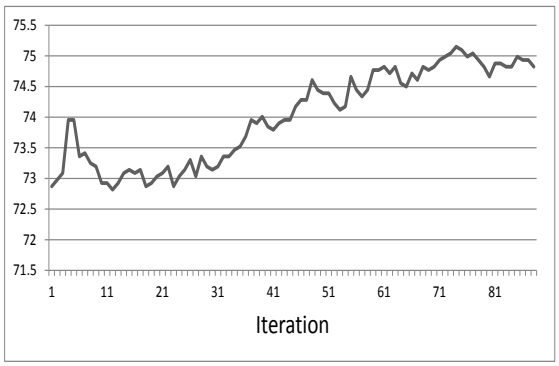


Figure 4.9: Correctly classified instances for Query By Committee.

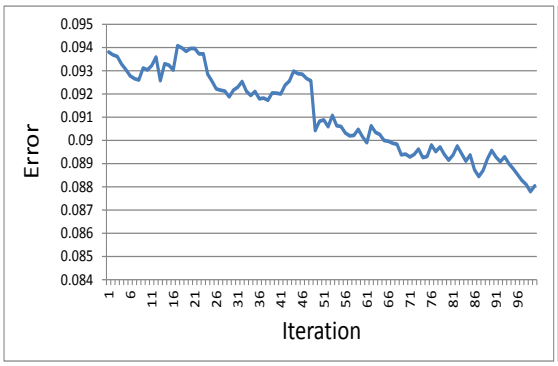


Figure 4.10: Mean absolute error for our active learner.

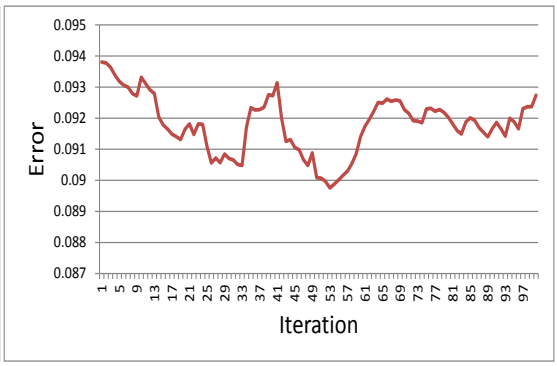


Figure 4.11: Mean absolute error for maximum entropy sampling.

based sampling manner. We initially started with a small labeled data set (5% of train data), and then made queries by using different active learning strategies. The results are shown for 100 iterations. We compare our algorithm with other popular query strategies - maximum entropy, least confidence and vote entropy or Query by Committee (QBC). QBC is a very effective alternative approach to uncertainty sampling which has been applied in many classification problems. QBC manipulates the version space and at each iteration it maintains a committee - an

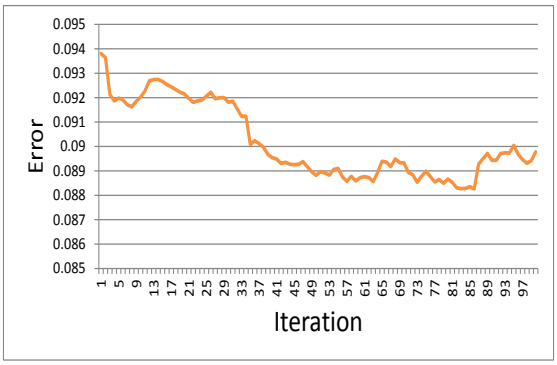


Figure 4.12: Mean absolute error for least confidence sampling.

Figure 4.13: Mean absolute error for Query By Committee (QBC).

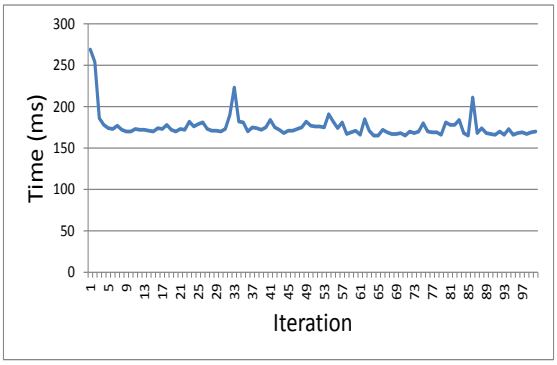


Figure 4.14: Instance selection time for our active learner.

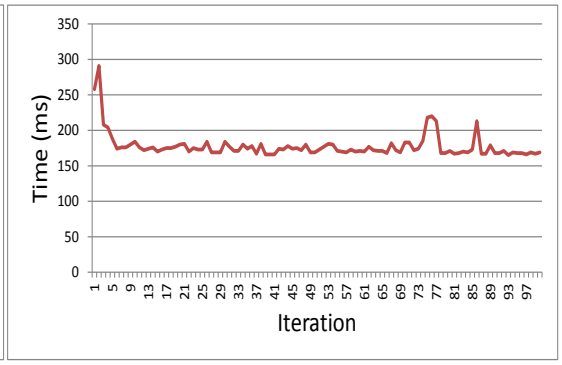


Figure 4.15: Instance selection time for maximum entropy sampling.

effective set of hypotheses based on current training set. The committee evaluates the potential utility of the unlabeled data instance. This utility measure is also called *disagreement measure*. The disagreement measure for QBC was defined by the following equation where $P_c(y|x)$ is the average probability that y is the correct activity label to the committee. The steps of a generalized QBC is show in algorithm 5.

$$x_{VE} = \text{argmax} - \sum_{y} P_c(y|x) \log P_c(y|x) \tag{4.26}$$

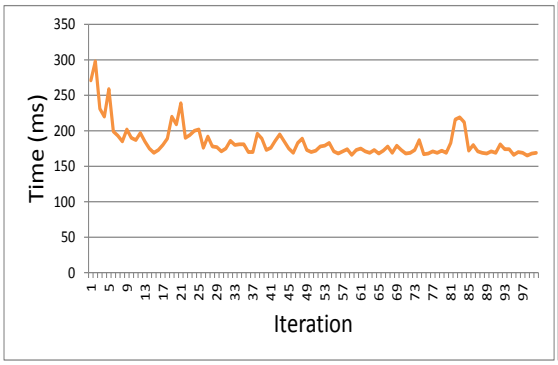


Figure 4.16: Instance selection time for least confidence sampling.

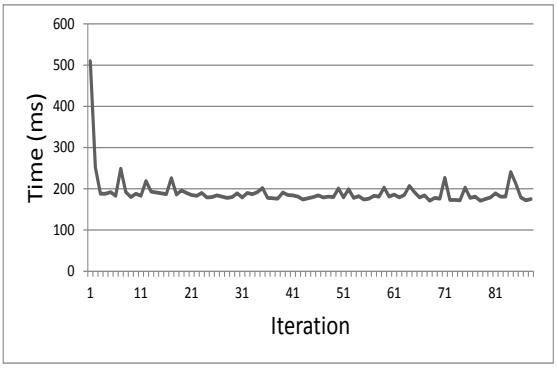


Figure 4.17: Instance selection time for Query By Committee (QBC).

First we cluster the unlabeled data set using our dynamic k-means clustering algorithm. In figure 4.5 we show the intra and inter cluster distance for our dynamic k-means clustering algorithm comparing to simple k-means algorithm. We discuss and compare our active learning algorithm based on each analysis criterion in the following:

4.10.4 Correctly Classified Instances

One of the most important performance measurement for an active learning algorithm is how many data instances were correctly classified. In figure 4.6 correctly classified instance for our algorithm over 100 iterations is plotted. Most of the query strategies show similar results and almost 75% of the instances are correctly classified. However we monitor a lot of changes for least confident sampling as least confident sampling only considers the best prediction and eventually throws away other important information.

4.10.5 Instance Selection Time

Instance selection time depicts the speed of the active learning algorithm. Average instance selection time for all of the strategies were close to 200 ms (figure 4.14, 4.15, 4.16, 4.17).

4.10.6 Mean Absolute Error

Mean absolute error expresses the difference between the predicted value and the actual value. In figure 4.10, 4.11, 4.12 and 4.13 the trend of mean absolute error at each iteration for the query strategies are plotted. For our active learner (figure 4.10), the algorithm converges to lower mean absolute error than other approaches. Mean absolute error for maximum entropy sampling was much higher. After investigating we found that maximum entropy sampling tends to be biased and over generalized. As a result first the mean absolute error started to decrease and in the end it started to increase.

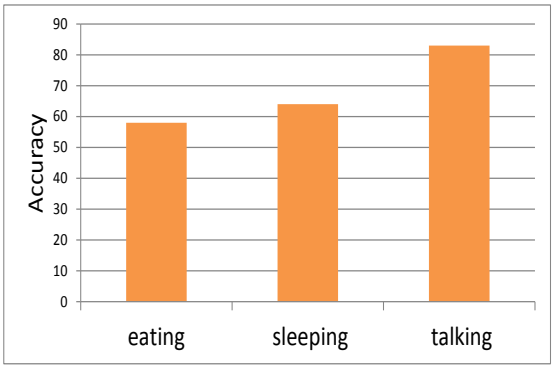


Figure 4.18: Accuracy of unseen activities.

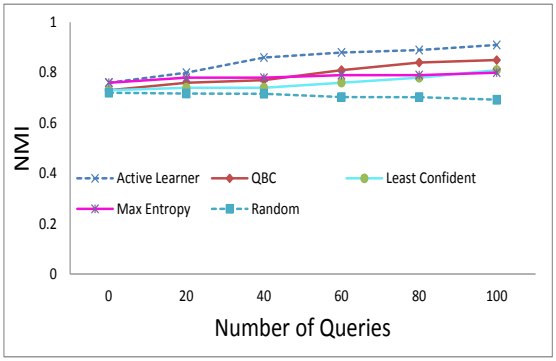


Figure 4.19: Normalized Mutual Information (NMI) for 100 queries.

Table 4.1: Average number of queries for each activity for gaining confidence level 80%

Activity	Avg. Number of Queries
Brooming	2
Cooking	9
Washing	3
Cleaning	5
Eating	12
Sleeping	16
Talking	3

4.10.7 Average Number of Queries

In our experiments we did not train our supervised model with the acquired label using active learning till the label confidence is 80%. Average number of queries for 100 data instances was ≈ 7. Average number of queries for each activity is shown in table 4.1. From the table it is visible that for Cooking, Eating and Sleeping activity the average number of queries were higher. As for these activities there was no object sensors involved, it was difficult for the annotators to take the decision. For sleeping activity, it was the highest as the movements detected while sleeping are mislabeled frequently by the annotators. Also mostly the eating activity was performed in the living room area, where the participants were frequently sitting and doing their chores. This created confusion among the annotators. Cooking was

another difficult high level activity to properly label as just being in the kitchen does not indicate that the participant was cooking. In figure 4.19 we show the change in NMI for our clustering algorithm for different active learning strategies. NMI close to 1 means better correlation. Our active learner converges to 1 faster than other query strategies.

4.10.8 Introducing Unseen Activities

We left out three activities (eating, sleeping and talking) while training our supervised learning model S. Using active learning, we acquired labels of these unseen activities. After querying them, we trained our supervised model with the received new activity labels. In figure 4.18 we show the accuracy of S for these new activities. Talking achieved the highest accuracy because of the object sensor attached to the phone. Whenever a movement of the phone is involved, the supervised learner predicts the motion sensor event as talking. For other two activities it was difficult as living room and bedroom involved so many movements. The accuracy of recognizing sleeping is better than eating because the participants performed more variety of activities in the living room than in the bedroom. And also sleeping during night was properly classified by our supervised learner than sleeping at daylight. After including the collected activity labels using our active learner, we retrained our supervised learning model. Initially we labeled our training data using a labeled data set L consisting of $21,014$ instances. On the other hand we had an unlabeled data set U consisting of $126,874$ samples. Initially the average

accuracy of S was ≈ 81. If we consider only the four activities, after applying our active learning strategy the 87% accuracy was observed. If new unseen activities are introduced, 77% accuracy is achieved.The actual accuracy decreased, because initially the recognition capability of new unseen activities are low. The mentioned accuracy is reported after 100 iterations, so we increased our iteration to 300 and we achieved 79.5% accuracy. So if we query more data instances using our active learner we can achieve better accuracy.

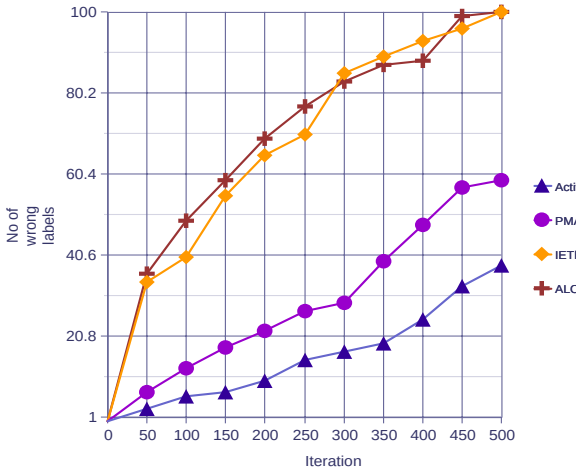


Figure 4.20: Comparison of number of wrong labels generated by different annotator selection model.

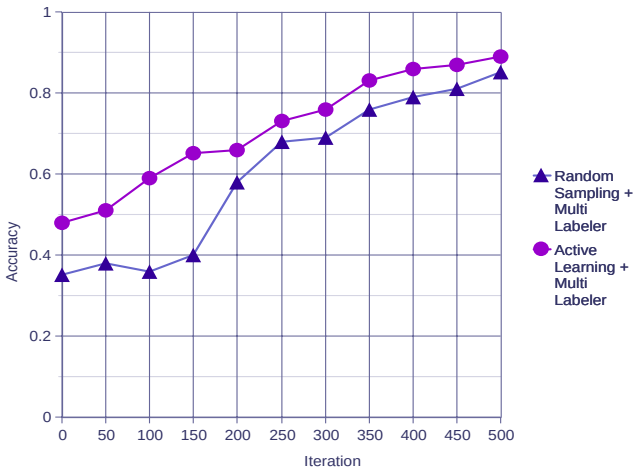


Figure 4.21: Annotator selection accuracy compared to random sampling in multi labeler setting.

4.10.9 Annotator Selection

In this experiment, we validate our annotator selection model and the impact of unsure options. Each annotator is modeled separately with respect to their expertise which means annotators are associated with class labels for which they are expert in

labeling. An annotator will provide correct labels with probability p_t and give unsure feedback when the data instance is confusing to them. We compare our model with other annotator selection models ALC [177], IEThresh [28] and PMActive [178]. As we have already mentioned, introducing wrong labels will enable noise in the model. The average number of wrong labels acquired by these three models compared to our algorithm is shown in Fig 4.20. It is evident that by providing *unsure option* we were able to mitigate the number of incorrect label introduced by our active learning algorithm (Active Learner). In Figure 4.21 we show the increase in accuracy while employing our annotator selection model with our active learning algorithm and random sampling.

4.11 Data Representation for Crowdsourcing and Active Learning

Sensor data representation in a user friendly way is a huge challenge in crowdsourcing domain. Even just applying active learning for querying the label from the same user demands an effective and eloquent representation. One straight forward approach for querying the user can be to ask what the user was doing at a certain timestamp. But all human do not have the ability to precisely recall an event at a certain timestamp. Previous works have proposed to annotate image or video recordings by the crowd [179] [17]. Although using image or video based surveillance provides proper idea about the performed activities to the crowd but it violates the privacy of the users.

It is certainly difficult to disjointly represent all the human activities, as there

are activities which are similar in functional sense. For example, watching television and studying both involves sitting. It is possible that they are performed at different locations in the apartment, so we can pair activities with different indoor locations. [119] used a visual representation of the motion sensor firing sequence and provided the apartment layout to assist the crowd. But motion sensor activation not always provides a straight forward indication of an activity. For example, a person might be in the bathroom just to look at the mirror and the motion sensor will still be activated. In a multi-inhabitant environment this becomes more complex. Object sensors can help us greatly in these cases. With proper usage data of an object paired with motion sensor and location information can ideally pinpoint the activity. Human activities are performed in a pattern. While being at home, people are used to eat, cook, vacuum or perform other household activities in a certain pattern. For example, a person may eat his lunch at a certain time in most of the days. Knowing these activity patterns can also help a crowd to understand the functional and behavioral patterns of the user. So while crowdsourcing, it is important to provide a semantic representation of the user activity pattern so that we can collect noise free labels.

4.12 Discussion and Future Directions

The theoretical foundation of active learning is very rich and resourceful. Researchers have proposed many effective approaches for active learning. But the problem is these solutions are theoretically sound but computationally expensive.

Simpler models like entropy based uncertainty calculation are effective but they imposes bias and runs the risk of becoming over confident on incorrect predictions. It is difficult to make computationally expensive approaches applicable. Also not all the approaches perform the same in a specific domain. Choosing the right approach with respect to the scope of the domain is very important for active learning to be effective. In smart home settings, multiple heterogeneous sensors impose various types of uncertainties like biased readings, failed sensors etc. In multi inhabitant environment the situation is much more complex. On the other hand in multi inhabitant settings, we can take advantage of the inhabitants by asking them to label each others data. But still it is extremely difficult to differentiate overlapped sensor reading and individual's activity.

Due to the dynamic nature and a huge variety of human activities it is difficult to collect the ground truth information for an activity learning model. We can record and annotate data sets for training the system from scratch for individual house and individual person but this will be extremely costly. So leveraging active learning can boost the ground truth collection process. As the basic movements and locations of two or more activities can be same, the probability that they will belong to a same cluster is higher. But automatic discovery of similar activities is not handled in our proposed solution. In such cases we have to consider overlapped clusters to properly separate two distinct activity classes. [18] demonstrated an approach for overlapped activities, but the informativeness measurement has not been discussed in their work.

One of the major challenges for making active learning and crowdsourcing

practical is the expertise of the crowd. The crowd will not always provide the labels correctly. Also the crowd may not be able to provide labels for some instances at all. In some cases getting label for a data instance from only one annotator will not be enough. Relabeling the data instances by other crowd will further validate the consistency of the acquired label. For handling noisy input from crowd this clear approach may be helpful, but still there is a chance that noise will be introduced. As an example, we have to assume that the crowd is not expert and so it will be challenging for them to differentiate between similar activities. Also if an unseen activity which is very much alike with an existing one is queried, the crowd may not provide new class label and annotate it with existing label. This impose both bias and noise in the classification model. So the crowdsourcing model should be able persist an agnostic environment. Researchers have proposed to incorporate reliability of the annotators as an important factor while querying the data. But opportunistically selecting reliable user compulsively is not a good approach as it may annoy the annotator. For this reason, cost-sensitive crowdsourcing has been gaining attention recently.

Chapter 5: Scaling Activity Recognition

5.1 Activity recognition & Deep Learning

Deep learning architectures have been applied increasingly in multi-modal problems which has empowered a large number of application domains needing much less human supervision in the process. As unlabeled data are abundant in most of the application domains, deep architectures are getting increasingly popular to extract meaningful information out of these large volume of data. One of the major caveat of these architectures is that the training phase demands both computational time and system resources much higher than shallow learning algorithms and it is posing a difficult challenge for the researchers to implement the architectures in low-power resource constrained devices. In this chapter, we propose a deep and active learning enabled activity recognition model, *DeActive*, which is optimized according to our problem domain and reduce the resource requirements. We incorporate active learning in the process to minimize the human supervision along with the effort needed for compiling ground truth. The *DeActive* model has been validated using real data traces from a retirement community center (IRB #HP-00064387) and 4 public datasets. Our experimental results show that our model can contribute better accuracy while ensuring less amount of resource usages in reduced time compared

to other traditional deep learning approaches in activity recognition.

5.2 Motivation

Many algorithmic techniques in activity recognition (AR) literature such as sparse coding [180], transfer learning [181], active learning [103], deep learning [182] have been investigated recently for versatile AR application development and deployment. While each approach has its own advantages and disadvantages in terms of scalability, adaptability, and transferability of activity learning, recognition and discovery models, in this work, we particularly focus on leveraging the simplicity of those techniques and exploiting that to fulfill the emergent requirements of large scale activity recognition in heterogeneous settings. Fundamentally we investigate how the underlying inference pipeline of the activity recognition process in deep architecture can be simplified. Such a simplified architecture can then be exploited in various constrained (resource deprived smart devices) and unconstrained environments (heterogeneous smart environments and multiple users population) where the requirements of the applications may vary significantly. This help ramify the performance of deep activity models, and reduce resource footprints in terms of memory, CPU usage and computational time without compromising the inherent power of the core methodologies. Incorporating resource efficiency and cost-effective intelligent labeling techniques with the deep activity models help scale the activity recognition models in diverse environments and showcase the effectiveness of deep activity learning methodology when augmented with other simplistic popular ma-

chine learning approaches.

One of the underpinning challenges in scaling this activity recognition models outside any constrained environments is efficient feature representation from unlabeled noisy source of data and accumulating significant amount of labeled training data. Deep learning based unsupervised machine learning techniques have been investigated to handle the scarcity of activity labels. While deep learning based techniques have shown significant improvements for large scale activity recognition problems [64], fitting the activity models in presence of unlabeled activity samples and mitigating the biasness of overfitting distributions are still challenging research problems [183]. The unsupervised training and fine tuning phase of deep nets also warrant substantial computational resources and labeled data sources, respectively [184]. While shallow and supervised learning [74] suffer from representing well the large scale activity learning model, the hierarchical deep learning [72] model helps capture the finer details of the activity model. It incrementally helps mitigate the need for handcrafted features in layer by layer but at the cost of more resource-hungry computational operations involving calculations of weight and biasness parameters of the model [185]. The main objective of this work is not to depreciate the intrinsic advantage of deep nets or appreciate the inherent advantages of clustering with neural networks, but to showcase the viability of various combination of these approaches for simplified and effective activity recognition at scale.

Trading the balance between this system-level resource need and application-level performance improvement is a non-trivial research problem, and have been investigated in recent activity recognition application domain [72]. Scaling the deep learning model for small footprint devices such as smartphone and smart wrist-watches have also been investigated by exploiting different inference phases of deep model [186]. While the recent approaches investigated the runtime layer compression and deep architecture decomposition by crunching deep learning computational complexity, we propose to investigate simple K-means clustering and active learning approach to curtail the complexity of feature extraction and the burden of ground truth data collection, respectively. While auto encoder is the approach to learn the features in the deep activity models, we investigate a simple K-means clustering strategy to learn and represent the features of the hidden nodes. This help to percolate the simplicity behind the feature extraction, representation, and learning in a deep activity model and its performance in terms of system resource, computational time and performance improvement.

Existing deep learning models assume that activity labels are available and if not human annotator can be passively employed. In particular, stable supply of structured and labeled data is substantial for an effective deep learning algorithm. Despite the existence of plethora of data in pervasive computing, these collections do not provide much information due to poorly or partially labeled. In order to improve the model incrementally, Active Learning (AL) has been employed to select the most informative data instances and subsequently acquire labels of these data

instances. This reduces the burden of labeling data manually and accelerates the training time. If infused together with deep learning, AL framework can help to improve the efficiency of deep model. However AL frameworks only filter out most informative instances from a pool of instances which are relatively small in number. As a result, most of the instances with low uncertainty gets ignored. A handful of labeled instances may not have a significant impact in the training of deep learning. Although most informative instances can play a vital role in learning an important pattern, but instances with low uncertainty can also help to fine tune the parameters of a deep model. In this chapter, we propose to embed active learning at the training phase of deep learning to query the most informative and cost-effective unlabeled sample points to collect the labels and also utilize the low uncertain instances. At first we demonstrate a scalable deep model, *DeActive* by exploiting the encoding capability of k-means clustering. Then we demonstrate a how active learning can help in training the model using a joint loss function.

5.3 DeActive

Our *DeActive* model is designed to work with sensor entities like ambient PIR sensor, accelerometer etc. which are used for activity recognition. As we handle the heterogeneous sensor entities, we need to pre-process the data accordingly due to the variation in data. For example, ambient PIR sensor provides binary values (1=Motion and 0 = No Motion). It is difficult to model an activity recognition classifier using only binary sensor values, so it is necessary to extract some more

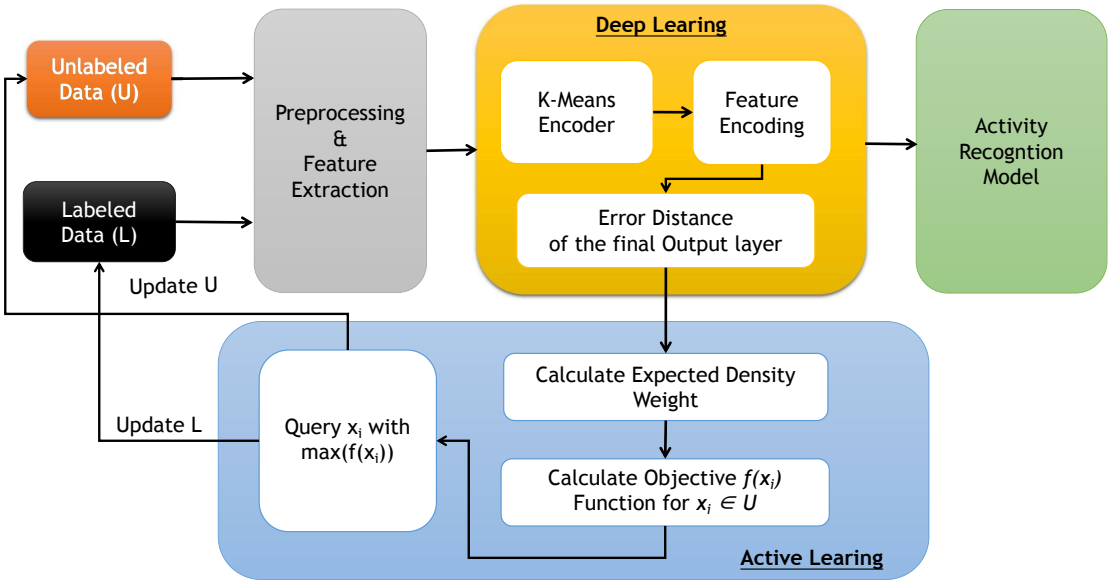


Figure 5.1: Overall framework for *DeActive* activity recognition model. Deep Learning phase is composed of k-means encoders. The output of k-means in the final layer is provided to the Active Learning phase which selects the most informative instances from the unlabeled data pool.

information using the data. On the other hand, accelerometer sensors provide human movement acceleration which has been an important indicator for activity pattern recognition in recent years. As acceleration does not encounter binary values, the processing of accelerometer data is different compared to PIR sensors. As a result our *pre-processing* step handles different sensor data sources and extracts features. *DeActive* model encompasses two important components - *Deep Architecture* and *Active Learning*.

Real-time human activity and context monitoring using mobile devices or wearables has become an essential constraint. Since deep learning algorithms have

high complexity in terms of computation and resource availability [187], researches are currently focusing on accelerating deep learning on mobile devices [64] [188]. Most of the smart home system controllers are embedded systems and have very limited resources. For example the specs of Samsung SmartThings hub V2 is: 1GHz ARM Cortex-A9 CPU, 512MB DDR3 RAM, and 4GB Flash memory. On the other hand these devices do not have any GPUs which can assist executing deep architectures. The authors of [74] proposed a software accelerator *DeepX* for low power mobile devices. *DeepX* is designed to optimize the execution by decomposing the deep architecture and innovative use of resources. In *DeActive*, we try to optimize the parameters of fully connected deep learning model by using k-means as our encoder. The authors of [185] have shown that if properly initialized according to the problem domain, k-means can accelerate the encoding process. We just need to tune the parameter k in the hidden layers, as a result of which the calculation of millions of parameters as in stacked RBM autoencoders is minimized.

Active Learning strategies are used to collect ground truth information with minimal human supervision. Simpler active learning strategies like margin sampling, uncertainty sampling and least confidence are easier to implement, however overtime these sampling strategies become biased and overconfident [189]. Other popular query strategies like Query by Committee (QBC) and disagreement based approaches have higher computational complexity as they have to maintain a set candidate hypothesis space which can get intractable overtime. It is possible to initialize a set of hypothesis space with smaller cardinality, however the probability of ignoring the true hypothesis always remains. Cluster based active learning methods

may provide a significant advantage over simpler ones in terms of effectiveness [190]. By exploiting the input distribution, we can cluster the most informative instances after each iteration and query the ideal instances of those clusters like the centroids. However cluster based approaches have limitations like querying the outlier cluster. In *DeActive*, we employ a density-weighted heuristic to calculate the most informative data instances. The idea is to query the data instances which lie in the dense region of the cluster so that we can label the neighboring instances as well. We consider the euclidean distance as our similarity measure among the instances while calculating the density. In order to remove the outliers from being queried we take advantage of our k-means clustering and use the *silhouette coefficient* in our final objective function. This coefficient is calculated by the distance measurement between the points in its cluster and other surrounding cluster centers. We also exploit this coefficient to filter out the representative instances of each clusters. If the coefficient value is higher than a predefined threshold, then we consider it and assign its label to all other instances inside the cluster. In Figure 5.1 we demonstrate our overall *DeActive* framework.

5.4 Deep Active Learner

In Figure 5.2 we show a high level architecture of our proposed model. Existing algorithms consider active learning separate from the principal deep learning process. Traditionally,a deep model is trained first using the labeled data instances and occasionally unlabeled instances are used to pre-train and initialize the model

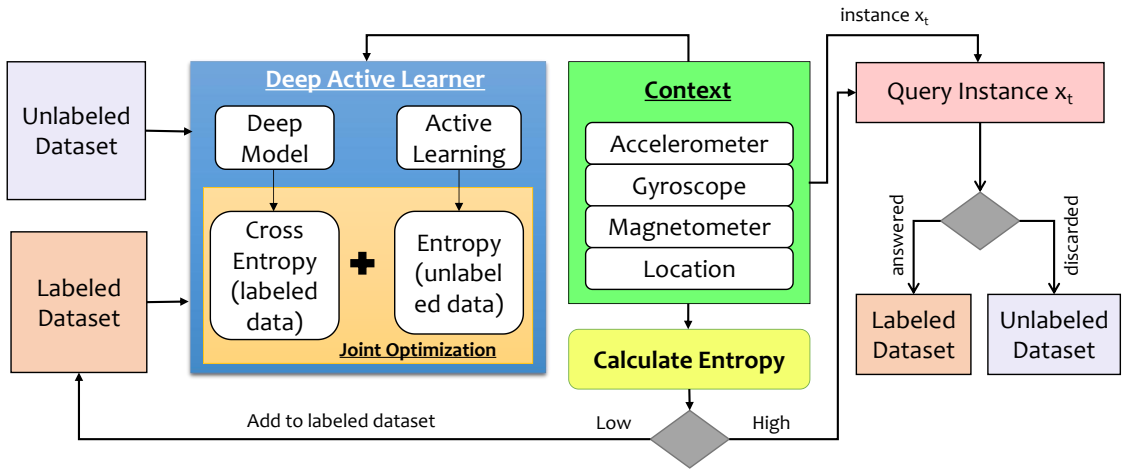


Figure 5.2: A high level architectural overview of Active deep learner.

parameters. The posterior probabilities of the class labels are then collected from the final softmax layer. An active learning sampling strategy is then applied on the posterior probabilities to measure the informativeness of the current instance. Here informativeness is considered in terms of the effect on the base learning algorithm if the label is known for an instance. The effect can be a shift in the decision boundary which eventually helps in reducing overall error of the model. Uncertainty based informativeness measurement methods are highly acclaimed in such cases because of their simplicity and reduced complexity compared to other sampling strategies like searching through hypothesis space or expected error reduction. In our model we measure the level of uncertainty by calculating the entropy of the data instance. The more uncertain an instance is, higher its entropy will be. This situation arises when posterior probabilities of the output class labels are close. This indicates that the classifier was unable to predict the actual label of the provided data instance. However, deciding the informativeness by just measuring

the entropy of the incoming data instance makes the process myopic and becomes overconfident as time progresses. Thus it is necessary to properly determine which instances are truly informative. For example, an outlier instance which does not belong to any class will have high entropy which makes it an informative instance accordingly. Therefore, the learning model needs to be able to properly differentiate which instances it should focus on. In order to accomplish this we fuse the deep model with the active learning algorithm by applying a joint optimization of the individual objective functions. We present a joint loss function which consists of cross-entropy loss of the neural network and the entropy function. We train our model with both labeled and unlabeled data instances. Labeled instances help in optimizing the cross-entropy part and the unlabeled instances help in optimizing the network parameters to learn the informative instances.

We collect data from the smartphones carried by the users. We exploit four sensors entities - accelerometer, gyroscope, magnetometer and the location sensor. The first three sensors provide three axis data and the location sensor provides 2D data. The streaming data are cached and form a pool of instances. We calculate the entropy of these instances and select the instances with maximum entropy. After selecting the informative instances, we pose the queries to the annotators.

5.5 Unsupervised Feature Learning

In recent years a lot of research have been conducted in the area of deep learning for representing data in lower dimension. One of the prominent approach for

feature learning in deep architectures is to use layers of non-linear processing units for extraction and embedding of features. These layers are referred as auto encoders. These auto encoders are responsible for assembling lower dimensional representation of the higher dimensional data. Given an input $x \in R_n$, an auto encoder attempts to learn an encoding function $f(x) \in R^k$, $k << n$ by iteratively minimizing the error of reconstructing x through a decoding function $g(f(x)) \cong x \in R_n$. The authors of [185] showed that spherical k-means or spectral clustering can be used as an alternative to encoders using sparse encoding or PCA. One major drawback for using k-means is that the capability of discovering sparse directions in data largely depends on the dataset size and dimension. If the data dimension is higher, we will need a large volume of data to outperform other encoders. However, we consider data is abundant and so our concern is to speed up the process. In this section we discuss how we can exploit the k-means algorithm as our encoder.

5.5.1 K-Means Clustering

K-means clustering is a partitioning method where a set of observations are partitioned into a specified number of clusters and similar observations reside in the same cluster. Given a set of observations $X = \{x1, .., x_n\}$, the observations are assigned to k clusters by minimizing the error distance between cluster centers $C = \{c_1, c_2, ..., c_k\}$ and X, while assigning $W = \{w1, ..., w_k\}$ class indexes:

$$E(C, W) = \sum_{i=1}^{n} \|x_i - c_{w_i}\| \tag{5.1}$$

In most of the cases this error distance Eqn. 5.1 is minimized using heuristics like Loyd, Elkan etc [191]. But these heuristics are unable to adapt in case of large amount of data. As in our case we plan to employ K-means as our hidden encoder layer in deep architecture, so the algorithm needs to process a large volume of data. To address this issue we propose to design our K-means using Stochastic Gradient Descent. This version of K-means clustering has been proposed in the literature for addressing large-scale learning tasks, due to its superior performance and low resource footprint [192]. The objective function for k-means is $Q_{kmeans} = \min_k \frac{1}{2}(x - w_k)$. We calculate the gradient $\bigtriangledown_{w_k} Q_{kmeans} = w_k - x_i$ by taking the partial derivative of the Objective function. After this we update the learning rate η and weight vector w according to Eqns. 5.2 and 5.3. The whole process iterates until the cluster centers are no longer changing.

$$\eta_k = \eta_k + 1 \tag{5.2}$$

$$w_k = w_k + \frac{1}{\eta_k}(x - w_k) \tag{5.3}$$

5.5.2 K-Means as Encoder

The *K-means* clustering algorithm takes two parameters, number of clusters k and a set of observation vectors V. The algorithm returns cluster centers or centroids $C = \{c_1, c_2, ..., c_k\}$ for each of the k clusters. While associating an observation vector v_i to a cluster k_j, the primary goal is to minimize the distance between the vector and cluster center. The result of k-means can be employed to quantize vectors. The goal of vector quantization is to form encoding of vectors which reduces the

expected distortion. Eventually k-means algorithm extracts a dictionary $D \in R^{n \times k}$ of k vectors where each vector $x^{(i)} \in R_n, i = 1, ..., m$ is mapped to an encoded vector which reduces the error in reconstruction. The definition of the dictionary is as follows:

$$minimize \sum_i \|D_s^{(i)} - x^{(i)}\|_2^2 \tag{5.4}$$

$$where \, \|s^{(i)}\|_0 \leq 1, \, \forall i \; and \; \|D^{(j)}\|_2 = 1, \, \forall j$$

In Eqn. 5.4, $s^{(i)} \in R^n$ is a code vector associated with input data points $x^{(i)}$. $D^{(j)}$ is the jth column of dictionary D. Our goal is to form the dictionary D and extrapolate the code vectors of each data point $x^{(i)}$ in such a way that if given $s^{(i)}$ and D, we can reconstruct the original $x^{(i)}$. Our objective is to reduce the squared difference between $x^{(i)}$ and its analogous reconstruction $D_s^{(i)}$. This is accomplished by two constraints described in Eqn. 5.4. The first constraint $\|s^{(i)}\|_0 \leq 1$ means that each code vector $s^{(i)}$ is forced to have at most one non-zero entity. The second constraint $\|D^{(j)}\|_2 = 1$ ensures that each column in the dictionary is of unit length. The encoding and reconstruction mechanisms are similar to sparse coding [193]. The difference between K-means and sparse coding is that the latter allows more than one non-zero entity in each code vector $s^{(i)}$ which leads to more precise representation. Although sparse coding can be interchangeable here but the simplicity and scalability of K-means can be useful in scaling our activity recognition system. Also we need to solve a convex optimization problem for every code vector in sparse coding which requires an immense endeavor and conclusively makes it

difficult to deploy at large scale. The optimal code vector $s^{(i)}$ used in K-means is:

$$s_j^{(i)} = \begin{cases} D^{(j)T}x^{(i)}, & \text{if } j == \text{argmax}_l \, |D^{(l)T}x^{(i)}| \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

Using Eqn. 5.5 we can form the code vectors rapidly and can train very large dictionaries immediately by alternative optimization of D and s. Also we only have one parameter to tune for K-Means which is the number of centroids for each hidden layer. At the final layer we apply k-means to find the desired k class indexes.

5.5.3 Initialization

One of the major problems of k-means algorithm is that it may produce empty clusters depending on the initial centroids. Although for static cases this problem is trivial and can be avoided by running the algorithm for couple of times. If empty cluster problem is not handled, it may lead to significant performance reduction. Therefore, it is important to properly initialize the centroids. Random initialization of initial central vectors is one of the simplest approach, but this will not be effective for sensor data. Whether the data are coming from ambient or wearable sensors, the data tend to group too densely in some areas which results in a large number of centroids starting in a dense region. Most of these centroids end up becoming clusters with very few data instances. In order to avoid such scenario, we propose to randomly initialize the centroids from a Normal distribution and then normalize them to unit length in accordance with our constraint. Let $X = \{x_1, ..., x_i\}$ be our

data set and $S = \{s_1, .., s_i\}$ is our corresponding code vector matrix. We update the centroids according to the following equation:

$$\begin{aligned} D &= \underset{D}{\operatorname{argmin}} ||DS - X||_2^2 + ||D - D_{old}||_2^2 \\ &= (SS^T + I)^{-}1(XS^T + D_{old}) \approx XS^T + D_{old} \end{aligned} \tag{5.6}$$

5.5.4 Feature Mapping

K-means returns a set of cluster centers or centroids with k cardinality, which we use to design our feature mapping function. We consider two choices for our feature mapping function: i) We add k binary features to each sample, where each feature j has value one if and only if the jth centroid learned by k-means is the closest to the sample under consideration (Eqn. 5.7). ii) A non linear mapping, where we calculate the mean distance (μ) between the sample under consideration and other centroids and then a feature has value if and only if the centroid learned by k-means is within the radius of μ (Eqn. 5.8).

$$f_k^1(x) = \begin{cases} 1 & \text{if } k == \operatorname{argmin}_j ||c^{(j)-x}||_2^2 \\ 0 & \text{Otherwise} \end{cases} \tag{5.7}$$

$$f_k^2(x) = max\{0, \mu(z) - z_k\} \tag{5.8}$$

5.5.5 Active Learning

Active Learning can help scaling our activity recognition model and reduce the amount of effort needed for manual annotation. While deep learning assumes to have

passive labeled data available in per-training phase or select them randomly from a pool of labeled datasets, we propose to investigate how active learning could help to improve the activity recognition performance at scale. Augmenting the training phase of deep activity models with active learning is a crucial step to reduce both computational time and system resource requirements. Therefore, our primary goal here is to help find the most informative data instance which we will query from the user. Here most informative instance is defined as an unlabeled instance which will bring the greatest change in our current training model if label is provided. Let U be the set of unlabeled data instances and L be the set of labeled data instances. The active learning algorithm will select the most informative data instance out of k samples from U in a pool based sampling setting. First we pre-train the deep learning network in an unsupervised way using the unlabeled instance set U. Then we use the labeled data set L to train the final output layer of k-means classifier, followed by fine tuning the network. We consider an active learning strategy using the data density by explicitly considering the structure of the data while selecting queries. If we consider the data instances with high information content, the sampling strategy will get biased and over confident as the time progress. So we also consider the data instances which are representative of the underlying distribution. Here we scrutinize the data instances which lie in the dense region of a cluster. The information density heuristic is calculated by the following equation:

$$f(x) = \underset{x}{\operatorname{argmax}}\, \Phi(x) \times \left(\frac{1}{card(U)} \sum_{x \in U} sim(x, x^*) \right)^{\beta} \tag{5.9}$$

In our objective function $f(x)$, *card(U)* depicts the cardinality of our unlabeled data instance pool and $\Phi(x)$ represents the utility of x according to expected error reduction of our k means classifier. The $sim(x, x^*)$ measures the similarity between x and all other data instances. Using equation 4.1 we get our loss function as following:

$$L(x) = \text{argmin} \sum_{i=1}^{n} \|x - x^*\| \tag{5.10}$$

Our activity recognition model has no idea about what the error will be when it receives a label from the query. Using the decision theoretic approach instead of reducing error as a known value, we minimize it as an expected value by using the model's posterior distribution as an acceptable approximation. Using this intuition our utility measure $\Phi(x)$ is defined as following:

$$\begin{aligned} \Phi(x) &= \text{argmin}_x \, E_{y|x}[L(x)] \\ &= \text{argmin}_x \sum_y P(y|x)[\sum_{i=1}^{n} \|x - x^*\|] \end{aligned} \tag{5.11}$$

The term $\left(\frac{1}{card(U)} \sum_{x \in U} sim(x, x^*) \right)^{\beta}$ in Equation 5.9 weights the informativeness of x by its average similarity to all other instances. The parameter β controls the relative importance of the density term. Our objective function can be less sensitive to the outliers as it works in a dense region only. However if the dense region is in between the boundary of two clusters it may choose unnecessary data instances and outliers. To ensure that we introduce *silhouette coefficient* s_c^i in our objective function. Let $d(i)$ be the average dissimilarity of x_i with all other data within the same vicinity. This portrays how well x_i is assigned to it's own cluster. $d(i)$ is defined as the average distance from x_i to all other points in its own cluster. We

define $e(i)$ to be the lowest average dissimilarity of x_i to any other cluster, of which x_i is not a member. The cluster with lowest $e(i)$ is said to be the *neighboring cluster* of the cluster where x_i resides. Now we define our silhouette coefficient as following:

$$s_c^i = \frac{e(i) - d(i)}{\max\{d(i), e(i)\}} \tag{5.12}$$

$$s_c^i = \begin{cases} 1 - \frac{d(i)}{e(i)}, & \text{if } d(i) < e(i) \\ 0, & \text{if } d(i) = e(i) \\ \frac{e(i)}{d(i)} - 1, & \text{if } d(i) > e(i) \end{cases} \tag{5.13}$$

The value of s_c^i ranges between -1 and 1. Smaller $d(i)$ represents x_i to be analogous to its own cluster. On the other hand large $e(i)$ illustrates x_i to be poorly matched to its neighboring cluster. As a result, s_c^i close to 1 depicts appropriately clustered instance and close to 0 means x_i resides on the border of two clusters. So by plugging in the coefficient into our objective function, we ensure that no outliers or unnecessary data instances get queried. The final objective function for our active learning method is

$$f(x) = \underset{x}{\operatorname{argmax}} \left[s_c^i \Phi(x) \times \left(\frac{1}{card(U)} \sum_{x \in U} sim(x, x^*) \right)^{\beta} \right] \tag{5.14}$$

The overall active learning strategy of our *DeActive* model is summarized in Algorithm 6.

5.6 Deep Active Learner

Our model fuses active learning with deep model instead of just considering the resultant posterior probabilities from the final layer of the neural network to

measure the informativeness of an instance. Our deep model uses the standard cross-entropy loss (Eqn. 5.15) to optimize the network parameters. In Eqn. 5.15 N denotes the number of class, c is the target vector and y is the output vector which is calculated using the softmax function $y_i = \frac{e^{p_i}}{\sum_k^N e^{p_k}}$. Here, $p_i = \sum_{j=1} h_j w_{ij}$ is the weighted sums of the hidden layer activations.

$$L_c = -\frac{1}{n_l} \sum_{i=1}^{N} [c_i \log(y_i)] \tag{5.15}$$

We exploit the entropy of an instance as the measure of informativeness or uncertainty. The entropy of an instance $x_i \in U$ from the unlabeled data instance pool U is defined as,

$$H(y_i) = -\sum_{j=1}^{N} y_i^j \log y_i^j \tag{5.16}$$

Here y_i^j denotes the probability of assigning instance x_i to class j. We get this assignment probability from the final layer of our neural network. In traditional active learning settings, unlabeled data are not used in the training process. Queries are posed based on the measurement of uncertainty only which is myopic and become overconfident about wrong predictions overtime. In order to fuse deep model and active learning together we employ a joint loss function [194] which is defined as following

$$\begin{aligned} L &= \frac{1}{n_l} \sum_{i=1}^{n_l} L(c_i, y_i) + \frac{\lambda}{n_u} \sum_{i=n_l+1}^{n} H(y_i) \\ &= -\frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{i=1}^{N} [c_i \log(y_i)] - \frac{\lambda}{n_u} \sum_{i=n_l+1}^{n} \sum_{j=1}^{N} y_i \log y_i^j \end{aligned} \tag{5.17}$$

In Eqn. 5.17, n_l denotes the number of labeled data, n_u is the number of unlabeled instances and the parameter λ regulates the effect of entropy loss. We

can derive the gradient of our loss function 5.17 by evaluating the partial derivative of the cross-entropy loss and the entropy. The gradient of the cross-entropy loss is

$$\begin{aligned}\frac{\partial L_c}{\partial h_{qr}} &= -\frac{\partial}{\partial h_{qr}}\left(\frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{i=1}^{N}[c_i \log(y_i)]\right) \\ &= \frac{\partial}{\partial h_{qr}}\left(\frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{i=1}^{N}[c_i \log(\frac{e^{p_i}}{\sum_k^N e^{p_k}})]\right) = \frac{1}{n_l}(p_{qr} - y_q)\end{aligned} \tag{5.18}$$

The partial derivative $\frac{\partial H}{\partial h_{ij}}$ is the gradient of entropy H with respect to hidden layer unit $h_i j$. The gradient of the entropy is

$$\begin{aligned}\frac{\partial H}{\partial h_{qr}} &= -\frac{\partial}{\partial h_{qr}}\left(\frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{N} x_i \log y_i^j\right) \\ &= -\frac{\partial}{\partial h_{qr}}\left(\frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{N}\frac{e^{p_i}}{\sum_k^N e^{p_k}}\log\frac{e^{p_i^j}}{\sum_k^N e^{p_k}}\right) \\ &= -\frac{\lambda}{n_u}\sum\sum(p_{ij}\log\sum_{j'} e^{h_{ij'}} + p_{ij}p_{ir} - p_{ij}\log\sum_{j'} e^{h_{ij'}}p_{ir}) \\ &= \frac{\lambda}{n_u}\Bigg[- p_{qr}h_{qr} - p_{qr} + p_{qr}\sum_{j=1}^{C} p_{qj}h_{qj} + p_{qr}\log\sum_{j'} e^{h_{qj'}} \\ &+ p_{qj}p_{qr} - \log\sum_{j'} e^{h_{qj'}}p_{qr}\sum_{j=1}^{C} p_{qj}\Bigg] \\ &= \frac{\lambda}{n_u}p_{qr}\left[\sum_{j=1}^{C} p_{qj}h_{qj} - h_{qr}\right]\end{aligned} \tag{5.19}$$

The overall gradient then for our back-propagation update is the sum of the cross-entropy gradient and the entropy gradient.

$$\frac{\partial L}{\partial h_{qr}} = \frac{1}{n_l}(p_{qr} - y_q) + \frac{\lambda}{n_u}p_{qr}\sum_{j=1}^{C} p_{qj}h_{qj} - h_{qr} \tag{5.20}$$

By assimilating the entropy in our loss function, the model becomes more sensitive to highly informative instances. Also we can use all the unlabeled data instances to train our model. Our deep active learning model is summarized in Algorithm 7. We employ a convolutional neural network for our deep model. The cross-entropy loss is computed using the labeled data instances and entropy loss is computed using the unlabeled data instances. In each step, a batch is formed using both labeled and unlabeled data instance. Our proposed joint optimization ensures the network to reduce entropy along with the classifier loss. This enables the classifier to filter out highly informative instances more efficiently and not focus on outliers.

5.7 Preprocessing

In order to validate our *DeActive* model, data from two types of sensor modalities are considered - ambient motion sensor and accelerometer sensor from smartphone or wearable. In this section we discuss the preprocessing of data from these sensor modalities.

5.7.1 Ambient Sensor

Ambient motion or infrastructural sensors are embedded in smart environments. Largely these sensors are PIR motion sensors which detect motion in the vicinity. It provides a value of 1 if a motion is detected otherwise 0. Other type of sensor deployed is door sensors which also provides binary values (OPEN and

Algorithm 6 *DeActive* Active Learning

Input: U = A pool of unlabeled instances $\{(x)^u\}_{u=1}^{U}$,

L = A pool of labeled instances $\{(x)^l\}_{l=1}^{L}$,

E_{dist} = Error Distance from K-Means in the final Output layer of the deep architecture

k = Number of Clusters or Activities

Output: Most informative data instances in each cluster.

D = {}

for every $x_i \in U$ **do**

    Calculate the loss using E_{dist} in eqn 5.10

    Calculate base utility measure $\Phi(x)$ by taking the expected value of the loss give label y.

    Calculate the silhouette coefficient s_c^i for instance x_i

    d Calculate the informativeness $f(x)$ of x_i using eqn 5.14

    $D = D + d$

end for

q = instance with maximum $f(x)$ and query for label l

if $s_c^q > \delta$ **then**

    I = Neighbor instances of q

    Assign label l to instances in I

end if

$L = l + I$

Algorithm 7 Deep Active Learning

Input: U = a pool of unlabeled instances $\{(x)^u\}_{u=1}^{U}$, L = a pool of labeled instances $\{(x)^l\}_{l=1}^{L}$, learning rate ϵ, batch size k, number of epoch t

Output: Trained Active Deep Model

initialize network parameters, weights w and bias b

for $t = 1, 2, \ldots T$ **do**

    Create mini batches of size k from U and L

    for each $X_{i:i+k}^{l} \in L$ and $X_{i:i+k}^{u} \in U$ **do**

        Calculate the gradient of cross-entropy loss L_c using $X_{i:i+k}^{l}$

        Calculate the gradient of entropy H

        Gradient of loss $\frac{\partial L}{\partial h_q r} \leftarrow \frac{\partial L_c}{\partial h_q r} + \frac{\partial H}{\partial h_q r}$

        Update network weights

    end for

end for

CLOSE) based on the motion of the door. Each sensor sequence is associated with a timestamp which is discretized to an integer value, day of the week which is also converted to an integer (0-6) where Monday being 0, ID of the previous activity performed and finally the length of the current activity measured in number of sensor events.

5.7.2 Accelermeter Sensor

Deep learning architectures are designed to process and deal noises of sequential sensor data by performing unsupervised feature learning. To extract meaningful information from the data we apply a noise filter and extract statistical features from the data. We apply a simple low pass filter to smooth out the arbitrary noises in the accelerometer data and a high pass filter to remove the effect of gravity. The accelerometer signals are then separated into frames using a fixed width sliding window with 10% overlapping. We used a 3 seconds sliding window and set the sampling frequency at 60Hz. We then extract statistical features which include:

- The mean, standard deviation and variance.
- Signal and differential vector magnitude.
- Signal entropy to differentiate between signals that correspond to different activity patterns but similar energy signals.
- Pairwise correlation of between each pair of dimensions.
- Zero-crossing rate in each dimensions.

- Weighted average of the frequency components to obtain a mean frequency.
- Magnitude and Energy of Fast Fourier Transform (FFT).

For our Deep Active Learner network, we collected data from 20 (14 male and 6 female) participants over the course of a month in a real-world settings by installing our developed application in the respective platform. Before the data collection, the participants were given instructions to manually log their daily activity routine as much as possible which can help us evaluate the performance of the annotators. The set of activities our system monitors include {*walking, eating, running, working, sitting, standing*}. The participants were instructed to keep the phone in their front pant pocket if possible. The 3D sensors were sampled at 60Hz and the location information were extracted every 10 mins. To train our classifier and the annotator selection model, we collected training data from 10 out of these 20 participants in a controlled environment where the participants followed a script and performed the 6 activities. Each subject participated in 5 sessions (30 mins per session) and performed each activity multiple times. We recorded the sessions using video camera in order to collect the ground truth information. In order to train our annotator selection model, we created a dataset empirically from the experts. The accelerometer data are divided into fixed sized frames using sliding window with 50% overlap with a window size of 64. The activity label for each frame are selected by the majority class label in that frame.

5.7.3 Data Normalization

After modeling the features for both sensor modalities, our next step is to normalize the data so that the features will be rescaled as we want all features to contribute equally. The normalized data will have the properties of standard normal distribution with zero mean ($\mu = 0$) and unit variance ($\delta = 1$). As we are using stochastic gradient descent for centroid calculation in our k-means encoder, certain weights may update faster than others since the feature values play a role in the weight updates with features being on different scales. Computation fo distance measurement in k-means envisages each feature uniformly and so we have to ensure that units of features do not alter the relative approximation of observations. Also If a variance of a feature which is orders of magnitude larger than others, it might influence the classification and make the class estimator unable to learn from other features correctly as expected. So normalizing the features so that they are centered around 0 with a standard deviation of 1 is important. The normalization is done using: $z = \frac{x-\mu}{\delta}$.

5.8 SenseBox Implementation

We collected real life data using our *SenseBox* [195] smart home system. The *SenseBox* system is composed of an ARMv5-based hub that is placed in the residences of volunteers, along with several sensors (passive infrared and accelerometer) that communicate back to the sensor hub via AXSEM AX5043 radios operating in the 900 MHz ISM band. The receivers for the AXSEM radios are connected via

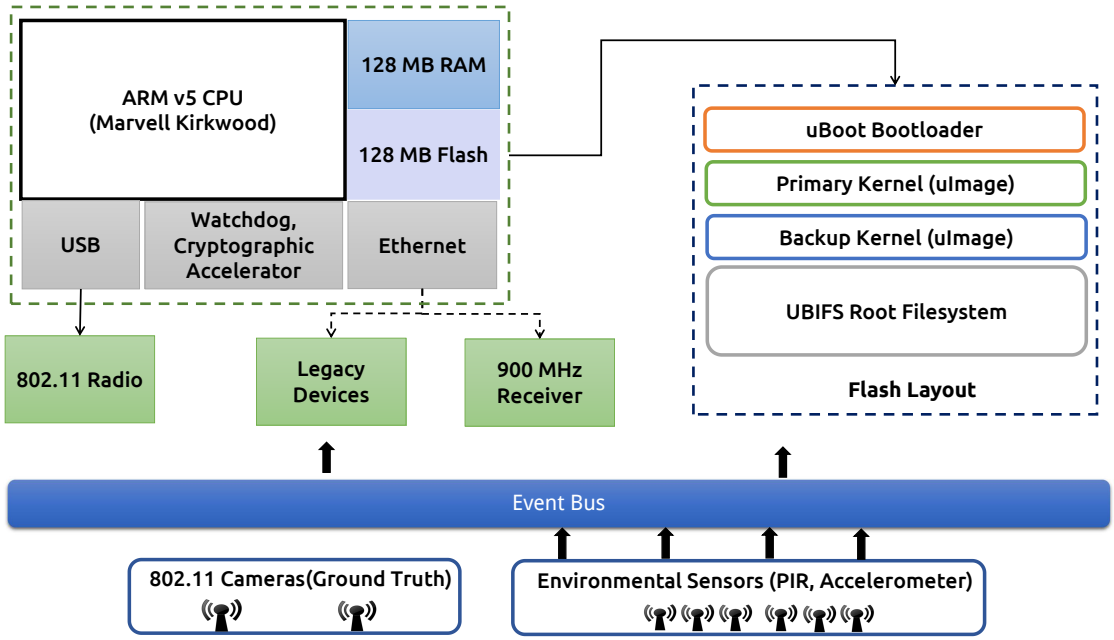


Figure 5.3: *SenseBox* architecture which has ARM v5 CPU. Multimodal sensors dump the streaming data in the Event Bus and the system reads the new data from there.

Ethernet which are inexpensive and provide adequate reliability for our application. The ARMv5-based hub is built on top of a consumer NAS device, the Cloud Engines PogoPlug. Using the publicly available GPL sources, we rebuilt the kernel to support kernel-level features we required in this application or felt we may require in the future (e.g. Video4Linux, NAT, support for various wireless devices.) As is typical in this scenario, subtle issues with downstream kernel code necessitated fixing several issues before the kernel was able to be successfully built and stable. As our deep learning algorithm uses torch [196] library, we have built torch for ARM processor.

5.9 Experimental Results

In this section we validate *DeActive* and compare the outcome with other popular strategies. Apart from using our own data collected using *SenseBox*, we also used four publicly available datasets to justify our framework. We provide descriptions of the datasets in the following:

Opportunity Dataset: The OPPORTUNITY dataset [197] encompasses both ambient motion and accelerometer sensor data from four participants. Each participant performed a session five times and in each of these sessions they performed a set of kitchen activities. Accelerometer sensors are placed on 12 different places of the body. We considered a subset of these data set which included accelerometer data from the upper limbs of the body. About 75% of the data instances do not correspond to any class.

CASAS Dataset: The CASAS dataset [198] contains ambient motion sensor data deployed in the WSU smart apartment. Couple of item sensors are also mounted on some objects to detect their usages. The data represents participants performing five ADL activities in the apartment (Make a phone call, Wash hands, Cook, Eat, Clean).

WISDM Dataset: The WISDM dataset [199] has triaxial accelerometer data from 29 users collected by android smartphone. This dataset has $1,098,207$ data instances of 6 classes - Walking, Jogging, Upstairs, Downstairs, Sitting and Standing.

Skoda Daphnet Dataset: The Skoda Daphnet dataset [200] contains the freezing of gait in users with Parkinson's disease. Three acceleration sensors on the hip,

thigh and ankle were attached to 10 subjects. The data are classified into three classes - Freeze, No Freeze and No Experiment.

SenseBox Dataset: Using our own smart home system *SenseBox* [195] we collected data from 10 participants [201] (IRB - #HP-00064387) from a retirement community. Three ambient motion sensors and seven object sensors were installed in each participant apartment. We installed the motion sensors in three different rooms (bedroom, living room and kitchen) of each single bedroom apartment. The object sensors were mounted on difference appliances (broom, trashcan, laundry basket, dustpan and phone). The users also wore a wearable device on their dominant hand which provided 3D acceleration data for each of the activities. Our dataset has five activities - Cooking, Cleaning, Brooming, Eating, Sleeping. The ground truth information was collected using video recordings. Each participant provided 24 hours of continuous sensor data for 20 days.

We evaluate our model using precision, recall and F1 measures. However these measures also exhibit biasness due to population prevalence and label bias as they inherently ignore handling of negative examples [202]. As a result we also calculate *Informedness* and *Markedness* measures to avoid bias by integrating inverse recall and inverse precision respectively.

$$markedness = precision + inversePrecision - 1$$

$$informedness = recall + inverseRecall - 1$$

Markedness and Informedness articulate how marked and informed the classifier is respectively with comparison to chance. We evaluated our active learning algorithm

by comparing with other simple active learning methodologies - Maximum Entropy sampling, Query By Committee and Random sampling. In order to compare these methods we calculated Normalized Mutual Information (NMI) using the ground truth information. Both the true activity class label and queried label assignment are considered as random variables in NMI. NMI measures the mutual information between these two assignments and normalizes them to zero to one range. If we consider K be the random variable of queried class labels of data instances and C be the true labels then the NMI is computed by equation: $NMI = \frac{2I(C;K)}{H(C)+H(K)}$. Here $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between random variables X and Y.

5.9.1 Performance Analysis

In this section, we examine the performance of our *DeActive* model on real world datasets described in the previous section. In our proposed model we use 3 hidden layers with 150 neurons in each layer. We first normalize the data with zero mean and standard variance. The deep activity recognition models are trained using stochastic gradient decent with mini-batch size of 0.60. In Table 5.1 we compare our model with other deep architectures for *SenseBox* dataset. It is apparent that our model achieves better accuracy (92.84%) with 3 hidden layers and 150 nodes at each layer. We experimented with different number of features and empirically we got better results for 500 features for all the deep architectures. For each of the dataset we pick a set of labeled samples with 1000 data instances and train our classifier

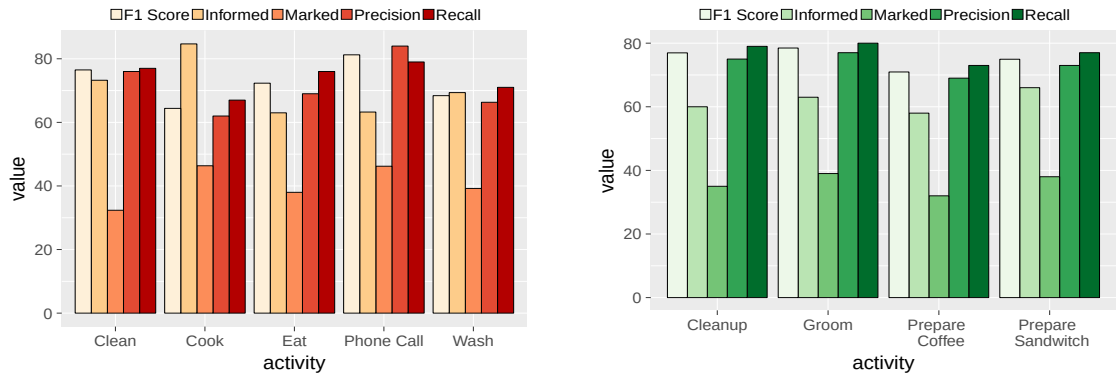
with 80% data of this set. We leave the rest of the 20% for validation. Due to relatively small number of classes available in our dataset we experienced overfitting problem. We applied "dropout" method which is a widely used technique to tackle the overfitting problem. We trained our model offline using our lab server.

Architecture	Accuracy(%)
DBN	85.52
RBM	89.78
CNN	86.16
Sparse Autoencoder	84.11
DeActive (1 Layer)	87.34
DeActive (2 Layers)	89.34
DeActive (3 Layers)	92.34

Table 5.1: Accuracy of different deep architectures on *SenseBox* dataset

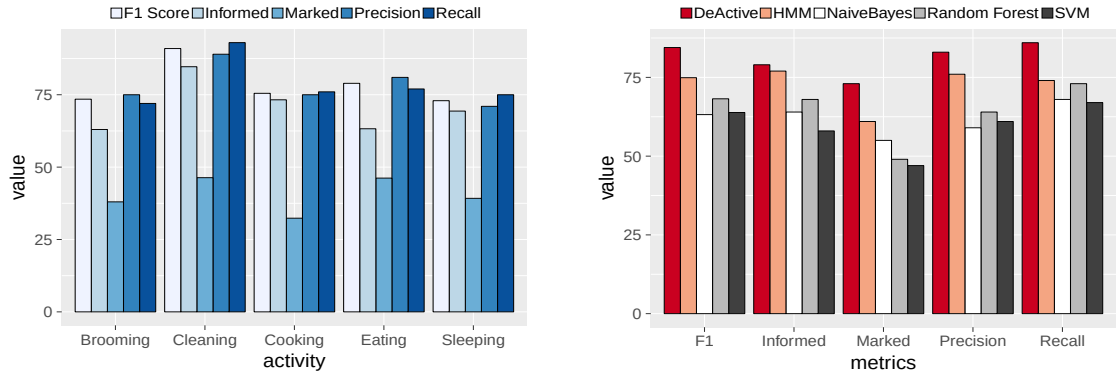
5.9.2 Classification Accuracy

We first evaluate accuracy of different datasets with ambient motion sensor data using our model. We show the Precision, Recall, F1, Informedness and Markedness score of individual datasets in Figure 5.4 and 5.5 . For *Opportunity* dataset (Fig 5.4b), we see that preparing coffee achieved lowest accuracy as these activities involved similar movement using kitchen appliances. For *SenseBox* dataset we experience comparatively low accuracy for cooking, eating sleeping and brooming



(a) CASAS (b) Opportunity

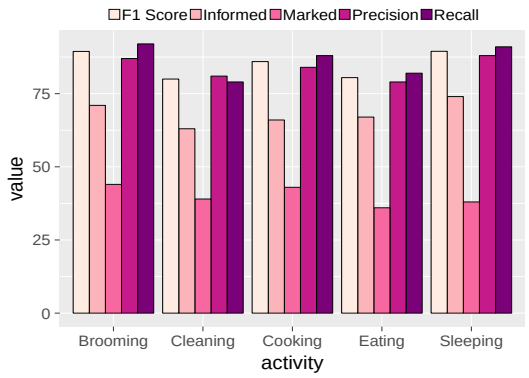
Figure 5.4: Precision, recall, F1, informedness and markedness score of each activity in CASAS and Opportunity datasets (ambient motion sensor data).



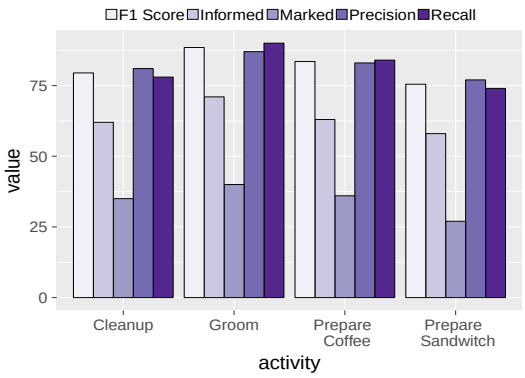
(a) SenseBox (b) Overall

Figure 5.5: (a) Precision, recall, F1, informedness and markedness score of each activity in SenseBox dataset (ambient motion sensor data) and (b) illustrates the accuracy of different shallow learning algorithms compared to our *DeActive* framework.

than cleaning. After further investigation we found that *cooking* activity has a lot of false positives. About 38% time our prediction algorithm predicted *cooking* as

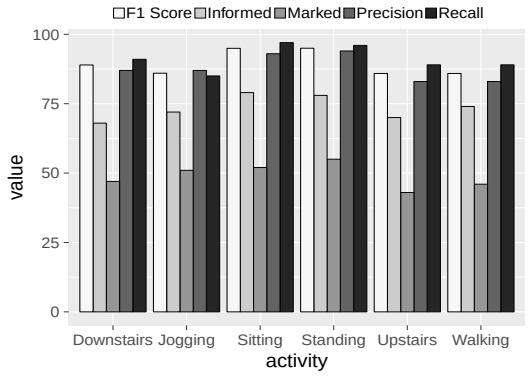


(a) SenseBox

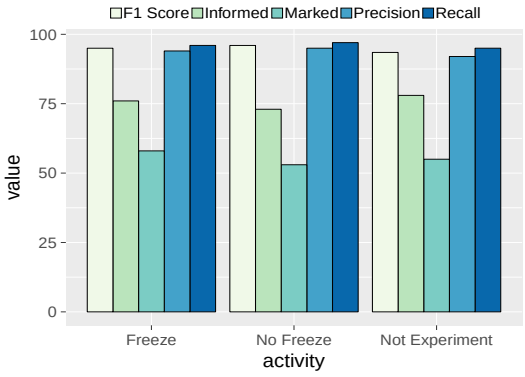


(b) Opportunity

Figure 5.6: Precision, recall, F1, informedness and markedness score of SenseBox and Opportunity datasets (3D acceleration data).



(a) WISDM

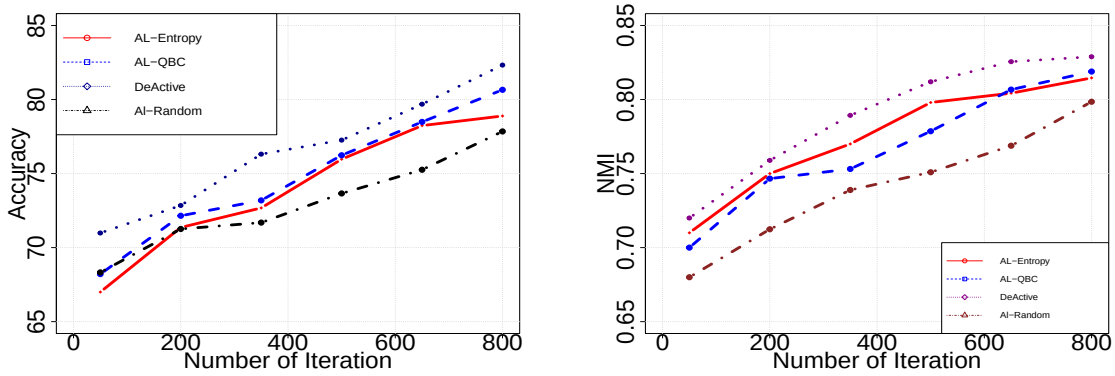


(b) Skoda

Figure 5.7: Precision, recall, F1, informedness and markedness score of WISDM and Skoda datasets (3D acceleration data).

eating and *cleaning*. By reviewing the ground truth information we confirmed that in these cases the participant was in the kitchen but not cooking. The participant sometimes ate in the kitchen and also there are times when he was cleaning the appliances. As a result our model confused these two classes with *cooking* activity.

Eating activity is also hard to detect using just the ambient motion sensor as the participants ate in different locations at times. We faced similar problem as *cooking* activity in this case and majority of the false positives were labeled as *cooking*. Although we have attached an acceleration sensor with the broom to detect the *Brooming* activity but due to mobility in different rooms while brooming it created false positives. Similarly for *CASAS* dataset we received much higher accuracy for all the activities except *Eat*. The line chart in Figure 5.11 shows the convergence of accuracy with respect to the percentage of dataset used in the experiment.



(a) The change in model accuracy after each iteration using different active learning strategies.

(b) The change in NMI after each iteration using different active learning strategies after each iteration.

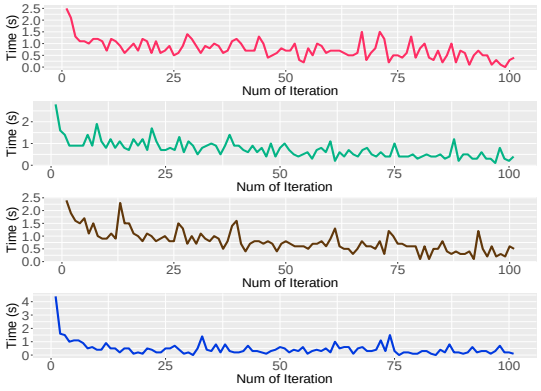
Figure 5.8: The figures demonstrate the performance of our active learning algorithm.

Now we validate our model with 3D acceleration data. In this case we also show the same metrics used in previous experiment for each activity in each dataset in Figure 5.7 and 5.6. Each dataset showed much better accuracy when accelerometers

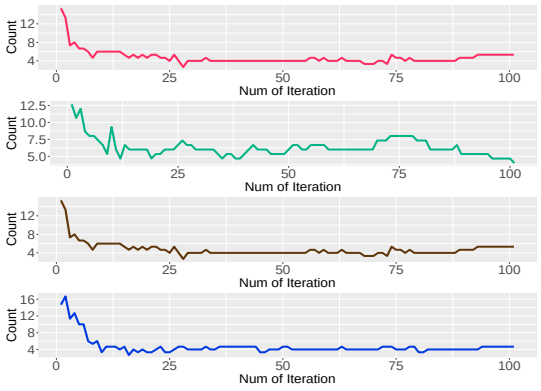
Activity Recognition System	Skoda	Opportunity	WISDM	SenseBox
Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition [79]	95.8	91.20	95.86	88.09
Convolutional Neural Networks for human activity recognition using mobile sensors [203]	88.19	93.17	94.75	84.20
Deep Activity Recognition Models with Triaxial Accelerometers [78]	89.38	86.39	94.46	87.54
Our deep learning framework *DeActive*	92.34	94.06	97.24	92.34

Table 5.2: Comparison of our DeActive algorithm with other existing approaches for different datasets.

were involved. In both $SenseBox$ and $WISDM$ a single accelerometer sensor entity is utilized. In our dataset we employed a wearable device placed on the dominant arm of the participant and for $WISDM$ a smartphone. *Brooming* (Figure 5.6a) achieved much better accuracy than using just the ambient sensors as our deep learning architecture was able to find better feature representation from the acceleration data. In *Opportunity* dataset we experience less accuracy for *Prepare Sandwich* and



(a) Instance selection time of Entropy, QBC, Random and DeActive strategies (up to down) in 100 iterations.



(b) Incorrectly classified instances of Entropy, QBC, Random and DeActive strategies (up to down) in 100 iterations.

Figure 5.9: The figures demonstrate the performance of our active learning algorithm.

Cleanup activity. After further investigation we found that about 31% of *Prepare Sandwich* class was labeled as *Cleanup* and *Prepare Coffee*. These two activities had similar feature representations in our model and as a result the predictor mislabeled them. With *Skoda* data set we have achieved much higher accuracy compared to other datasets as the activities considered had distinct feature representations. In case of *SenseBox* dataset again we achieved low accuracy for *eating* and *cooking* classes. For further validation we looked into our video recordings and found that different participants had different eating behavior. Also the eating pattern largely depends on the cuisine and the type of food you are eating. For example, some foods are eaten using fork and knives (rice, steak vegetables etc.), some using only spoon (soup, stew, chowder etc.) and some using only hand (burger, sandwich etc.).

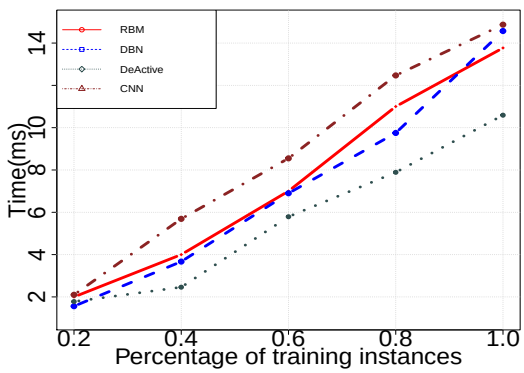


Figure 5.10: The execution time of *DeActive* in *SenseBox* with respect to percentage of data instances.

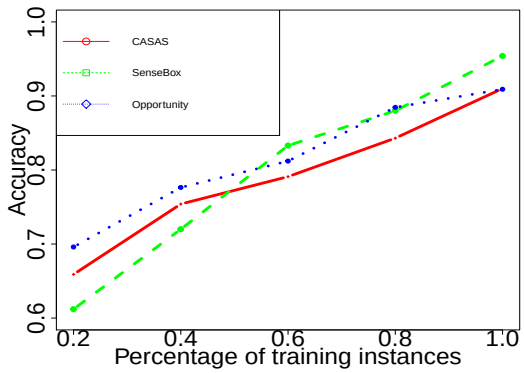


Figure 5.11: Convergence of Accuracy with respect to percentage of data instances.

Due to these variations it was difficult to capture distinguishing feature between different eating movements. For *cooking* activity, we experienced similar challenges due to variations in cooking style. Some of the participants were not spending much time in cooking. Also during cooking, we saw that the participants were doing other activities concurrently like talking over the phone, moving stuffs or watch television etc. As a result we achieved low accuracy for *cooking* activity. For *WISDM* dataset, the overall accuracy was much higher ($\approx$ 92%) than other datasets as the activities considered have distinct signature pattern in the accelerometer data. In Table 6.3 we compare our *DeActive* model with some recent existing activity recognition works which are based on deep learning. Although these state-of-the-art models experimented on different datasets, still we have achieved similar or better accuracy. These models also take more training time and require more resources than our model.

5.9.3 Effect of Active Learning

We applied our active learning algorithm in a 10-fold cross validation manner. We started our active learning experiment with 20,000 unlabeled data instances and randomly selected 1000 labeled instances. We adopt pool based sampling in our experiment. After analyzing the results of silhouette coefficient, we empirically define 0.73 as our threshold. In Figure 5.8a we exhibit the change in model accuracy over 800 iterations. In each iteration we query the most informative 500 data instances and after receiving the label we add it to our training dataset. The instances which are within our predefined threshold (0.73), we also annotate them in accordance with their associated most informative instance. We compare our algorithm with other popular active learning strategies like maximum entropy, Query by Committee(QBC) and random sampling. It is evident from the figure that our active learning strategy outperforms other popular strategies and converges faster. Our model achieves better performance with respect to recognition accuracy after acquiring same percentage of labeled examples.

In Figure 5.8b we show the effect in NMI for our model. From the figure we see that our model is converging to optimal accuracy faster. The higher NMI represents that the assigned label by our classifier and the label from the annotator is getting closer. We also look at how many instances were incorrectly classified in 100 iterations using our active learning algorithm in figure 5.9b. It is noticeable from the figure that our active learning algorithm is more stable in correctly classifying instances compared to other strategies which indicates that only vital instances are

being selected for querying. Another important parameter for evaluating active learning algorithm is to monitor the speed or the time it takes to select instances in each iteration. Average instance selection times for *entropy, QBC, random sampling* and *DeActive* are - 0.76s, 0.73s, 0.80s, 0.468s. *DeActive* is almost 40% faster than other strategies while selecting instances. In figure 5.9a we show the progression of instance selection time for the first 100 iterations for all active learning strategies.

5.9.4 Device Performance

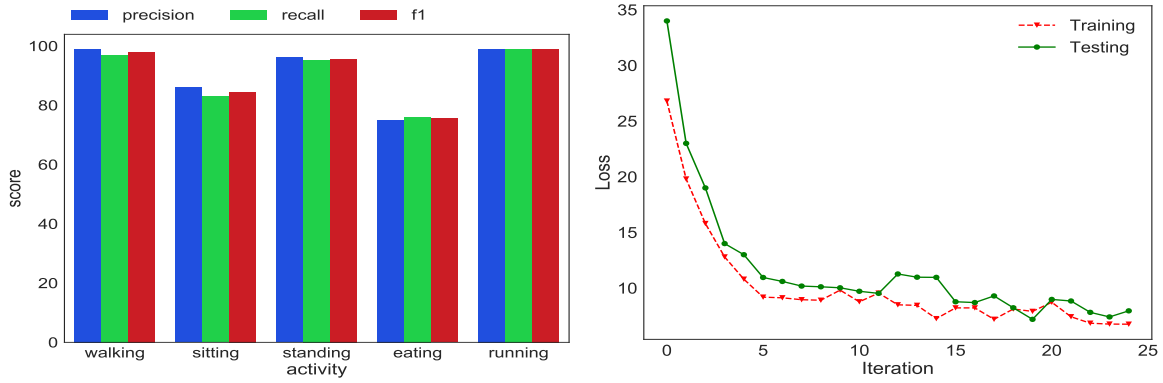
We investigate the performance of our *DeActive* model in *SenseBox* architecture. In Figure 5.10 we see that our algorithm executes much faster than other algorithms. In [64], the execution time is reported as 20.78 msec with 50 hidden layers and 3,289,600 parameters. In our case the execution time is close to 10 msec with 3 hidden layers. However [64] used Snapdragon 400 quad core CPU whereas we used single core CPU.

5.9.5 Deep Active Learner Network

We have employed Convolutional Neural Network (CNN) as our deep model. In a frame we have 64 instances generated from 4 sensors (accelerometer, gyroscope, magnetometer, location), so our input size is 64×11 (9 from 3D sensors and 2 output from the location sensor). Our network has 3 convolution layers with filter size of 5 followed by a max pooling layer with filter size 2 and a fully connected layer. A softmax classifier is used at the final layer for the classification task. We applied

batch normalization after each convolution layer to handle internal covariate shift. We also used dropout as our regularization method at the dense layer. Number of filters in the convolution layers are 32, 64, 128 respectively and network uses ReLU activation function. We set the model parameters $\alpha = 0.005$ and $\beta = 0.00002$. We use adam optimizer with batch size of 32.

5.9.6 Classifier Performance



(a) (b)

Figure 5.12: (a) shows precision, recall and F1-score of our classifier for different activities. (b) the trend of loss function during training and testing.

At first we utilize the whole dataset to train our model to evaluate the performance of our classifier. We split the entire dataset into training (60%), validation(20%) and testing(20%) dataset. We achieved an overall accuracy of 92.05%. The confusion matrix is shown in Table 5.3. We see that we achieved comparatively low accuracy for *eating* and *sitting* activity. Upon further investigation we found that this is due to the nature of these two activities. Most of the time the users

were eating while sitting and some of the time the users were eating while walking and standing as well. Thats majority of the instances of misclassified are labeled as sitting, while some were predicted to be standing and walking. The same problem persists for the *sitting* activity as well. Majority of the mislabeled instances are labeled as eating. However, we have received better accuracy for sitting in compared to eating activity. After looking at the pattern of both of these activities, we noticed that when the users were idly sitting they used their phone quite often and while eating the phone remained idle most of the time. This deviation was captured by the model and hence it achieved better accuracy for sitting activity. The precision, recall and F1-score of our classifier are exhibited in Figure 5.12a. In Figure 5.12b we demonstrate the trend of our loss function during training and testing.

Table 5.3: Confusion Matrix

	Walking	**Eating**	**Running**	**Sitting**	**Standing**
Walking	98.19%	0%	1.12%	0.19%	0.50%
Eating	2%	79%	0%	16.10%	3.9%
Running	0.72%	0%	99.17%	0%	0.11
Sitting	1.01%	11%	0.38%	87.56%	0.05%
Standing	0.02%	2.38%	0.21%	1.03%	96.36%

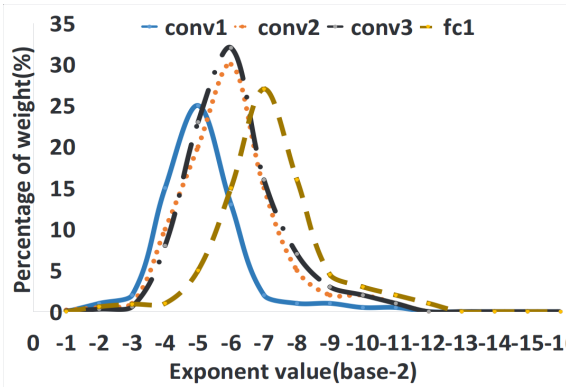


Figure 5.13: Weight distribution in log scale of different layers using cross-entropy loss.

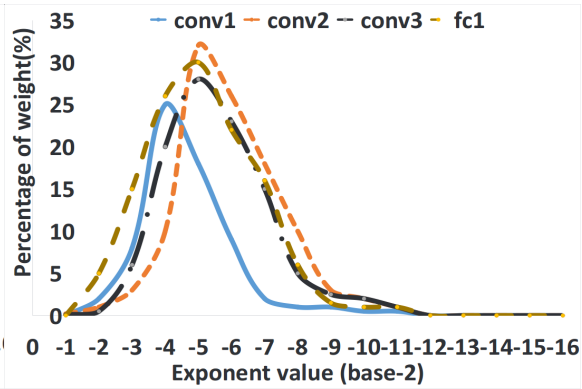


Figure 5.14: Weight distribution in log scale of different layers using our joint loss function.

We analyze the performance of our model while employing active learning. For this we train our model with only 30% labeled instances from our dataset. In order to train our joint optimization function we also fed same number of unlabeled instances to be consistent with batch size during training. We have used 15% of our unlabeled dataset for this purpose. Initially we have achieved 83.26% accuracy with the provided labeled and unlabeled instances. In order to see the effect of our joint optimization we examined the weight distribution in different layers of our network. In Figure 5.13 and 5.14 we plot the value of the weight distribution in log scale while optimizing only cross-entropy loss and our joint loss. It is evident from the figure that our joint loss function enabled lower values for the individual layer weights compared to traditional cross-entropy loss function. When the weights are large, the layers are more sensitive to small noises in the input data. This had significant effect when we were doing entropy calculation for the unlabeled data instances. The network

produced much different value in the output layer due to the larger weight which eventually enabled higher number of queries that included outliers. We randomly sampled 100 unlabeled instances from the holdout unlabeled data instance pool and calculated entropy for both network. We applied this over 10,000 iterations and calculated the average number of queries received for both these loss functions. We received on average 29 and 8 queries from the network with cross-entropy loss and the network with our joint loss respectively. The difference in number of informative data instances selected is significant for both these networks. This poses a problem as the queries begin to pile up drastically for individual annotators. By doing joint optimization we were able to mitigate the effect of noisy instances.

Chapter 6: Annotator Selection

In this chapter we discuss two annotator selection model in conjunction with active learning by exploiting the social relationships among the activity witnesses. We discuss active learning, the contextual multi armed bandit problem and the modeling of arms or actions of the bandit, the rewards and the context of our problem domain and a reinforcement learning based annotator selection model.

6.1 Architecture

Our annotator selection framework, *SocialAnnotator* is composed of three major components. We have an activity recognition classifier which is trained on labeled data instances from wearable devices that provide raw accelerometer data. After building a stable classifier, we start feeding unlabeled instances and predict the class label. Let us consider the output of the classifier are class probabilities. The class with highest probability or likelihood is the final prediction. However, if the range of values does not fall within a pre-defined threshold then we assume that the classifier is uncertain about which class the data instance belongs to. If labels are provided for these kinds of instances, it may help improve the efficiency of our classifier. In order to filter these instances we send the unlabeled data instances in

a pool to our *Active Learner* module. In our active learner module, we measure the entropy of the instances and select an instance with maximum entropy. We then send the selected instance to our *Annotator Selection* module. Note that in our daily life we interact with a number of people. The interaction can be physical or virtual through social network but every interaction is an opportunity to observe and share information. The key insight here is that we are connected and have more interactions with the people who we are related with us. These connected people might be direct witnesses of what we are doing in our day to day life. As a result these social relationships and correspondence lead us to have knowledge about the activity patterns of the people we are connected with. We get the direct information about their location and any activity performed by them in their work place from the ontology. Using this information, we try to extract the hidden facts whether they can be direct witnesses of each others activity.

We calculate the *spatio-temporal* distance between two connected users using their probability distribution of location. The *spatio-temporal* distance lets us know about the intersection between their location distribution. We also incorporate a weight based on the strength of the relationship. The people with whom we interact more have higher potential to know about our daily routines. After formulating the distance parameter, we model a budget constrained context aware multi-armed bandit. The task of the bandit is to select annotator given the distance parameter and context. We design the bandit in such a way so that it does not act in a greedy way by introducing costs associated with each annotator and a budget constraint. We adapt a game theoretic approach where we have to ensure maximum gain and

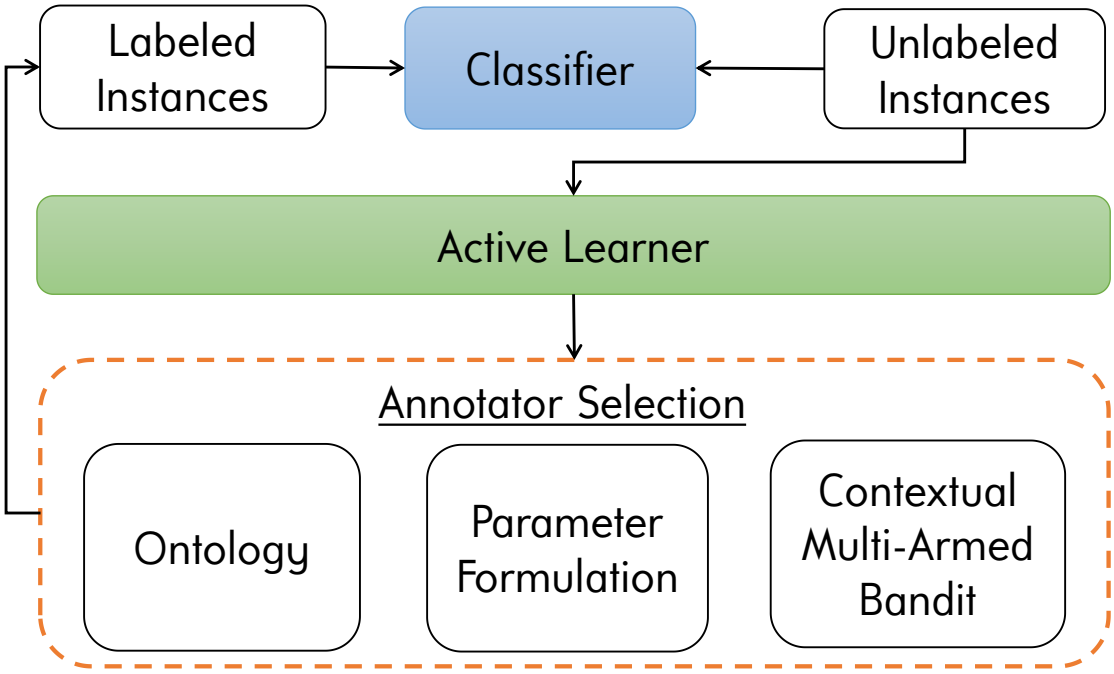


Figure 6.1: A high level structure of the modules in SocialAnnotator framework. Based on classifier's feedback, a pool of unlabeled instances are supplied to the Active Learner which then filters out the data instance with maximum uncertainty. The Annotator Selection module receives the instance and based on the context information it poses the query to the selected annotator. Upon receiving the label it adds the instance to the labeled dataset.

keep track of our budget as well. The costs associated with the annotators are not static, as the level of interaction evolves over time. For example, we have regular interactions with the people at our work place during the week days, but over the weekend we tend to mingle with close friends. Also a person can be connected through multiple relationships (close friend and colleague in a work place at the same time). Therefore, we consider the cumulative relationship weights of all the relations for quantifying the level of correspondence between two users. Figure 6.1 depicts a conceptual structure of our *SocialAnnotator* framework.

In our second approach, instead of bounding our reward values, we adopt a functional approximation approach using deep reinforcement learning. We extend our deep active learner architecture by placing our annotator selection module in Figure 6.2. Our proposed model is comprised of two interlinked networks - actor/action and critic/policy network. The goal of the actor network is to predict the optimum annotator given the current context of the user. Given the current context and the candidate annotator selected from the actor network, the critic network tries to reach the optimum policy and approximates the reward gained by choosing the candidate annotator. The reason we adapt an actor-critic model is because of our continuous state space. In traditional Q-learning approach, we have a finite set of actions and states for which we calculate the reward received for every possible combinations. However, if one of them among action and state space is continuous then the traditional reward calculation becomes intractable. As a result we model our annotator selection component as a functional approximation problem. After receiving the feedback from the selected annotator we add the newly received label

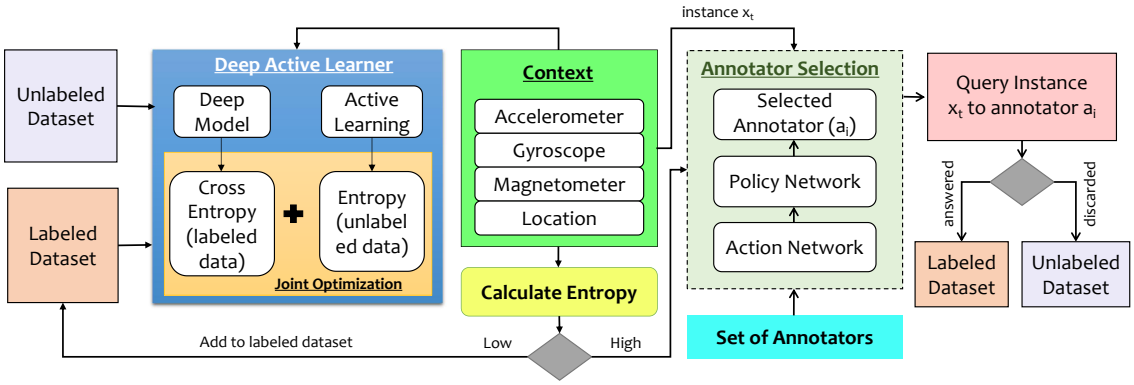


Figure 6.2: The left side of the figure illustrates our active learning enabled deep model and in the right side of the figure our annotator selection pipeline is shown.

to our labeled data set. We re-train our model after a certain interval with the expanded labeled dataset. If an annotator refuses to answer the query and discards it, then it is added back to the unlabeled data instance pool.

6.2 Distance Metrics

In this section we discuss the metrics which correlate our annotators with activities.

6.2.1 Spatio Temporal Distance

We calculate the spatio-temporal distance of the related users when an activity is performed. This distance metric implies if an user has any knowledge about the performed activity by another user he is related to. While computing this parameter, we also consider the neighboring locations. We calculate the likelihood of each related user j being in a location l_i given the current context x_t. We process this as

a *categorical distribution*. Let us consider a set of activities W with whom the user u_i is connected. The location of each activity is an observation of our distribution, and the location set $L(W)$ is a sample of that distribution with cardinality m. Each location in $l_i \in L(P)$ has a prior probability. We denote the probabilities of locations as vector $p = (p_1, p_2, p_3...p_m)$. Let us consider q be the location probability distribution of an annotator a_i who is connected to user u_i through a relation. We then calculate the conditional distributions of p and q given context x_t and time t. Using these conditional distributions we calculate the distance between them using *Bhattacharyya distance* [204]. The distance between these two conditional distributions is defined as:

$$\begin{aligned} d_{st}(p(x,t), q(x,t)) &= -ln(\mathbb{B}(p(x), q(x))) \\ &= \sum_{i=1}^{m} \sqrt{p_i(x,t), q_i(x,t)} \end{aligned} \tag{6.1}$$

In Eqn 6.1, $\mathbb{B}$ is the *Bhattacharyya coefficient* which provides the measurement of overlap between the two probability distributions. This distance provides us information regarding the annotators who reside closer to the user. We calculate this spatio-temporal distance for all the connected users and take the annotators who were closest to the vicinity. If no annotator was present in the vicinity where the user performed the activity, we assume that annotators dwelling in the neighboring locations may have knowledge about the activity label. In such cases we consider the distance of the neighboring locations which are inferred using the *neighbor Of* relation from the ontology.

6.2.2 Activity-Activity Distance

We exploit the connectivity among activities to filter appropriate annotators. Our intuition is that if the properties of an activity W_i prevails in a similar spatial and temporal space to another activity W_j and an annotator a_k has efficiently provided reliable labels to activity W_j then a_k is a potential annotator who can provide label of activity W_i. To calculate this distance we consider three components of an activity pair - *correlation*, *spatial* and *temporal*. Correlation calculates the co-occurrence frequency of the activity pair, the spatial and temporal component models the probability of an activity pertaining to the same location and time constraints. The distance is defined as:

$$d(w_i, w_j) = f(w_i, w_j)\,\mathcal{N}(||t_{a_i} - t_{a_j}||^2, \mu_t, \sigma_t)$$
$$\mathcal{N}(||l_{a_i} - l_{a_j}||^2, \mu_s, \sigma_s) \tag{6.2}$$

In eqn 6.2, $f(w_i, w_j)$ denotes the co-occurrence frequency between a pair of activity, $l_{a_i}, l_{a_j}, t_{a_i}, t_{a_j}$ are the spatial and temporal parameters of the associated activities.

6.2.3 Relationship Weight

The strength of the social relationship can be integral in selecting annotator. There may not be any annotator who directly witnessed the user doing an activity. However, human being follows a cognitive routine most of the time and the persons

mostly associated with his life are acquainted with the routine. For example, the family members living with the user are usually more familiar with his routine. Some annotators can also be remotely connected (e.g. updates on social network, talking over the phone or even playing online games together). So certain relationships provide more emphasis and demand more attention while choosing the annotator. For this reason, we try to provide weight to each connected user according to the relation. However this weight can not be static for all of the users as in real life not all relationships are same and they evolve over time. For example, consider the relationship with your office colleagues, initially they could be just colleagues but over time some might become your close friend. On the other hand one might be in touch with their parents on regular basis, but a different person might not. So for each person the weight of relationship is different. We use the relationship intensity strength proposed in [205] to model our relationship weight. The interaction between two users (e.g. phone call, messaging, meeting etc.) or shared information (e.g. playing soccer together, common hobby) are designated as "rate factor". Depending on the social aspect these rate factors regulates the strength of a relationship. The partial relationship weight between user k and j for one factor is defined as:

$$W_f(k,j) = \frac{\omega_{kj} \sum_{i=1}^{l} f_t}{1 + ln(1 + l_c)} \tag{6.3}$$

In eqn 6.3 ω_{kj} is the weight of the rate factor, l is the count of rate factors, l_c is the count of instances of the rate factor and f_t models the time influence. The final weight $W(k,j)$ is measured by taking the arithmetic mean of the partial weights of

all the rate factors.

Now that we have formulated all our distance metrics, we now define our final user to user distance metric. The activity-activity distance metric provides the distance between activities and finds the similarity among them. We maintain the count of such activities for which the annotators have performance score more than a pre-defined threshold δ. We utilize this count as an additional weight W_c for the annotators. The final distance is calculated using the following equation:

$$D(k,j) = W(k,j)\, W_c\, d_{kj}\Big(p(x,t), q(x,t)\Big) \tag{6.4}$$

6.3 Background & Methodology

In this section, we setup the annotator selection problem investigated in the thesis. We discuss the preliminaries of both multi-armed bandit and reinforcement learning algorithms.

6.3.1 Active Learning

Conventional supervised learning algorithms compile models based on the available patterns in the provided training dataset. The level of sophistication for these models largely depends on both quantity and quality of the provided labeled instances. Active learning takes a different approach by making the learning process ongoing and interactive. It also helps to relieve the pressure of collecting large amount of labeled data instances. Active learning is fitting for problems pertaining

to large amount of unlabeled data instances. In the context of activity recognition using wearable devices, we have to process overwhelming number of data instances which makes active learning befitting. We only label the data instances which provide highest gain which is reducing the generalized error of our classifier. In our proposed model we propose to use Active learning using pool-based sampling as we receive a stream of data in a very short period of time. We select a data instance from a pool of instances in a greedy way Queries are typically conforming to the measure of uncertainty. Here our assumption is that the instances which are least certain are close to the decision boundary and labeling these instances will provide maximum gain. To measure the uncertainty we calculate the entropy of the provided instances and query the instance with maximum entropy. We calculate the maximum entropy and select an instance by following equation

$$\begin{aligned} x_H &= \underset{x}{\operatorname{argmax}}\, H_\theta(Y|x) \\ &= \underset{x}{\operatorname{argmax}} - \sum_y P_\theta(y|x) log\, P_{theta}(y|x) \end{aligned} \tag{6.5}$$

6.3.2 Contextual Multi-Armed Bandit

A contextual bandit problem is composed of N arms or actions. In our context an action refers to selecting an annotator. On each iteration, based on the revealed context the learner decides one action to choose and observes the associated reward of *only the action applied*. As the rewards corresponding to other actions remain hidden from the learner so the learner receives only partial information about the

reward space. The goal is to maximize this reward in each iteration. However, by selecting a sub set of the actions in a regular manner might always provide maximum reward. For example, a person's spouse or close friend has better idea about his daily activity routine than any one else. So by selecting the spouse or close friend in each round will maximize the reward outcome. If we consider the annotators as resources, prompting the same set of annotators will lead to resource exhaustion. In order to tackle this, we introduce a resource constraint or budget for each of the annotator. The annotators who ensures higher potential reward, incur higher cost. As a result given an overall budget our aim is to maximize the total reward while ensuring aggregated resource consumption remains bounded by a given budget.

Let us denote the action set as $\mathcal{A} = \{a_1, \cdots, a_k\}$. We consider the cardinality of $\mathcal{A}$ to be finite as an user is connected to a finite number of people. A d-dimensional feature vector $x_t \in \mathcal{X}$ denotes the context information received at time t. At each time t, an agent or policy π decides to choose an action a_i based on the context x_t and receives reward r_i^t. The history of taken actions and received reward is denoted by $\mathcal{H}_{t-1} \rightarrow \{a_i(\tau), r_\tau, x_i(\tau)\}$ for $i = 1, \ldots, N$ and $\tau = 1, ..., t-1$ where $a_i(\tau)$ denotes the chosen action which generated reward r_τ. The reward of an action is generated from an unknown distribution regulated by the given context. Let us consider the optimum action at t is a_i^* and its corresponding reward is $\tilde{r}_i^t$.

We want to select the action which results in reward close to the optimum one, so the aim is to maximize the reward in each step and minimize the difference between the overall optimum reward and the reward received. The difference between the optimal reward and the aggregated reward received is called *regret*. We provide

a formal definition of *regret* as following

$$\mathbb{R} = \sum_{t=1}^{T} \mathcal{R}_t = \sum_{t=1}^{T} (\tilde{r}_i^t - r_i^t) \tag{6.6}$$

In this eqn, $\tilde{r}_i^t$ is the optimum reward at step t and r_i^t is the reward received. Let us define our reward function as $r_t = f(x_t, a_i(t))$, where $f(x_t, a_i(t))$ is the reward mapping function for arm $a_i(t)$. In general the reward mapping function using context and action is defined as $f(x_t, a_i(t)) = x_t^T w_i + \epsilon_i$ where x_t^T is the transpose of the contextual information, w_i is a coefficient vector and ϵ_i is drawn from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. The weight w_i for each annotator is projected independently based on the received reward from each user. However in our case, there is collective influence among users.

In order to maximize the reward function, the agent needs to learn the underlying function f which maps the context to action. In order to acquire knowledge about the latent function f, the agent has to explore other actions instead of choosing the optimum action which provides the best outcome. ϵ is our exploration parameter. The predictive distribution of our reward function depends on the current context and the history of actions taken. This is a normal distribution with mean μ_r and variance V and defined as following

$$p_\theta(r_t | \mathcal{H}_{T-1}, x_t) = \mathcal{N}(\mu_r(t), V_t) \tag{6.7}$$

Each action $a_i(t)$ is also associated with a cost $c_{a_i}^t$. The cost associated with an annotator is variable in each round as the distance between users defined in eqn 6.4 varies over time. The costs are independently and identically drawn from an unknown continuous distribution with mean μ_c. We adhere to the same settings

in [206]: (i) the rewards of an action are independent of its costs (ii) the rewards and costs of an arm are not influenced by other actions (iii) the rewards and costs of an action are independent and identically distributed at each iteration. Let us define a known parameter, *budget* B which designates the number of time the algorithm can invoke annotators. This budget constrain also helps us to supervise the stopping time $t_s(B)$ of our algorithm which is defined as following

$$\sum_{i=1}^{t_s(B)} c_{a_i}^i \leq B < \sum_{i=1}^{t_s(B)+1} c_{a_i}^i \tag{6.8}$$

Let us denote $\mathcal{R}^*$ as our optimum aggregated reward at stopping time $t_s(B)$. We calculate the expected regret, evaluated over the randomness of rewards and costs by modifying eqn 6.6.

$$\mathbb{R} = \mathcal{R}^* - E\Big[\sum_{t=1}^{t_s(B)} (r_{a_t}^t)\Big] \tag{6.9}$$

6.3.2.0.1 Actions

The action space for an user is proportional to the number of connected annotators in the ontology. An action corresponds to selecting an annotator from the correspondence vector M built using the ontology of annotator relationship. Each element $m_{ij} \geq 0$ is congruent to how relevant the annotator is with respect to the user in terms of our distance metric and labeling accuracy. The expected reward to cost ratio of an annotator a_i is $\rho_{a_i} = \frac{\mu_r^{a_i}}{\mu_c^{a_i}}$. According to [206], if both reward and cost distribution of an action is known, pulling the arm with maximum ρ can provide the

expected reward as the optimal algorithm. When the distributions are unknown, we should select the annotator with the maximum ρ and also ensure exploration on the other rarely selected annotators.

6.3.2.0.2 Context

A context vector x_t portrays the features and characteristics of each annotator. The features considered in a context vector are the timestamp t, location s, n performance metrics of the annotator with respect to each activity $p_1, \ldots, p_n$. We do not include the sensor data in the context vector.

6.3.2.0.3 Reward

Our reward mapping function randomly generates reward according to the conditional probability measure defined in eqn 6.7. Initially the model is uncertain about the value θ.

Our reward mapping function f is defined to measure the reduction in variance of our classification model between two iterations. For making things simple, our objective is to minimize the squared loss of the true label and the label received from an annotator. We define our expected error as following

$$\begin{aligned}\mathbb{E}\Big[(\hat{y}-y)^2|x_t, y_l\Big] &= \mathbb{E}_{Y|x}\Big[(y-\mathbb{E}_{Y|x}[y|x,y_l])^2\Big] \\ &\quad + (E_L[\hat{y}]-\mathbb{E}_{Y|x}[y|x])^2 + E_L\Big[(\hat{y}-E_L[\hat{y}])^2\Big] \qquad (6.10)\end{aligned}$$

In eqn 6.10 $\mathbb{E}_L[.]$ is the expectation over the labeled training set L, $\hat{y}$ is the

label received from an annotator and y is the true label of the instance. $\mathbb{E}_{Y|x}\Big[(y - \mathbb{E}_{Y|x}[y|x])^2\Big]$ indicates noise or uncertainty of y given x. The second term represents bias which is the error due to the selected action. The third term represents the output variance of our model. Therefore minimizing the variance will ensure to minimize the generalization error of our model. So we try to reduce error by selecting annotators that establish highest variance reduction of our activity recognition model. For any action a_i, number of times it is invoked $n_{a_i,t}$, average cost $\bar{c}_{a_i,t}$ and average reward $\bar{r}_{a_i,t}$ and the exploration parameter is $\epsilon_{a_i,t} = \sqrt{\frac{2log(t-1)}{n_{i,t}}}$. We calculate index $D_{a_i,t}$ for each annotator:

$$J_{a_i,t} = \frac{\bar{r}_{a_i,t}}{\bar{c}_{a_i,t}} + \frac{\bar{r}_{\epsilon_i,t}}{\bar{c}_{a_i,t}} + \frac{\bar{r}_{\epsilon_i,t}}{\bar{c}_{a_i,t}} D(k,i) \tag{6.11}$$

In eqn 6.11, the average reward to cost ratio represents the exploitation. The first influences our algorithm to choose the arms with higher rewards. The exploration term $\frac{\bar{r}_{\epsilon_i,t}}{\bar{c}_{a_i,t}}$ favors the annotators who provide less reward and as a result invoked infrequently with lower costs. Exploring weaker annotators may be conducive as our budget is limited. The final term enforces joint exploitation and exploration.

6.3.3 Reinforcement Learning Preliminaries

Majority of the real-world problems involving reinforcement learning are modeled in accordance to Markov decision processes (MDP) [207]. By modeling through MDPs, we can formalize the decision-making process by considering not only the reward of the immediate action but also the outcome of the consecutive actions. We

consider an agent interacting with the environment E in discrete timesteps t. At each time step t, the agent observes a representation of the environment $s_t \in S$ and takes an action a_i from a set of actions A namely selects an annotator and receives a reward $r_{t+1} \in \mathcal{R} \subset \mathbb{R}$. In our context, the set of actions or annotators and rewards are finite as we have a countable number of annotators who are connected with a specific user. The discrete probability distributions of random variables a_i and r_t depend only on the prior state and action. The process of choosing action given the state of the environment and receiving reward continues until we reach a terminal state. However, in our problem domain there is no terminal state as the process of annotation will never finish. A state s_t is defined by the data we are receiving from the sensor modalities. As the sources of our data stream provide continuous value, hence our state space is also continuous meaning the probability density of the next state is continuous in the action taken at the current state. The state-transition probability function is defined as $\int_{s'} T(s, a, s')ds' = P(s_{t+1} = s'|s_t = s, a_i = a)$

A policy π defines the behavior of an agent which outlines states to a probability distribution over the annotators $\pi : S \rightarrow P(A)$. The primary goal of any RL agent is to maximize discounted cumulative future reward or *return*, $G_t = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$ from a state at given any given time t where γ is the *discount rate* and $0 \leq \gamma \leq 1$. The value function $v_\pi(s)$ of a state s under policy π is the expected return starting from that state which provides insight about how good a state is. The formal definition of this $state - value$ function is defined by

$$v_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|s_t = s] \tag{6.12}$$

Similarly the *action-value* $q_\pi(s, a)$ function for policy π taking action a while being in state s provides insight about how good a state-action pair is. It is defined by the following:

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi[G_t | s_t = s, a_t = a] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s, a_t = a] \end{aligned} \tag{6.13}$$

The optimal *Bellman equation* for both these value functions are then defined as

$$\begin{aligned} v_*(s_t) &= \max_\pi v_\pi(s) \\ &= \max_{a_i} \sum_{s_{t+1}, r} p(s_{t+1}, r | s_t, a_i)[r + \gamma v_*(s_{t+1})] \end{aligned} \tag{6.14}$$

$$\begin{aligned} q_*(s, a) &= \mathbb{E}[r_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a') | s_t = s, a_t = a] \\ &= \sum_{s_{t+1}, r | s_t, a} [r + \gamma \max_{a'} q_*(s_{t+1}, a')] \end{aligned} \tag{6.15}$$

We can solve these value functions using value iteration algorithm by bootstrapping the feedback received to attain optimum policy and action at each step iteratively. As our state space is continuous, it is impossible to calculate the value function for every state-action pair. Therefore, it is difficult to evaluate the value functions in Eqn 6.15 and 6.14. Our goal is then to find a good approximate solution given the circumstantial prior encounters.

6.3.4 Deep Reinforcement Learning

The deep deterministic policy gradient (DDPG) [208] method based on deterministic policy gradient (DPG) [209] can adapt to domains with stochastic continuous state transitions. Instead of learning the value functions directly DDPG aims to approximate and improve both the policy and action value functions. The policy π is parameterized by θ_π that directly models the action probabilities by tweaking the parameters θ at each time step t. The action-value function $q(s, a; \theta_q)$ is parameterized by θ_q. We train two separate neural network for these value functions where the policy update network is called the *actor* network and the action-value update network is called the *critic* network. The critic network evaluates the action-value function by minimizing the following loss function

$$L(\theta_q) = \mathbb{E}_{s,a\sim\rho(.)}[(y_i - q(s_t, a_t; \theta_q^i)^2] + w_{\theta_q^i}{\theta_q^i}^2 \tag{6.16}$$

$$y_i = \mathbb{E}[r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a'; \theta_q^{i-1} | s_t, a)] \tag{6.17}$$

The parameters θ_q are then updated using the batch stochastic gradient descent method in Eqn. 6.18 where $\nabla_{\theta_q} L(\theta_q)$ is the derivative of the loss function with respect to θ_q and α_q is the learning rate for the critic network.

$$\theta_q \rightarrow \theta_q - \alpha_q \nabla_{\theta_q} L(\theta_q) \tag{6.18}$$

We define the quality of our policy π as the average rate of reward. So we need to update the *actor* network by calculating the policy gradient of the rate of reward.

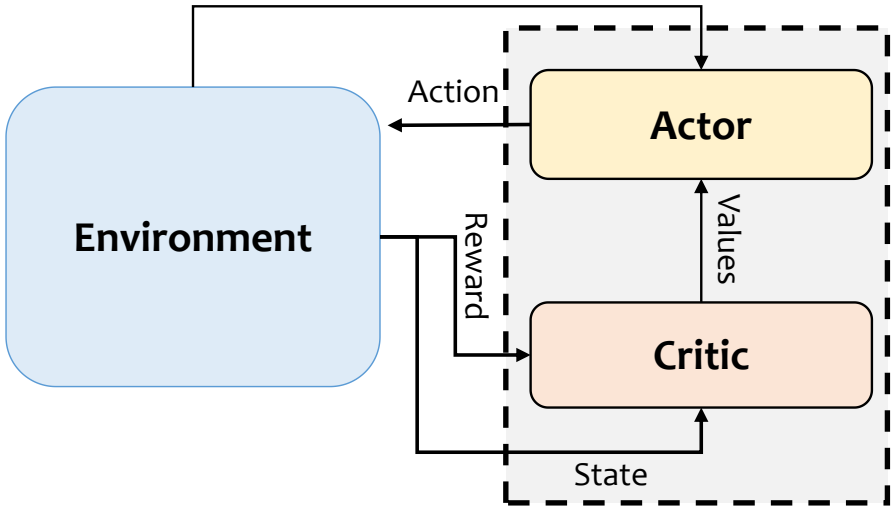


Figure 6.3: Actor critic network flow

The average reward received is defined as:

$$\begin{aligned} J(\theta_\pi) &= \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[r_t | s_0, a_{0:t-1} \sim \pi] \\ &= \lim_{t\to\infty} \mathbb{E}[r_t | s_0, a_{0:t-1} \sim \pi] \\ &= \sum_s \mu_\pi(s) \sum_a \pi(a|s_t, \theta_\pi) \sum p(s_{t+1}, r|s_t, a) r \end{aligned} \tag{6.19}$$

In Eqn 6.19 s_0 is our initial state and μ_π is the steady-state distribution under policy π, $\mu_\pi(s) = \lim_{t\to\infty} Pr(s_t = s | a_{0:t} \sim \pi)$. The *actor* network is updated using batch gradient descent as well using the gradient of the policy gradient function $J(\theta_\pi)$ with the step size parameter α_q.

$$\theta_\pi \to \theta_\pi - \alpha_\pi \nabla_{\theta_\pi} L(\theta_\pi) \tag{6.20}$$

Our annotator selection model is inspired by the actor-critic based reinforcement learning method. The flow of information and the training process is shown in figure 6.3. Taking an action is analogous to selecting an annotator from a finite set

Algorithm 8 Annotator Selection Policy using DDPG

1: **Input:** activity fragments $F_k, k = 1 \dots K$ with associated annotator fragments (s_t, a_i^t)

2: **Input:** parameterized policy $\pi(a_i^t | s_t, \theta_\pi)$

3: **Input:** parameterized state-value function $v(s_t, \theta_q)$

4: initialize state-value parameter θ_q and policy parameter θ_π to 0

5: initialize target function $\hat{\theta}_q \leftarrow \theta_q$ and $\hat{\theta}_\pi \leftarrow \theta_\pi$

6: initialize starting state s_0

7: **for** each time step t **do**

8: Select annotator a_j^t using current policy π

9: Evaluate state-value function $q(s_t, a_j^t, \theta_q)$

10: Evaluate expected state-value y_i using optimum annotator a_i^t

11: Calculate the loss of *critic* network $L(\theta_q)$

12: Calculate the average reward rate $J(\theta_\pi)$ for state s_t

13: Evaluate the gradients $\nabla L(\theta_q)$ and $\nabla J(\theta_\pi)$

14: Update θ_q and θ_π

15: $\hat{\theta_q} \leftarrow (1 - \eta)\hat{\theta_q} + \eta\hat{\theta_q}$

16: $\hat{\theta_\pi} \leftarrow (1 - \eta)\hat{\theta_\pi} + \eta\hat{\theta_\pi}$

17: Update state $s_t \leftarrow s_{t+1}$

18: **end for**

of annotators A. At any given time t a state is defined as a tuple , $s_t = \{d_t, m_t, g_t, l_t\}$ where d_t, m_t and g_t represents three dimensional accelerometer, magnetometer and gyroscope data and l_t is the two dimensional location information of the user. The *actor* network is updated using the gradient of the policy gradient function defined in Eqn. 6.19. The gradient is defined as following:

$$\begin{aligned}\nabla_\pi J(\theta_\pi) &= \nabla_\pi \Big[\sum_s \mu_\pi(s) \sum_a \pi(a|s_t, \theta_\pi) \sum p(s_{t+1}, r|s_t, a) r \Big] \\ &= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) \end{aligned} \tag{6.21}$$

While training the $actor - critic$ networks, the loss function defined Eqn. 6.17 and the gradient calculation in 6.21 is calculated over a minibatch size of 32 samples. Also we use Adam optimizer to ensure efficient learning over directly applying gradient update using Eqn. 6.18 and 6.20. We exploit the method proposed by [208] to fix the problem of divergence when directly setting the parameters of the target function in Eqn. 6.17 at every step. We record the update of θ_q and θ_π by calculating moving average $\hat{\theta} \leftarrow (1 - \eta)\hat{\theta} + \eta\hat{\theta}$ where $\hat{\theta}$ represents θ_q and θ_π and $\eta = 0.001$ is the decay factor.

6.4 Experimental Evaluation

In this section we evaluate our proposed model in details by discussing our experimental setup, data collection process, network architecture and our evaluation methodologies. We focus on addressing the following questions:

- What is the effect of joint optimization while fusing deep learning and active learning methods?
- Is the system able to generalize for the other users for both activity classification and active learning method?
- How does the performance vary with the unlabeled dataset used in the training phase?
- What is the performance of the system in terms of calculating most informative data instances? How effective the model is while handling outliers?
- How many queries are generated on average for a user in a day? How hard were the queries for the users?
- What are the performance scores of the respective annotators?
- Is the feedback received from the annotators improving the performance of the model?

6.4.1 Setup

We collected activity data using wearable devices from 5 users over the course of 16 days. We used android smart watch Moto360 to collect the accelerometer data. We also collected the location information of the users using GPS which we only used for ground truth. We have implemented an app in both android and iOS platform (`http://mpsc.umbc.edu/sajjad/socialannotator/`) to collect mobile sensor data and pose query to the users . A picture of our system running

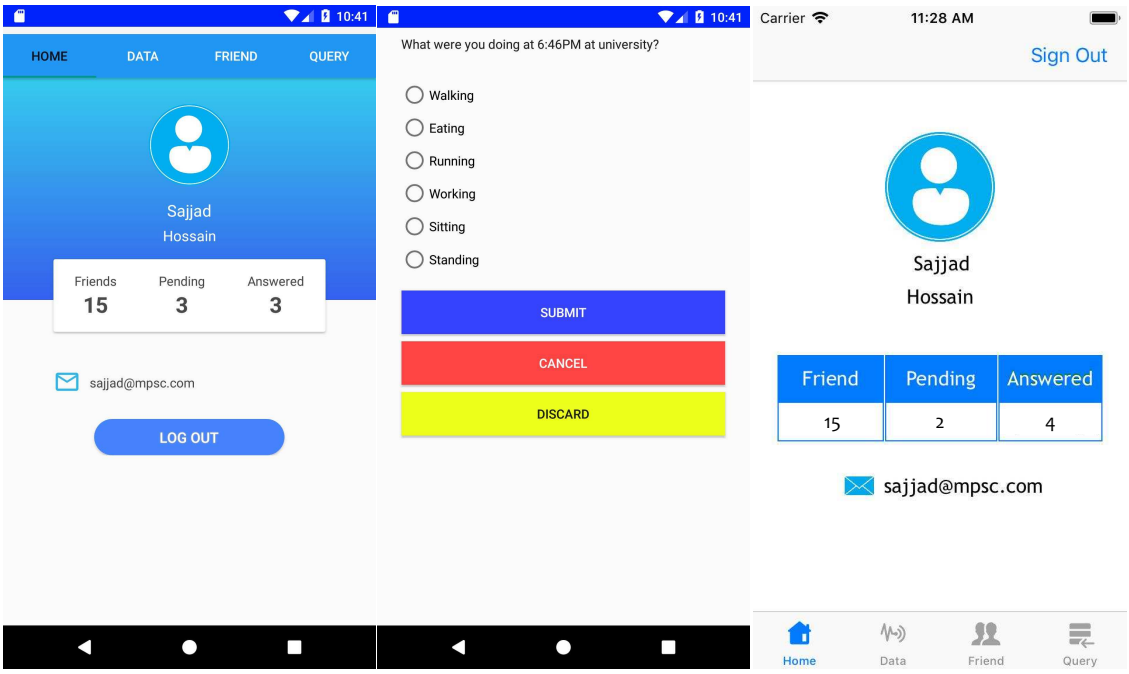


Figure 6.4: SocialAnnotator application interface.

on both the platforms are shown in Figure 6.4. Our app exploits the three dimensional accelerometer, gyroscope, magnetometer and two dimensional location sensors. The users can control which sensor to turn on and off based on their preference of the battery status. Individual users can create account and add potential annotators who are referred as "Friend" in the application. While adding a friend to the annotator list, we also prompt the user to mention the relationship with the annotator from a list of pre-defined relationships - {*Parent, Child, Sibling, Friend, Colleague, Neighbor, Spouse*}. The users can also tag locations with a semantic name like -"work", "home" etc. so that the location information can be displayed in the query. Instead of providing notification to the users whenever a new query arrives, we queue the new query in the *Query* tab. A user can then view the queries at his suitable time and provide feedback to the queries or discard them if he/she is

unsure about the potential activity information. The back-end is developed using MongoDB and python-flask based web service which handles all the requests. The deep active learner classifier and the annotator selection model are implemented using tensorflow. A web-service is deployed using the trained model which interacts with the request handler service. In our experiment we monitored 5 daily activities - {*eating, sleeping, phone calling, working, cooking*}.

The sensor data are directly uploaded to our lab server from the wearable device and we preprocess(feature extraction, filtering, noise reduction etc.) the data in the server. As previously stated in our SocialAnnotator pipeline in Figure 6.1, we train a supervised classifier first to recognize the performed activity. We have used a simple decision tree based classifier. Initially after training our model with labeled instances we achieved an average accuracy of 77%. Even if we have achieved low accuracy compared to the existing literature, we are only concerned about investigating how efficient labeling can help to improve the performance of activity recognition model. We have For our budgeted multi-armed bandit, the reward and cost of each annotator are sampled from a beta distribution. The parameters of the these distributions are sampled from [1,5]. The budget of our framework is chosen from the set {200, 300, 400, 500, 1000}. We compare our annotator selection model with different contextual multi armed bandit algorithm - LinUCB, ϵ-Greedy, EXP4 and Random sampling. In case of Random sampling, the annotator is chosen at random in each iteration. All the contextual bandit algorithms are executed up to 300 iterations per user in this experiment.

Two separate neural networks are trained to approximate the actor and critic

functions. In figure 6.5 we show the architecture of our actor and critic network. The critic network consists of two fully connected layers with 300 and 500 units respectively. The action component a is provided in the second layer of the network. The actor network has one fully connected layer with 500 units. Leaky rectified linear units (ReLU) are used as the activation function for both of these network.

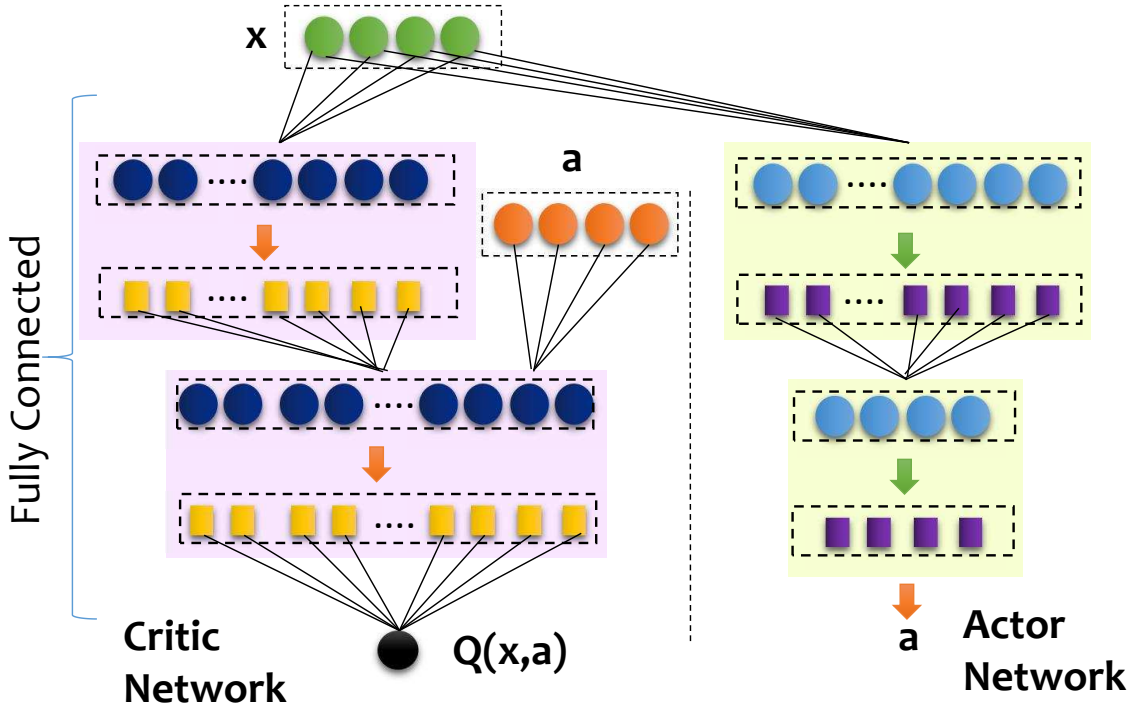
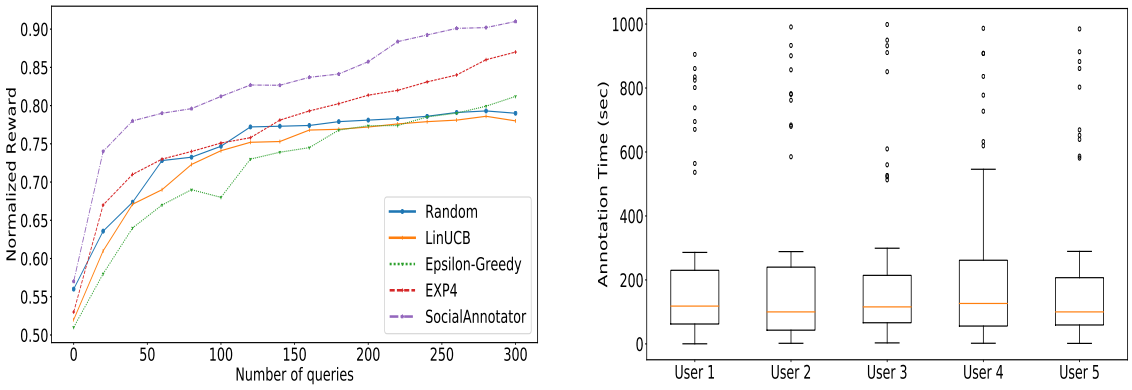


Figure 6.5: Architecture of actor-critic network.

The actor and critic network are interlinked which facilitates the activations of the neurons to flow from actor to critic network. Therefore, the gradient of the critic network influences the actor network as well during back propagation. By connecting the networks together, the critic network control the directions of improvement in the action space without explicit targets. We incrementally decrease the learning rate of the critic network when the critic value increases. The learning rate is updated

to $\alpha_q = 0.0001 \times 10^{\alpha \tilde{Q}}$ after every hundred steps where $\alpha < 0$ and $\tilde{Q}$ is the mean of critic value in the last hundred steps. This helps the model in generalization and be autonomous from the convergence speed. For the actor network, we use a constant learning rate $\alpha_\pi = 0.0001$.



(a) (b)

Figure 6.6: (a) shows the distribution of normalized reward with respect to number of queries. (b) shows per-annotator labeling time distribution.

6.4.2 Bandit Performance

In Table 6.1, we list and compare average rewards($\bar{r}$), average costs($\bar{c}$), average reward to cost ratio ($\bar{r}/\bar{c}$) and the percentage of time optimal annotator gets selected (% opt) of different bandit algorithms. From the statistics we see that LinUCB and ϵ-Greedy perform worst with respect EXP4 and our model. Both these algorithms are not meant for problems with budget constaints and as a result they do not take budget into consideration. Our model can achieve higher reward at lower cost contrast to other bandits which verifies that we are choosing optimal annotator at

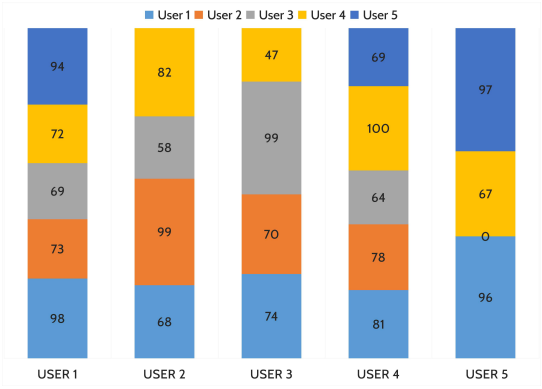
each step. In figure 6.6a the trend of average reward obtained at each step is shown. It is evident that our proposed algorithm outperforms the other bandit algorithm settings. More context information like detailed interaction, fine grained location information etc. might further improve the model.

Table 6.1: Statistics of SocialAnnotator compared to other multi-armed bandit algorithm

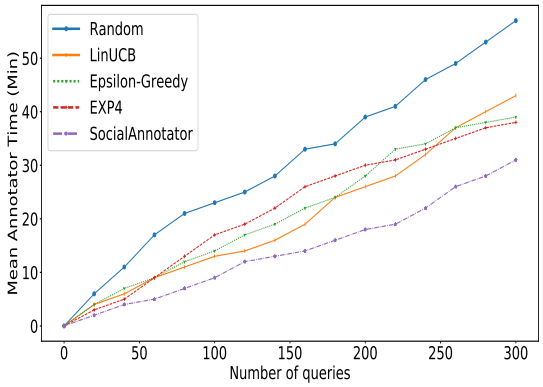
	$\bar{r}$	$\bar{c}$	$\bar{r}/\bar{c}$	% opt
Random	0.763	0.793	0.962	1.67
LinUCB	0.758	0.814	0.932	0.8
ϵ-Greedy	0.796	0.886	0.8984	1.32
EXP4	0.864	0.810	1.067	61.17
SocialAnnotator	0.913	0.267	3.419	73.42

6.4.3 Annotator Selection

We monitor the performance of each annotator and maintain a score for each activity associated with the connected users. In Figure 6.6b we provide the annotation time distributions of each user using boxplot. A box depicts the majority of annotation times and the median time is marked with a solid line inside the box. It is noticeable from the figure that each user have different time distribution which means the efficiency, promptness and reliability of each user varies. We also deduce that the annotator might not provide the label at all. We show the percentage of



(a)



(b)

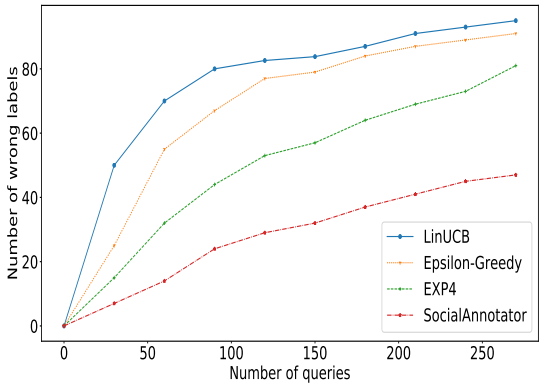
Figure 6.7: (a) represents the stack plot of percentage of correctly labeled instances of all the connected users for each user. (b) illustrates the mean annotation time for different bandit algorithms.

correctly labeled instances by each user in Figure 6.7a. As *User 5* is only connected to *User 4* and *User 1* there are only three scores for him including the score of labeling his own activity. It is apparent that all the users are efficient in labeling their own activity.

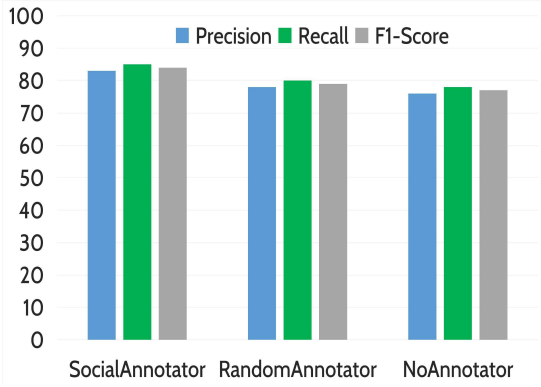
Table 6.2: Labeling result of each user

	Correct Label	Wrong Label	No Label
User 1	324	49	27
User 2	306	68	26
User 3	285	83	32
User 4	310	73	17
User 5	345	37	18

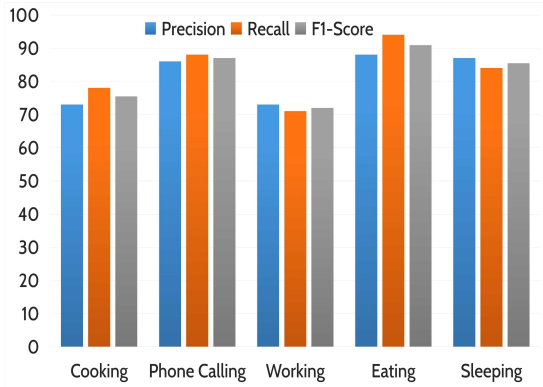
We notice that *User 1* and *User 5* were able to label each others data quite precisely. We found that these two users were living in the same apartment and User 1 is *spouse* of User 5. Their quantity of interaction was also very high as apart from living together they were also talking with each other over the phone couple of times a day. We also notice from the figure that *User 2* and *User 4* were able to label the activity of each other with good accuracy (82% for *User 2* and 78% for *User 4*). After investigating into it, we observed that these two users were working together at the same place and had a lot of interactions. *User 3* and *User 4* also worked at the same place but they had very less interaction with each other which is reflected in their annotation efficiency. Receiving the label information as early as possible is aslo imperative. If we do not receive a label for the queried instance within a certain pre-defined threshold tme, we discard the annotator. As a result, if we pose the query to an user who may delay in providing the label or not provide the label at all, we not only spoil resources but also lose valuable information. In Figure 6.7b we show the mean annotation time needed using different bandit algorithm for varying number of queries. Our model exhibits lowest mean (30 mins) than all other approaches and getting the labels at the right time. As a result our model also ensures immediate result along with the conservation of information. To further validate our claim, we show the number of wrong labels received for different bandit algorithms in 300 iterations in Figure 6.8a. After 300 iterations, *SocialAnnotator* indicates lowest number of wrong labels, which proves that we are posing the queries to the right person at the right time. The cumulative labeling accuracy of each user is also described in Table 6.2.



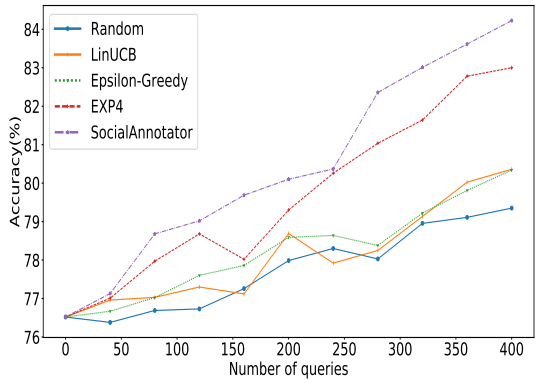
(a)



(b)



(a)



(b)

Figure 6.9: (a) shows the number of wrong lables received in 300 iterations for different bandit algorithms. (b) illustrates the precision, recall and f1-score of our base classifier using different settings. (c) depicts the final precision, recall and f1-score of each activity. (d) demonstrates the progression of accuracy after each iteration.

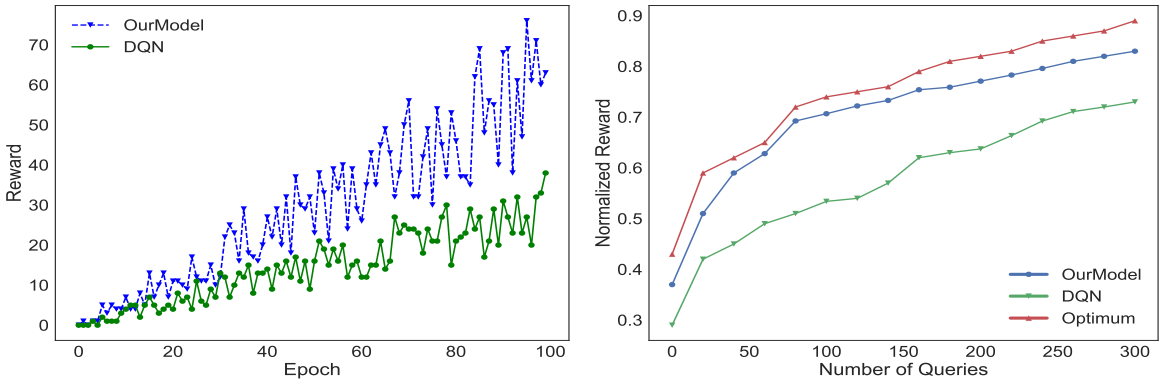
6.4.4 Actor-Critic Performance

To demonstrate the effectiveness of our annotator selection algorithm, we compared our algorithm with other popular deep reinforcement learning algorithm -

Deep Q-Network (DQN) [210]. In Figure 6.10a we show the evolution of the average total reward during training for both our model and DQN. Both averaged reward plots are quite noisy, pointing that the learning algorithms are not making steady progress. However for our model, we achieve higher reward on average than DQN. In Figure 6.10b we show the normalized reward with the progression of number of queries posed. We see that our model, approximates better reward than DQN and converges faster to the optimum reward received. The reward distribution for individual users are shown in Figure 6.11. We received comparatively high reward from User12, User14, User19 and User9 than other users. Upon investigation, we noticed most of the queries posed to these users were of themselves, which they were able to label with high efficiency. Their annotator list or number of connected users were below average. The average number of connection each user had in our experiments were 6. For the aforementioned users, they had fewer than 3 connections which made the algorithm to pose the queries to them more often. In Figure 6.12 the distributions of annotation time for individual users are presented.

6.4.5 Active Learning Performance

For our SocialAnnotator model, We have achieved an overall accuracy of 77% for our base classifier. We trained our model with only 5% (1050) of the total labeled data instances. We apply active learning and incrementally query instances. We show the overall accuracy of our classifier after 400 iterations in Figure 6.8b. By employing *SocialAnnotator* we accomplish an average accuracy of $\approx 84\%$ which is



(a) (b)

Figure 6.10: (a) illustrates average reward received with respect to number of epochs while training. (b) shows the progression of normalized reward with respect to number of posed queries.

an improvement of 7% compared to our base classifier. *NoAnnotator* title demonstrates the results when we do not administer annotator selection. We notice that it only improve the accuracy by $\approx$ 1-2% even if we have applied active learning. So posing the query to the right person helps to improve the accuacy of our classifier. In Figure 6.9a we show the accuracy of individual activities after applying *SocialAnnotator*. We see that *Cooking* and *Working* show low accuracies with respect to other activities. After further investigation, we noticed that these two activities itself are very complex and experienced low accuracies in our base classifier as well (68% and 64%). However, *SocialAnnotator* actually increased the accuracy by $\approx$ 8-10%. Consequently *SocialAnnotator* helps to improve the accuracies of complex activities which are hard to infer. Figure 6.9b shows the change of accuracy in 400 iterations. Our algorithm converges to optimum accuracy faster than other

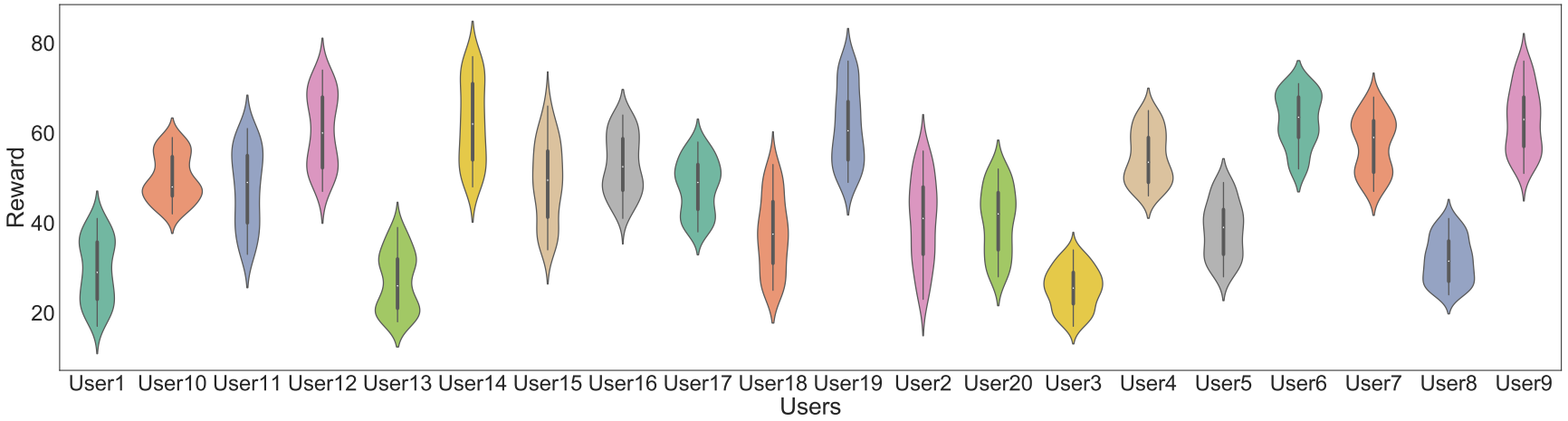


Figure 6.11: The plot illustrates the reward distribution of 20 users during our experiment.

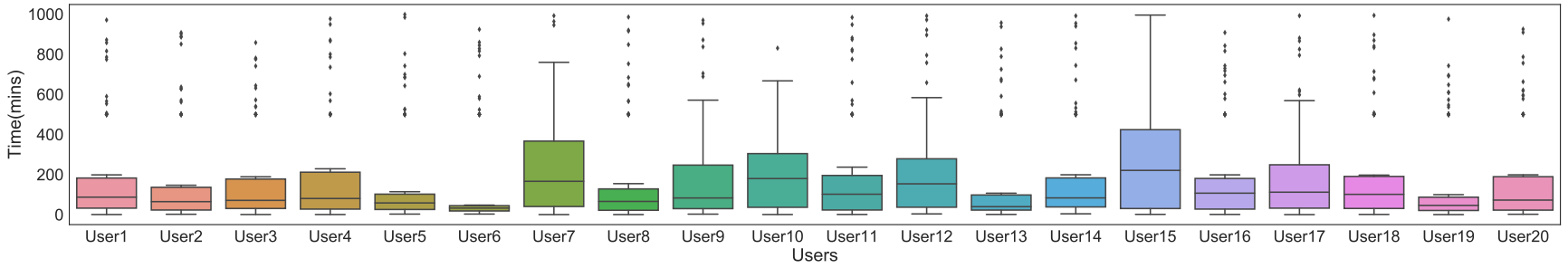


Figure 6.12: The figure illustrates the labeling time distribution of 20 users.

approaches.

After training our deep active learner model with 30% labeled instances only, we achieved 83.26% prediction accuracy. We then calculate most informative instances from a pool of unlabeled instances at each iteration. After each 100 queries we retrain our model and adjust the model parameters with the received label information. We then validate our model accuracy using our test dataset. We compare our model with other existing deep models in Table 6.3. In Table 6.3 we report the progression of model accuracy after each 100 queries. We see that using active learning all the models are exhibiting gradual improvement. The model proposed in [203] exhibits highest increment in accuracy ($\approx 9\%$) after 400 iterations. Our model shows better initial accuracy and converges faster to optimum accuracy which is (92.05%).

Table 6.3: Comparison of our algorithm with other existing approaches with varying number of queries.

AR System	**Number of Queries**				
	0	100	200	300	400
Francisco [79]	81.2	84.14	87.87	88.57	89.47
Ming Zeng et al. [203]	80.39	82.47	86.33	87.59	89.25
Mohammad et al. [78]	81.01	83.17	84.11	87.89	88.32
Our model	83.26	85.65	88.39	90.48	91.64

Chapter 7: Conclusion and Future Works

In this thesis, we have proposed and validated a number of active learning methodologies to help us build a scalable activity recognition model. By acquiring active feedback from the users, we get more refined and precise activity information. We have proposed a novel annotator selection method *SocialAnnotator*, by exploiting social relationships among the users to improve the efficiency of active learning in activity recognition context. Our proposed model selects annotator based on the strength of the relationships and *spatio-temporal* distance metrics among the users. We also consider the similarities between the activities in our model to calculate the level of *correspondence* among the users. Prior works with active learning that propose to mitigate the labeling effort, have not considered the influence of annotators in their model. Our results show that, *SocialAnnotator* can compliment active learning and establish reliable, prompt and accurate label information. We have demonstrated that by using our methodology, we improved the accuracy of our base classifier by $\approx 7\%$. In our current approach while calculating the distance between two users, we only consider a very few interactions between them. In future we want to apprehend more interactions as well as more context information unobtrusively. Also in terms of evaluating the performance of the annotators we are not considering

the complexity of the queries. If we evaluate the annotators based on how many queries he or she has answered in that case it does not properly reflect the true efficiency and effectiveness of them. Here the complexity of the query reflect how subtle the activity was. For example, during performing an activity, the user might be doing another activity for a very short period of time. The information about that would be fundamental for our model to learn. So if an annotator were to answer that, it would make a profound impact in learning the feature representation. For this reason we want to investigate how we can define the sample complexity in future and make the active learning process more effective. One more important factor for making active learning effective is considering the availability of the annotators. This is important because the timing to pose the query is important. If we provide the query to the user in the middle of the night, then generally a prompt response is not expected. In future we want to address this issue of annotator availability. We want to monitor the users phone usages and social network interactions without needing any feedback from them and add more sensor modalities like ambient infrastructure sensor to record the movements in detail. We also plan to do the regret anaylsis of our algorithm and derive the upper-bound in future. Another important aspect we did not address in our work is the underlying loss of efficiency for employing annotator selection model. We intend to investigate and quantify the trade off between traditional active learning approach without annotator selection and with annotator selection. We can then understand the true improvement of our proposed approach. If the annotators provide misleading labels in the earlier iterations then the annotator selection methodology will fail to adapt and eventually the selected

candidates will be wrong. In such cases we want to investigate trust region based policy optimization in future to address this problem and ensure we do not explore to unknown region while doing exploration for other candidate annotators.

Bibliography

[1] H. M. Sajjad Hossain, Nirmalya Roy, and Md Abdullah Al Hafiz Khan. Sleep well: A sound sleep monitoring framework for community scaling. In *16th IEEE International Conference on Mobile Data Management, MDM 2015, Pittsburgh, PA, USA, June 15-18, 2015 - Volume 1*, pages 44–53, 2015.

[2] Timilehin Labeodan, Wim Zeiler, Gert Boxem, and Yang Zhao. Occupancy measurement in commercial office buildings for demand-driven control applicationsa survey and detection system evaluation. *Energy and Buildings*, 93:303 – 314, 2015.

[3] A. Corna and et al. Occupancy detection via ibeacon on android devices for smart building management. In *DATE '15*, pages 629–632.

[4] M. M. Baig and H. Gholamhosseini. Smart health monitoring systems: an overview of design and modeling. *J Med Syst*, 37(2):9898, 2013.

[5] C. Perera, C.H. Liu, and S. Jayawardena. The emerging internet of things marketplace from an industrial perspective: A survey. *Emerging Topics in Computing, IEEE Transactions on*, PP(99):1–1, 2015.

[6] Yu Guan and Thomas Plötz. Ensembles of deep LSTM learners for activity recognition using wearables. *IMWUT*, 1(2):11:1–11:28, 2017.

[7] Hande Alemdar and Cem Ersoy. Multi-resident activity tracking and recognition in smart environments. *J. Ambient Intelligence and Humanized Computing*, 8(4):513–529, 2017.

[8] Muhammad Shoaib, Stephan Bosch, Özlem Durmaz Incel, Hans Scholten, and Paul J. M. Havinga. A survey of online activity recognition using mobile phones. *Sensors*, 15(1).

[9] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2):201–221, 1994.

[10] Heng-Tze Cheng, Feng-Tso Sun, Martin L. Griss, Paul Davis, Jianguo Li, and Di You. Nuactiv: recognizing unseen new activities using semantic attribute-based learning. In *MobiSys*, pages 361–374.

[11] Du Tran and Alexander Sorokin. Human activity recognition with metric learning. In *Computer Vision–ECCV 2008*, pages 548–561. Springer, 2008.

[12] Wesley Kerr, Anh Tran, and Paul Cohen. Activity recognition with finite state machines. In *Proc. of IJCAI'11*, 2011.

[13] Tom Diethe, Niall Twomey, and Peter Flach. Bayesian active transfer learning in smart homes. In *Advances in Active Learning : Bridging Theory and Practice, ICML 2015*, 2015.

[14] Rong Liu, Ting Chen, and Lu Huang. Research on human activity recognition based on active learning. In *ICMLC*, volume 1, pages 285–290, 2010.

[15] Maja Stikic, Kristof Van Laerhoven, and Bernt Schiele. Exploring semi-supervised and active learning for activity recognition. In *Proceedings of the 2008 12th IEEE International Symposium on Wearable Computers*, ISWC '08, pages 81–88, 2008.

[16] Yu-chen Ho, Ching-hu Lu, I-han Chen, Shih-shinh Huang, Ching-yao Wang, and Li-chen Fu. Active-learning assisted self-reconfigurable activity recognition in a dynamic environment. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, pages 1567–1572, Piscataway, NJ, USA, 2009. IEEE Press.

[17] Walter S Lasecki, Young Chol Song, Henry Kautz, and Jeffrey P Bigham. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 1203–1212, 2013.

[18] E. Hoque and J. Stankovic. Aalo: Activity recognition in smart homes using active learning in the presence of overlapped activities. In *PervasiveHealth*, pages 139–146, 2012.

[19] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[20] B. Settles. From theories to queries: Active learning in practice. In Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire, and Alexander Statnikov, editors, *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, volume 16 of *Proceedings of Machine Learning Research*, pages 1–18. PMLR, 16 May 2011.

[21] Zalán Bodó, Zsolt Minier, and Lehel Csató. Active learning with clustering. In *Active Learning and Experimental Design workshop, In conjunction with AISTATS 2010, Sardinia, Italy, May 16, 2010*, pages 127–139, 2011.

[22] James Love and Rudy Hirschheim. Crowdsourcing of information systems research. *EJIS*, 26(3):315–332.

[23] Kehua Guo, Yayuan Tang, and Peiyun Zhang. CSF: crowdsourcing semantic fusion for heterogeneous media big data in the internet of things. *Information Fusion*, 37:77–85, 2017.

[24] Yung-Ju Chang, Gaurav Paruthi, Hsin-Ying Wu, Hsin-Yu Lin, and Mark W. Newman. An investigation of using mobile and situated crowdsourcing to collect annotated travel activity data in real-word settings. *Int. J. Hum.-Comput. Stud.*, 102:81–102, 2017.

[25] Niall Twomey, Tom Diethe, Ian Craddock, and Peter A. Flach. Unsupervised learning of sensor topologies for improving activity recognition in smart environments. *Neurocomputing*, 234:93–106, 2017.

[26] Laura Fiorini, Filippo Cavallo, and volume = 17 number = 5 pages = 1034 year = 2017 et al. title = Unsupervised Machine Learning for Developing Personalised Behaviour Models Using Activity Data, journal = Sensors.

[27] Hristijan Gjoreski and Daniel Roggen. Unsupervised online activity discovery using temporal behaviour assumption. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers, ISWC 2017, Maui, HI, USA, September 11-15, 2017*, pages 42–49, 2017.

[28] Pinar Donmez and Jaime G. Carbonell. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 619–628, 2008.

[29] Elizabeth Gerber and Panos Ipeirotis, editors. *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2015, November 8-11, 2015, San Diego, California*. AAAI Press, 2015.

[30] Qiang Yang and Michael Wooldridge, editors. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press, 2015.

[31] H. M. Sajjad Hossain, Md Abdullah Al Hafiz Khan, and Nirmalya Roy. Active learning enabled activity recognition. *Pervasive and Mobile Computing*, 38:312–330, 2017.

[32] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D. Lane, Cecilia Mascolo, Mahesh K. Marina, and Fahim Kawsar. Multimodal deep learning for activity and context recognition. *IMWUT*, 1(4):157:1–157:27, 2017.

[33] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. Deep learning algorithms for human activity recognition

using mobile and wearable sensor networks: State of the art and research challenges. *Expert Syst. Appl.*, 105:233–261, 2018.

[34] Gabriele Civitarese, Claudio Bettini, Timo Sztyler, Daniele Riboni, and Heiner Stuckenschmidt. NECTAR: knowledge-based collaborative active learning for activity recognition. In *2018 IEEE International Conference on Pervasive Computing and Communications, PerCom 2018, Athens, Greece, March 19-23, 2018*, pages 1–10, 2018.

[35] Farhad Shahmohammadi, Anahita Hosseini, Christine E. King, and Majid Sarrafzadeh. Smartwatch based activity recognition using active learning. In *Proceedings of the Second IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies, CHASE 2017, Philadelphia, PA, USA, July 17-19, 2017*, pages 321–329, 2017.

[36] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive Computing, Second International Conference, PERVASIVE 2004, Vienna, Austria, April 21-23, 2004, Proceedings*, pages 1–17, 2004.

[37] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, 2013.

[38] Zhenyu He, Zhibin Liu, Lianwen Jin, Li-Xin Zhen, and Jian-Cheng Huang. Weightlessness feature - a novel feature for single tri-axial accelerometer based activity recognition. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[39] Marko Kos and Iztok Kramberger. A wearable device and system for movement and biometric data acquisition for sports applications. *IEEE Access*, 5:6411–6420, 2017.

[40] Florian Daiber and Felix Kosmalla. Tutorial on wearable computing in sports. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI 2017, Vienna, Austria, September 4-7, 2017*, pages 65:1–65:4, 2017.

[41] H. M. Sajjad Hossain, Md Abdullah Al Hafiz Khan, and Nirmalya Roy. Soccermate: A personal soccer attribute profiler using wearables. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017, Kona, Big Island, HI, USA, March 13-17, 2017*, pages 164–169, 2017.

[42] Samer Samarah, Mohammed G. H. al Zamil, Ahmed F. Al-Eroud, Majdi Rawashdeh, Mohammed F. Alhamid, and Atif Alamri. An efficient activity recognition framework: Toward privacy-sensitive health data sensing. *IEEE Access*, 5:3848–3859, 2017.

[43] T. Subetha and S. Chitrakala. A survey on human activity recognition from videos. In *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, pages 1–7, Feb 2016.

[44] Weixin Li and Nuno Vasconcelos. Complex activity recognition via attribute dynamics. *International Journal of Computer Vision*, 122(2):334–370, 2017.

[45] Ahmad Jalal, Yeonho Kim, Yong-Joong Kim, Shaharyar Kamal, and Daijin Kim. Robust human activity recognition from depth video using spatiotemporal multi-fused features. *Pattern Recognition*, 61:295–308, 2017.

[46] Timo Sztyler, Heiner Stuckenschmidt, and Wolfgang Petrich. Position-aware activity recognition with wearable devices. *Pervasive and Mobile Computing*, 38:281–295, 2017.

[47] Majid Janidarmian, Atena Roshan Fekr, Katarzyna Radecka, and Zeljko Zilic. A comprehensive analysis on wearable acceleration sensors in human activity recognition. *Sensors*, 17(3):529, 2017.

[48] Juan Carlos Davila, Ana-Maria Cretu, and Marek B. Zaremba. Wearable sensor data classification for human activity recognition based on an iterative learning framework. *Sensors*, 17(6):1287, 2017.

[49] Igor Bisio, Alessandro Delfino, Fabio Lavagetto, and Andrea Sciarrone. Enabling iot for in-home rehabilitation: Accelerometer signals classification methods for activity and movement recognition. *IEEE Internet of Things Journal*, 4(1):135–146, 2017.

[50] Narayanan Chatapuram Krishnan and Diane J. Cook. Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, 2014.

[51] Michalis Vrigkas et al. A review of human activity recognition methods. *Front. Robotics and AI*, 2015.

[52] Yonggang Lu, Ye Wei, Li Liu, Jun Zhong, Letian Sun, and Ye Liu. Towards unsupervised physical activity recognition using smartphone accelerometers. *Multimedia Tools and Applications*, 76(8):10701–10719, 2017.

[53] Md Abdullah Al Hafiz Khan and Ruthvik Kukkapalli. RAM: radar-based activity monitor. In *Proc. of IEEE INFOCOM*, 2016.

[54] Tâm Huynh and Bernt Schiele. Unsupervised discovery of structure in activity data using multiple eigenspaces. In *Proc. of LoCA*, 2006.

[55] Andreas Bulling et al. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.*, 2014.

[56] Tâm Huynh et al. Discovery of activity patterns using topic models. In *Proc. of UbiComp*, 2008.

[57] E. Kim, S. Helal, and D. Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1):48–53, 2010.

[58] Young-Seol Lee and Sung-Bae Cho. *Activity Recognition Using Hierarchical Hidden Markov Models on a Smartphone with 3D Accelerometer*, pages 460–467. 2011.

[59] Danny Wyatt, Matthai Philipose, and Tanzeem Choudhury. Unsupervised activity recognition using automatically mined common sense. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, AAAI'05, pages 21–27, 2005.

[60] Liang Lin, Keze Wang, Wangmeng Zuo, Meng Wang, Jiebo Luo, and Lei Zhang. A deep structured model with radius-margin bound for 3d human activity recognition. *International Journal of Computer Vision*, 118(2):256–273, 2016.

[61] Shreyank N. Gowda. Human activity recognition using combinatorial deep belief networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, Honolulu, HI, USA, July 21-26, 2017*, pages 1589–1594, 2017.

[62] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Syst. Appl.*, 59:235–244, 2016.

[63] Valentin Radu, Nicholas D. Lane, Sourav Bhattacharya, Cecilia Mascolo, Mahesh K. Marina, and Fahim Kawsar. Towards multimodal deep learning for activity recognition on mobile devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp Adjunct 2016, Heidelberg, Germany, September 12-16, 2016*, pages 185–188, 2016.

[64] Sourav Bhattacharya and Nicholas D. Lane. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2016, Sydney, Australia, March 14-18, 2016*, pages 1–6, 2016.

[65] L. Chen, C. D. Nugent, and H. Wang. A knowledge-driven approach to activity recognition in smart homes. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):961–974, 2012.

[66] Daniele Riboni, Timo Sztyler, Gabriele Civitarese, and Heiner Stuckenschmidt. Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning. In *Proceedings of the 2016*

ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2016, Heidelberg, Germany, September 12-16, 2016, pages 1–12, 2016.

[67] K. S. Gayathri, K. S. Easwarakumar, and Susan Elias. Probabilistic ontology based activity recognition in smart homes using markov logic network. *Knowl.-Based Syst.*, 121:173–184, 2017.

[68] Isibor Kennedy Ihianle, Usman Naeem, and Syed Islam. Ontology-driven activity recognition from patterns of object use. In *Adjunct Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers, UbiComp/ISWC 2017, Maui, HI, USA, September 11-15, 2017*, pages 654–657, 2017.

[69] Tomoya Hirano and Takuya Maekawa. A hybrid unsupervised/supervised model for group activity recognition. In *Proceedings of the 17th Annual International Symposium on Wearable Computers. ISWC 2013, Zurich, Switzerland, September 8-12, 2013*, pages 21–24, 2013.

[70] Sebastian Münzner, Philip Schmidt, Attila Reiss, Michael Hanselmann, Rainer Stiefelhagen, and Robert Dürichen. Cnn-based sensor fusion techniques for multimodal human activity recognition. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ISWC '17, pages 158–165, 2017.

[71] Kevin Bouchard, Bruno Bouchard, and Abdenour Bouzouane. Unsupervised discovery of spatial relationships between objects for activity recognition inside smart home. In *The 2012 ACM Conference on Ubiquitous Computing, Ubicomp '12, Pittsburgh, PA, USA, September 5-8, 2012*, pages 655–656, 2012.

[72] Nicholas D. Lane and Petko Georgiev. Can deep learning revolutionize mobile sensing? In *Proc. of HotMobile 2015*.

[73] Nils Y. Hammerla and Shane Halloran. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proc. of IJCAI 2016*.

[74] N. D. Lane and S. Bhattacharya. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *Proc. of IPSN*, 2016.

[75] Wenchao Jiang and Zhaozheng Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proc. of ACM MM 2015*.

[76] Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *Proc. of IJCAI 2011*, 2011.

[77] Tushar Dobhal and Vivswan Shitole. Human activity recognition using binary motion image and deep learning. *Procedia Computer Science*, 2015.

[78] Mohammad Abu Alsheikh, Ahmed Selim, Dusit Niyato, Linda Doyle, Shaowei Lin, and Hwee-Pink Tan. Deep activity recognition models with triaxial accelerometers. In *Artificial Intelligence Applied to Assistive Technologies and Smart Environments, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12, 2016*, 2016.

[79] Francisco Javier et al. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 2016.

[80] Moez Baccouche and Franck Mamalet. Sequential deep learning for human action recognition. In *Proc. of IEEE HBU, 2011*.

[81] Nicholas D. Lane and Petko Georgiev. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proc. of ACM UbiComp 2015*.

[82] Xinyu et al. Li. Deep learning for rfid-based activity recognition. SenSys '16.

[83] Juarez Monteiro, Roger Granada, Rodrigo C Barros, and Felipe Meneguzzi. Deep neural networks for kitchen activity recognition. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 2048–2055. IEEE, 2017.

[84] Francisco Javier Ordóñez Morales and Daniel Roggen. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ISWC '16, pages 92–99, 2016.

[85] Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. AROMA: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. *IMWUT*, 2(2):74:1–74:16, 2018.

[86] Akhil Mathur, Tianlin Zhang, Sourav Bhattacharya, Petar Velickovic, Leonid Joffe, Nicholas D. Lane, Fahim Kawsar, and Pietro Liò. Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2018, Porto, Portugal, April 11-13, 2018*, pages 200–211, 2018.

[87] Kundan Krishna, Deepali Jain, Sanket V. Mehta, and Sunav Choudhary. An LSTM based system for prediction of human activities with durations. *IMWUT*, 1(4):147:1–147:31, 2017.

[88] Md Abdullah Al Hafiz Khan, Nirmalya Roy, and Archan Misra. Scaling human activity recognition via deep learning-based domain adaptation. In *2018 IEEE*

International Conference on Pervasive Computing and Communications, PerCom 2018, Athens, Greece, March 19-23, 2018, pages 1–9, 2018.

[89] Md Abdullah Al Hafiz Khan and Nirmalya Roy. Transact: Transfer learning enabled activity recognition. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017, Kona, Big Island, HI, USA, March 13-17, 2017*, pages 545–550, 2017.

[90] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, and Philip S. Yu. Stratified transfer learning for cross-domain activity recognition. In *2018 IEEE International Conference on Pervasive Computing and Communications, PerCom 2018, Athens, Greece, March 19-23, 2018*, pages 1–10, 2018.

[91] Seyed Ali Rokni, Marjan Nourollahi, and Hassan Ghasemzadeh. Personalized human activity recognition using convolutional neural networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[92] Pinar Donmez and Jaime G. Carbonell. Paired sampling in density-sensitive active learning. In *ISAIM*.

[93] Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *In Proceedings of the 21st International Conference on Machine Learning.*, pages 623–630, 2004.

[94] Burr Settles. Active learning literature survey. Technical report, 2010.

[95] Steve Hanneke. Theory of disagreement-based active learning. *Found. Trends Mach. Learn.*, 7(2-3):131–309, 2014.

[96] M. Aminian. Active learning for reducing bias and variance of a classifier using jensen-shannon divergence. In *Machine Learning and Applications, 2005. Proceedings. Fourth International Conference on*, 2005.

[97] Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. Multi-instance learning by treating instances as non-i.i.d. samples. In *ICML*, pages 1249–1256, 2009.

[98] Shiliang Sun and David R. Hardoon. Active learning with extremely sparse labeled examples. *Neurocomput.*, 73(16-18):2980–2988, 2010.

[99] Maria-Florina Balcanet et al. Agnostic active learning. *J. Comput. Syst. Sci.*, 75(1):78–89, 2009.

[100] Alina Beygelzimer and et al. Importance weighted active learning. In *ICML*, pages 49–56, 2009.

[101] Shervin Javdani and et al. Near optimal bayesian active learning for decision making. *CoRR*, 2014.

[102] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *ICML*, pages 208–215, 2008.

[103] H. M. Sajjad Hossain et al. Active learning enabled activity recognition. In *IEEE PerCom*, 2016.

[104] Hande Alemdar, TLM van Kasteren, and Cem Ersoy. Active learning with uncertainty sampling for large scale activity recognition in smart homes. *Journal of Ambient Intelligence and Smart Environments*, 9(2):209–223, 2017.

[105] Dan Wang and Yi Shang. A new active labeling method for deep learning. In *Proc. of IJCNN 2014.*

[106] Shusen Zhou and Qingcai Chen. Active deep networks for semi-supervised sentiment classification. In *Proc. of COLING.*

[107] Fabian Stark and Rudolph Triebel. Captcha recognition with active deep learning, 2015.

[108] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016.

[109] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *CoRR*, abs/1701.03551, 2017.

[110] Sohrab Saeb, Thaddeus R Cybulski, Konrad P Kording, and David C Mohr. Scalable passive sleep monitoring using mobile phones: Opportunities and obstacles. *Journal of Medical Internet Research*, 19(4):e118, 2017.

[111] James M. Parish. Sleep-related problems in common medical conditions. *CHEST Journal*, 135(2):563–572, 2009.

[112] Christopher Kline. *Sleep Quality*, pages 1811–1813. Springer New York, New York, NY, 2013.

[113] Fitbit. `http://www.fitbit.com/`, 2007.

[114] Actigraph. `http://www.actigraphcorp.com/`, 2004.

[115] Basis Band B1. `http://www.mybasis.com/`, 2014.

[116] Janna Mantua, Nickolas Gravel, and Rebecca Spencer. Reliability of sleep measures from four personal health monitoring devices compared to research-based actigraphy and polysomnography. *Sensors*, 16(5):646, 2016.

[117] Chih-En Kuo, Yi-Che Liu, Da-Wei Chang, Chung-Ping Young, Fu-Zen Shaw, and Sheng-Fu Liang. Development and evaluation of a wearable device for sleep quality assessment. *IEEE Trans. Biomed. Engineering*, 64(7):1547–1557, 2017.

[118] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[119] Salikh Bagaveyev and Diane J. Cook. Designing and evaluating active learning methods for activity recognition. In *The 2014 ACM Conference on Ubiquitous Computing, UbiComp '14 Adjunct*, pages 469–478, 2014.

[120] Hande Özgür Alemdar, Tim van Kasteren, and Cem Ersoy. Using active learning to allow activity recognition on a large scale. In *Ambient Intelligence - Second International Joint Conference on AmI 2011, Amsterdam, The Netherlands, November 16-18, 2011. Proceedings*, pages 105–114, 2011.

[121] Anand Inasu Chittilappilly, Lei Chen, and Sihem Amer-Yahia. A survey of general-purpose crowdsourcing techniques. *IEEE Trans. Knowl. Data Eng.*, 28(9):2246–2266, 2016.

[122] Kyuwoong Hwang and Soo-Young Lee. Environmental audio scene and activity recognition through mobile-based crowdsourcing. *IEEE Trans. Consumer Electronics*, 58(2):700–705, 2012.

[123] Android Wear. `http://www.android.com/intl/en_us/wear/`, 2014.

[124] Stephan Jonas, Andreas Hannig, Cord Spreckelsen, and Thomas M. Deserno. Wearable technology as a booster of clinical care. volume 9039, 2014.

[125] EZ430-Chronos. `http://www.ti.com/tool/ez430-chronos`, 2013.

[126] Marko Borazio, Eugen Berlin, Nagihan Kücükyildiz, Philipp M. Scholl, and Kristof Van Laerhoven. Towards benchmarked sleep detection with inertial wrist-worn sensing units. In *ICHI 2014*, Verona, Italy, 09/2014 2014.

[127] Andrew L. Chesson et al. Practice parameters for the indications for polysomnography and related procedures. *Sleep*, 20:406–422, 1997.

[128] Murray W. Johns. Sensitivity and specificity of the multiple sleep latency test (mslt), the maintenance of wakefulness test and the epworth sleepiness scale: Failure of the mslt as a gold standard. *Journal of Sleep Research*, 9(1):5–11, 2000.

[129] Oakley NR. Validation with polysomnography of the sleep-watch sleep/wake scoring algorithm used by the actiwatch activity monitoring system. In *Technical report to Mini Mitter Co., Inc.*, 1997.

[130] A. Sadeh. The role and validity of actigraphy in sleep medicine: an update. *Sleep Med Rev*, 15(4):259–267, Aug 2011.

[131] Kristof Van Laerhoven, Marko Borazio, David Kilian, and Bernt Schiele. Sustained logging and discrimination of sleep postures with low-level, wrist-worn sensors. In *Proceedings of the 2008 12th IEEE International Symposium on Wearable Computers*, ISWC '08, pages 69–76, 2008.

[132] Aarti Sathyanarayana, Jaideep Srivastava, and Luis Fernández-Luque. The science of sweet dreams: Predicting sleep efficiency from wearable device data. *IEEE Computer*, 50(3):30–38, 2017.

[133] Rachael Purta, Stephen Mattingly, Lixing Song, Omar Lizardo, David Hachen, Christian Poellabauer, and Aaron Striegel. Experiences measuring sleep and physical activity patterns across a large college cohort with fitbits. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers, ISWC 2016, Heidelberg, Germany, September 12-16, 2016*, pages 28–35, 2016.

[134] Xiao Sun, Li Qiu, Yibo Wu, Yeming Tang, and Guohong Cao. Sleepmonitor: Monitoring respiratory rate and body position during sleep using smartwatch. *IMWUT*, 1(3):104:1–104:22, 2017.

[135] Shun Matsui, Tsutomu Terada, and Masahiko Tsukamoto. Smart eye mask: eye-mask shaped sleep monitoring device. In *Adjunct Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers, UbiComp/ISWC 2017, Maui, HI, USA, September 11-15, 2017*, pages 265–268, 2017.

[136] Aarti Sathyanarayana, Ferda Ofli, Luis Fernandes-Luque, Jaideep Srivastava, Ahmed K. Elmagarmid, Teresa Arora, and Shahrad Taheri. Robust automated human activity recognition and its application to sleep research. *CoRR*, abs/1607.04867, 2016.

[137] Mahsan Rofouei, Mike Sinclair, Ray Bittner, Tom Blank, Nick Saw, Gerald DeJean, and Jeff Heffron. A non-invasive wearable neck-cuff system for real-time sleep monitoring. In *BSN*, pages 156–161, 2011.

[138] Anh Nguyen, Raghda Alqurashi, Zohreh Raghebi, Farnoush Banaei-kashani, Ann C Halbower, and Tam Vu. A lightweight and inexpensive in-ear sensing system for automatic whole-night sleep stage monitoring. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 230–244. ACM, 2016.

[139] K. Nakajima, Y. Matsumoto, and T. Tamura. A monitor for posture changes and respiration in bed using real time image sequence analysis. In *Engineering in Medicine and Biology Society.*, volume 1, 2000.

[140] Wen-Hung Liao and Chien-Ming Yang. Video-based activity and movement pattern analysis in overnight sleep studies. In *ICPR*, pages 1–4, 2008.

[141] Matthew Kay, Eun K. Choe, and et al. Lullaby: A capture & access system for understanding the sleep environment. In *UbiComp*, 2012.

[142] Yanzhi Ren, Chen Wang, Jie Yang, and Yingying Chen. Fine-grained sleep monitoring: Hearing your breathing with smartphones. In *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, pages 1194–1202, 2015.

[143] Jin Zhang, Qian Zhang, Yuanpeng Wang, and Chen Qiu. A real-time auto-adjustable smart pillow system for sleep apnea detection and treatment. In *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, pages 179–190, April 2013.

[144] N. Foubert, A.M. McKee, R.A. Goubran, and F. Knoefel. Lying and sitting posture recognition and transition detection using a pressure sensor array. In *Medical Measurements and Applications Proceedings (MeMeA)*, pages 1–6, 2012.

[145] Yunyoung Nam, Yeesock Kim, and Jinseok Lee. Sleep monitoring based on a tri-axial accelerometer and a pressure sensor. *Sensors*, 16(5):750, 2016.

[146] Enamul Hoque and John A. Stankovic. Monitoring quantity and quality of sleeping using wisps. In *IPSN*, pages 370–371, 2010.

[147] J.J. Liu, Wenyao Xu, and et al. A dense pressure sensitive bedsheet design for unobtrusive sleep posture monitoring. In *PerCom*, pages 207–215, 2013.

[148] Yunyoung Nam, Yeesock Kim, and Jinseok Lee. Sleep monitoring based on a tri-axial accelerometer and a pressure sensor. volume 16, 2016.

[149] Abdulakeem Odunmbaku, Amir-Mohammad Rahmani, Pasi Liljeberg, and Hannu Tenhunen. *Elderly Monitoring System with Sleep and Fall Detector*, pages 473–480. Springer International Publishing, Cham, 2016.

[150] Oana Ramona Velicu, Natividad Martínez Madrid, and Ralf Seepold. Experimental sleep phases monitoring. In *Biomedical and Health Informatics (BHI), 2016 IEEE-EMBS International Conference on*, pages 625–628. IEEE, 2016.

[151] Zeo. `http://www.myzeo.com/sleep`, 2003.

[152] Tian Hao, Guoliang Xing, and Gang Zhou. isleep: Unobtrusive sleep quality monitoring using smartphones. In *SenSys*, pages 4:1–4:14, 2013.

[153] Muhammad Fahim, LeBa Vui, Iram Fatima, Sungyoung Lee, and Yongik Yoon. A sleep monitoring application for u-lifecare using accelerometer sensor of smartphone. In *Ubiquitous Computing and Ambient Intelligence. Context-Awareness and Context-Driven Interaction*, volume 8276, pages 151–158. 2013.

[154] Zhenyu Chen, Mu Lin, Fanglin Chen, N.D. Lane, and et al. Unobtrusive sleep monitoring using smartphones. In *PervasiveHealth*, pages 145–152, 2013.

[155] Nicholas D. Lane, Mashfiqui Mohammod, and et al. Bewell: A smartphone application to monitor, model and promote wellbeing. In *Pervasive Computing Technologies for Healthcare*, 2011.

[156] Yin Bai, Bin Xu, Yuanchao Ma, Guodong Sun, and Yu Zhao. Will you have a good sleep tonight?: Sleep quality prediction with mobile phone. In *Proceedings of the 7th International Conference on Body Area Networks*, 2012.

[157] Weixi Gu, Zheng Yang, Longfei Shangguan, Wei Sun, Kun Jin, and Yunhao Liu. Intelligent sleep stage mining service with smartphones. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 649–660, 2014.

[158] Mimo. http://mimobaby.com/, 2016.

[159] Sleep Time Smart Alarm Clock. `https://play.google.com/store/apps/details?id=com.azumio.android.sleeptime`, 2015.

[160] Beddit. `http://www.beddit.com/features/`, 2015.

[161] Hello Sense. `https://hello.is/`, 2014.

[162] Darkhovsky B Fell J Kaplan A, Röschke J. Macrostructural eeg characterization based on nonparametric change point segmentation: application to sleep analysis. In *Journal of Neuroscience Methods*, pages 81–90, 2001.

[163] Klerman EB Barbieri R. Citi L, Bianchi MT. Instantaneous monitoring of sleep fragmentation by point process heart rate variability and respiratory dynamics. In *Engineering in Medicine and Biology Society, 2011.*, 2011.

[164] Nikos Karampatziakis and John Langford. Online importance weight aware updates. In *Proceedings of the Twenty-Seventh Conference on Uncertainty (UAI) in Artificial Intelligence*, pages 392–399, 2011.

[165] Ryan Prescott Adams and David J.C. MacKay. Bayesian online changepoint detection. Cambridge, UK, 2007.

[166] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *ICML*, pages 49–56, 2009.

[167] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322, August 2010.

[168] Vowpal Wabbit. `http://hunch.net/~vw/`, 2016.

[169] Aeotec Multisensor. `http://aeotec.com/z-wave-sensor`, 2006.

[170] Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *ICML*, pages 350–358, 1998.

[171] Charles Elkan. Using the triangle inequality to accelerate k-means, 2003.

[172] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Large-scale text categorization by batch mode active learning. In *WWW*, pages 633–642, 2006.

[173] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.

[174] Rong Yan, Jie Yang, and Alexander Hauptmann. Automatically labeling video data using multi-class active learning. In *ICCV*, 2003.

[175] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[176] Cloudengines pogoplug. `https://pogoplug.com/`.

[177] Yan Yan, Glenn M Fung, Rómer Rosales, and Jennifer G Dy. Active learning from crowds. In *ICML*, pages 1161–1168, 2011.

[178] Weining Wu, Yang Liu, Maozu Guo, Chunyu Wang, and Xiaoyan Liu. A probabilistic model of active learning with multiple noisy oracles. *Neurocomput.*, 118:253–262, October 2013.

[179] Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar. Robust active learning using crowdsourced annotations for activity recognition. In *Human Computation*. AAAI.

[180] Sourav et al. Bhattacharya. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive Mob. Computing.*, 2014.

[181] Kyle D. Feuz and Diane J. Cook. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr). *ACM Trans. Intell. Syst. Technol.*, 2015.

[182] Nicholas D. etl al. Lane. Deepear: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proc. of ACM UbiComp*, 2015.

[183] Scott E. Reed et al. Training deep neural networks on noisy labels with bootstrapping. *CoRR*, 2014.

[184] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174, 2016.

[185] Adam Coates et al. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade - Second Edition, 2012*.

[186] Valentin Radu and Nicholas D. Lane. Towards multimodal deep learning for activity recognition on mobile devices. In *Proc. of UbiComp '16*.

[187] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Trans. Neural Netw. Learning Syst.*, 2014.

[188] Sourav Bhattacharya and Nicholas D. Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, pages 176–189, 2016.

[189] Sanjoy Dasgupta. Two faces of active learning. *Theor. Comput. Sci.*, 2011.

[190] Youngja Park and et al. Generating balanced classifier-independent training samples from unlabeled data. *Knowl. Inf. Syst.*, 2014.

[191] Charles Elkan. Using the triangle inequality to accelerate k-means.

[192] Tapas et al. Kanungo. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 2002.

[193] Honglak Lee and Alexis Battle. Efficient sparse coding algorithms. In *Advances in neural information processing systems, 2016*.

[194] Hiranmayi Ranganathan. *Deep Active Learning Explored Across Diverse Label Spaces*. PhD thesis, Arizona State University, Tempe, USA, 2018.

[195] H M Sajjad Hossain et al Joseph Taylor. Sensebox: A low-cost smart home system. In *PerCom Demo Workshop*, 2017.

[196] Torch: A scientific computing framework for luajit. `http://torch.ch/`, 2017.

[197] Daniel Roggen and Alberto Calatroni. Collecting complex activity datasets in highly rich networked sensor environments. In *Proc. of INSS*, 2010.

[198] Diane J Cook and Maureen Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 2009.

[199] Jennifer R. Kwapisz and et al. Activity recognition using cell phone accelerometers. *SIGKDD Explorations*, 2010.

[200] Marc Bächlin and et al. Wearable assistant for parkinson's disease patients with the freezing of gait symptom. *IEEE Trans. Information Technology in Biomedicine*, 2010.

[201] Umbc-umb partnership awards a catalyst for collaboration: Research at umbc news feb 17 2015. `http://research.umbc.edu/umbc-research-news/?id=49901`.

[202] David Martin Ward Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, 2011.

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

[204] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1943.

[205] Ivan Srba and Mária Bieliková. Tracing strength of relationships in social networks. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops, Toronto, Canada, August 31 - September 3, 2010*, pages 13–16, 2010.

[206] Yingce Xia, Tao Qin, and et al. Budgeted multi-armed bandits with multiple plays. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2210–2216, 2016.

[207] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[208] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 2015.

[209] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, 2014.

[210] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.