DeActive: Scaling Activity Recognition with Active Deep Learning

H M SAJJAD HOSSAIN, MD ABDULLAH AL HAFIZ KHAN, NIRMALYA ROY, SHIMEI PAN,
Department of Information Systems, University of Maryland Baltimore County

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 paper, 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.

ACM Reference format:
H M Sajjad Hossain, Md Abdullah Al Hafiz Khan, Nirmalya Roy, Shimei Pan. 2017. *DeActive*: Scaling Activity Recognition with Active Deep Learning. *PACM Interact. Mob. Wearable Ubiquitous Technol.* 0, 0, Article 0 (August 2017), 19 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Human activity recognition is becoming an ubiquitous need for many emerging smart environment applications such as remote heath monitoring, participatory sensing, interactive gaming, and green building energy management. Many algorithmic techniques in activity recognition (AR) literature such as sparse coding [8], transfer learning [29], active learning [20], deep learning [46] 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

 2474-9567/2017/8-ART0 $15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

environments and showcase the effectiveness of deep activity learning methodology when augmented with other simplistic popular machine 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 [25], fitting the activity models in presence of unlabeled activity samples and mitigating the biasness of overfitting distributions are still challenging research problems [26]. The unsupervised training and fine tuning phase of deep nets also warrant substantial computational resources and labeled data sources, respectively [27]. While shallow and supervised learning [42] suffer from representing well the large scale activity learning model, the hierarchical deep learning [43] 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 [17]. 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 [43]. Scaling the deep learning model for small footprint devices such as smartphone and smart wristwatches have also been investigated by exploiting different inference phases of deep model [57]. 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 pelthora of data in pervasive computing, these collections do not provide much information due to poorly or partially labeled. In order to improve the model increamentally, 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 work, 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. The key contributions of this work include:

- We propose a K-means clustering technique to represent and learn the features in the pre-training phase of auto-encoder in presence of unlabeled data and fine-tuning phase of output layers in presence of labeled data. Our proposed approach helps to bridge the continuation of generative and discriminative learning of deep activity models while reducing the resource overhead with improved classification performance.

- We propose an active learning technique in the fine tuning phase of the deep learning training process to accumulate the most informative and meaningful labeled samples. Also we label the instances with low uncertainty score by calculating the similarity indices between them and the most informative instances. The density weighted active learning based heuristic augmented with optimized classifier help scale the deep activity models.
- We evaluate our proposed, *DeActive* framework on real activity recognition data traces and a real smart home system, *SenseBox* which we built to validate that simplistic off-the-shelf machine learning algorithms augmented with deep activity models so that we can showcase competitive performance gain and less resource usage.

2 RELATED WORK

Activity Recognition using heterogeneous sensor and data sources has been investigated extensively over the past decade. In this section we contrast our model *DeActive* with other most relevant proposed models.

2.1 Activity Recognition

Inferring Activities of Daily Living (ADL) are approached from two perspective - vision based and sensor modality based. In vision based approach, researchers have exploited microphone and cameras to extract the performed human activity from audio and video or image data [24] [59]. A variety of sensor modalities like accelerometer, ambient sensors, RFID tags, Radar etc [40] [22] have been used in sensor based approach. The major upsurge of mobile and wearable technologies have also accelerated the activity recognition research [19] [51]. Bao and Intille [6] used five biaxial accelerometers which are placed in different parts of the body and detected 20 activities. Hafiz et al. [39] 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 [40]. The widely used features in activity recognition domain include basis transform coding (e.g.signals with wavelet transform and Fourier transform) [35], statistical parameters extracted from raw sensor signals [16] and symbolic representation [28]. 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 [16].

2.2 Deep Learning & Activity Recognition

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 architectures, researchers are now proposing activity recognition models using deep learning [44] [31] [42] [36] [55] [14]. The authors of [21] proposed a hybrid approach using HMM and deep learning to infer activities from triaxial accelerometer data. Francisco et al [18] 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 [25]. 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 [4]. *DeepEar* [45] a deep learning enabled mobile audio sensing framework addresses the issue of audio data classification. Li et al. [49]

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 [52]. The authors of [53] 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. [30] proposed a Long Short Term LSTM based ensemble model and tackled the problems of having imbalanaced and problematic data for wearable devices.

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] [20]. The authors of [50] [61] used uncertainty based active learning for activity recognition. The authors of [10] used entropy based measure to calculate the informativeness of activity data instances. Legion:AR [47], an activity recognition framework uses active learning and inquires for labels from crowd on demand. AALO [34] uses active learning for labeling overlapped activities. The authors of [3] validates that by using active learning the actual annotation cost can be reduced by 30-75%. [33] 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 [62] [64] [60]. Zhu et al. [32] 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 [63].

3 OVERALL DESIGN

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 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 [9], researches are currently focusing on accelerating deep learning on mobile devices [25] [7]. 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 [42] 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 [17] 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.

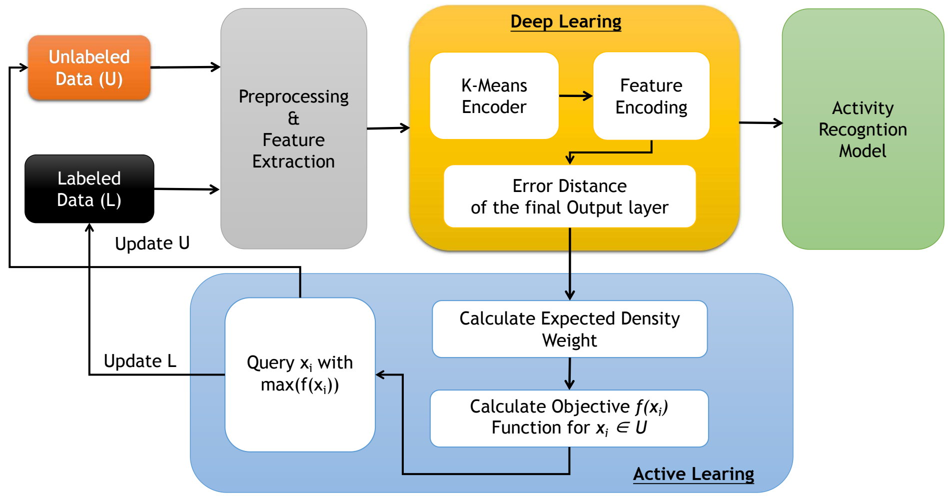


Fig. 1. *Overall framework for DeActive activity recognition model. Deep Learning phase is composed of k-means encoders. The output of k-means in the final layer is provided to the Active Learning phase which selects the most informative instances from the unlabeled data pool.*

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 [12]. 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 [54]. 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 1 we demonstrate our overall *DeActive* framework.

4 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 [17] 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.

4.1 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:

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

In most of the cases this error distance Eqn. 1 is minimized using heuristics like Loyd, Elkan etc [15]. 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 [38]. The objective function for k-means is $Q_{kmeans} = \min_k \frac{1}{2}(x - w_k)$. We calculate the gradient $\triangledown_{w_k} Q_{kmeans} = w_k - x_i$ by taking the partial derivative of the Objective function. After this we update the learning rate η and weight vector w according to Eqns. 2 and 3. The whole process iterates until the cluster centers are no longer changing.

$$\eta_k = \eta_k + 1 \tag{2}$$

$$w_k = w_k + \frac{1}{\eta_k}(x - w_k) \tag{3}$$

4.2 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:

$$minimize \sum_i \|D_s^{(i)} - x^{(i)}\|_2^2 \tag{4}$$

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

In Eqn. 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. 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 [48]. 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:

$$s_j^{(i)} = \begin{cases} D^{(j)T} x^{(i)}, & \text{if } j == \text{argmax}_l \, |D^{(l)T} x^{(i)}| \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Using Eqn. 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.

4.3 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:

$$\begin{aligned} D &= \underset{D}{\text{argmin}} \, ||DS - X||_2^2 + ||D - D_{old}||_2^2 \\ &= (SS^T + I)^{-}1(XS^T + D_{old}) \approx XS^T + D_{old} \end{aligned} \tag{6}$$

4.4 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. 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. 8).

$$f_k^1(x) = \begin{cases} 1 & \text{if } k == \text{argmin}_j \, ||c^{(j)-x}||_2^2 \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

$$f_k^2(x) = max\{0, \mu(z) - z_k\} \tag{8}$$

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

$$f(x) = \underset{x}{\operatorname{argmax}}\, \Phi(x) \times \left(\frac{1}{card(U)} \sum_{x \in U} sim(x, x^*) \right)^{\beta} \tag{9}$$

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 1 we get our loss function as following:

$$L(x) = \operatorname{argmin} \sum_{i=1}^{n} \|x - x^*\| \tag{10}$$

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:

$$\begin{aligned} \Phi(x) &= \underset{x}{\operatorname{argmin}}\, E_{y|x}[L(x)] \\ &= \underset{x}{\operatorname{argmin}} \sum_{y} P(y|x)[\sum_{i=1}^{n} \|x - x^*\|] \end{aligned} \tag{11}$$

The term $\left(\frac{1}{card(U)} \sum_{x \in U} sim(x, x^*) \right)^{\beta}$ in Equation 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 $n(i)$ 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 $n(i)$ is said to be the *neighboring cluster* of the cluster where x_i resides. Now we define our silhouette coefficient as following:

$$s_c^i = \frac{e(i) - d(i)}{\max\{d(i), e(i)\}} \tag{12}$$

$$s_c^i = \begin{cases} 1 - \frac{d(i)}{e(i)}, & \text{if } d(i) < e(i) \\ 0, & \text{if } d(i) = e(i) \\ \frac{e(i)}{d(i)} - 1, & \text{if } d(i) > e(i) \end{cases} \tag{13}$$

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

$$f(x) = \operatorname*{argmax}_x \, [s_c^i \, \Phi(x) \times \left(\frac{1}{card(U)} \sum_{x \in U} sim(x, x^*) \right)^{\beta}] \tag{14}$$

The overall active learning strategy of our *DeActive* model is summarized in Algorithm 1.

ALGORITHM 1: *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 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 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
$L = l + I$

6 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.

6.1 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 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.

6.2 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).

6.3 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: $z = \frac{x-\mu}{\delta}$.

7 SENSEBOX IMPLEMENTATION

We collected real life data using our *SenseBox* [37] 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 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

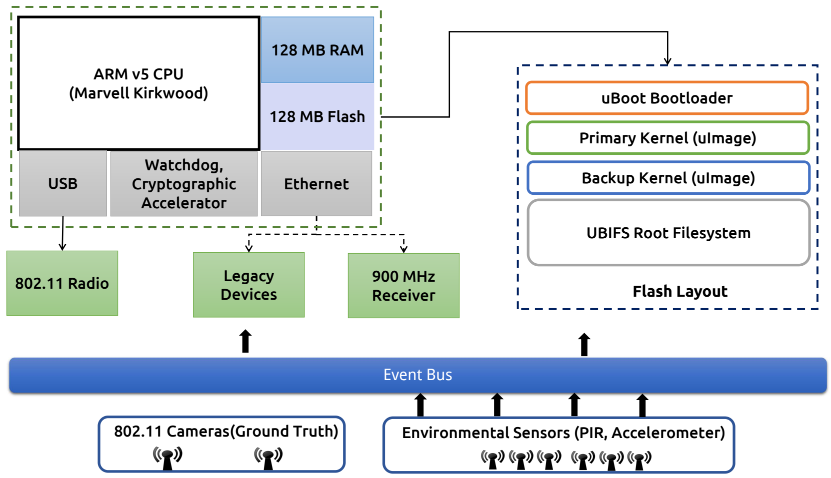


Fig. 2. *SenseBox architecture which has ARM v5 CPU. Multimodal sensors dump the streaming data in the Event Bus and the system reads the new data from there.*

fixing several issues before the kernel was able to be successfully built and stable. As our deep learning algorithm uses torch [2] library, we have built torch for ARM processor.

8 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 [58] 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 [11] 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 [41] 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 [5] 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* [37] we collected data from 10 participants [1] (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 [56]. 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.

8.1 Performance Analysis

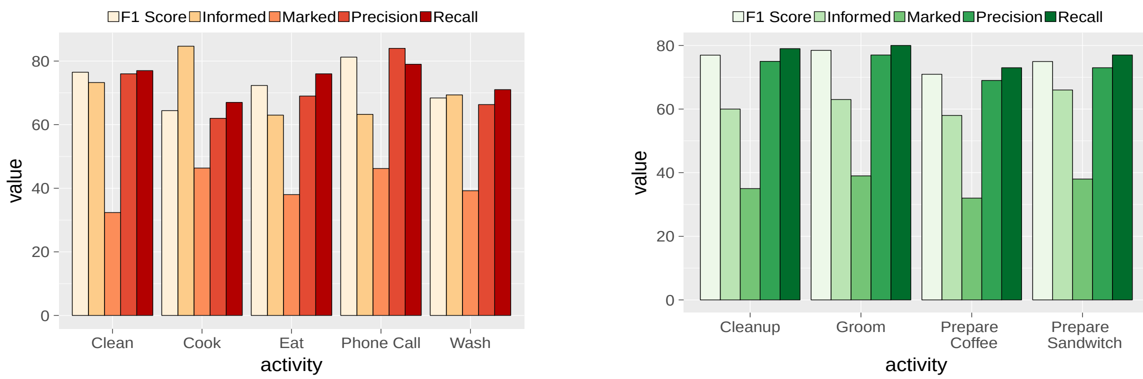
In this section, we examine the performance of our *DeActive* model on real world datasets described in the previous section. In our proposed model we use 3 hidden layers with 150 neurons in each layer. We first normalize the data with zero mean and standard variance. The deep activity recognition models are trained using stochastic gradient decent with mini-batch size of 0.60. In Table 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 1. *Accuracy of different deep architectures on SenseBox dataset*

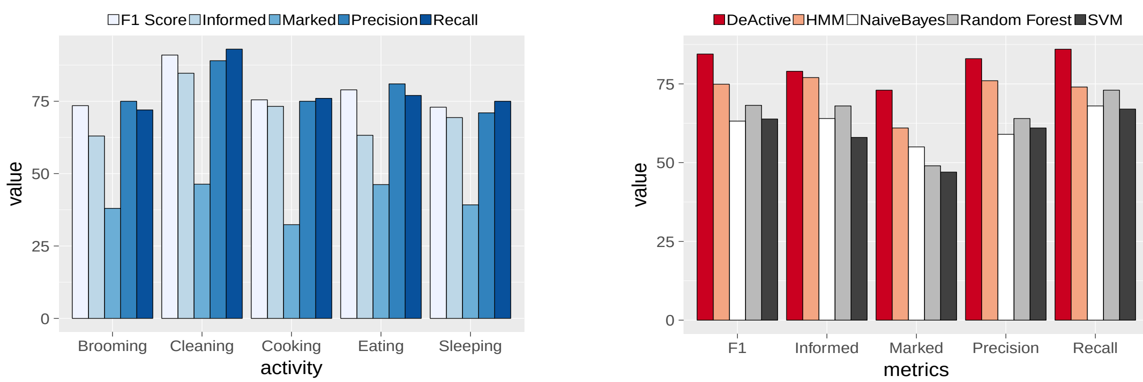
8.2 Classification Accuracy

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 3 and 4 . For *Opportunity* dataset (Fig 3b), 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



(a) *CASAS* ***(b)*** *Opportunity*

Fig. 3. *Precision, recall, F1, informedness and markedness score of each activity in CASAS and Opportunity datasets (ambient motion sensor data).*

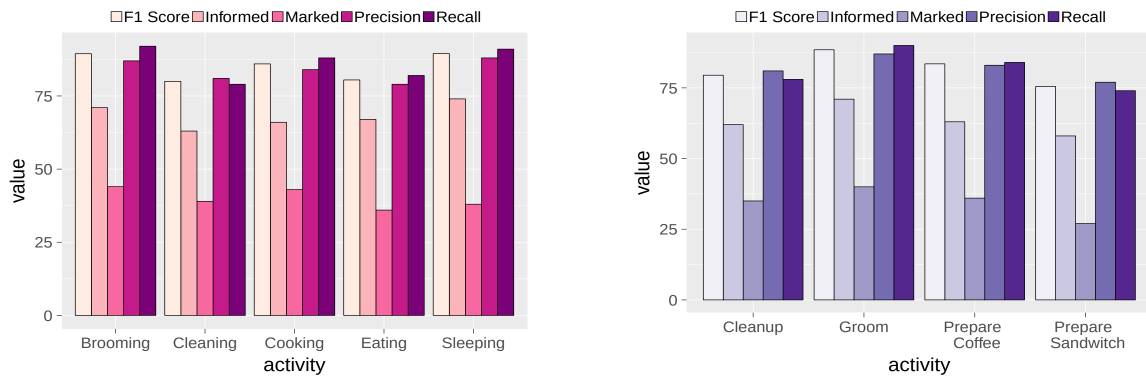


(a) *SenseBox* ***(b)*** *Overall*

Fig. 4. *(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.*

cooking, eating sleeping and brooming 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 *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 10 shows the convergence of accuracy with respect to the percentage of dataset used in the experiment.

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 6 and 5. Each dataset showed much better



(a) SenseBox *(b) Opportunity*

Fig. 5. *Precision, recall, F1, informedness and markedness score of SenseBox and Opportunity datasets (3D acceleration data).*

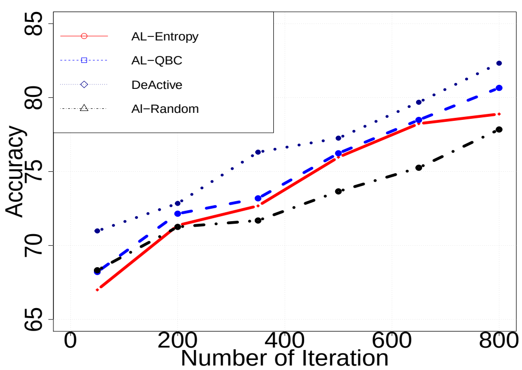
(a) WISDM *(b) Skoda*

Fig. 6. *Precision, recall, F1, informedness and markedness score of WISDM and Skoda datasets (3D acceleration data).*

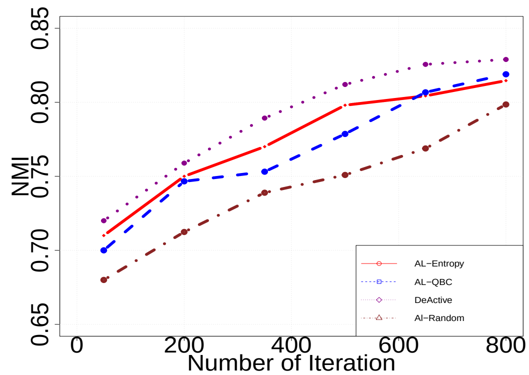
Activity Recognition System	Skoda	Opportunity	WISDM	SenseBox
Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition [18]	95.8	91.20	95.86	88.09
Convolutional Neural Networks for human activity recognition using mobile sensors [23]	88.19	93.17	94.75	84.20
Deep Activity Recognition Models with Triaxial Accelerometers [21]	89.38	86.39	94.46	87.54
Our deep learning framework *DeActive*	92.34	94.06	97.24	92.34

Table 2. *Comparison of our DeActive algorithm with other existing approaches for different datasets.*

accuracy when accelerometers 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 5a) achieved much better accuracy than using just the ambient



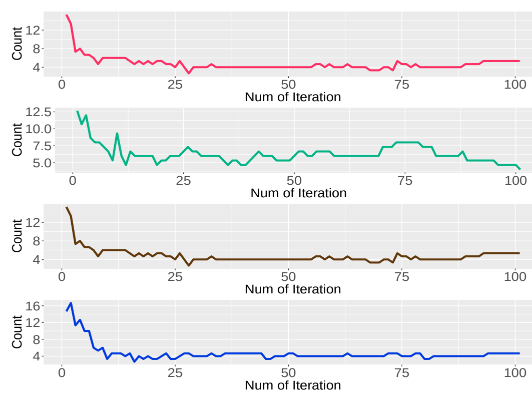
***(a)** The change in model accuracy after each iteration using different active learning strategies.*



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

***Fig. 7.** The figures demonstrate the performance of our active learning algorithm.*

***(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.*

***Fig. 8.** The figures demonstrate the performance of our active learning algorithm.*

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 *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.). 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

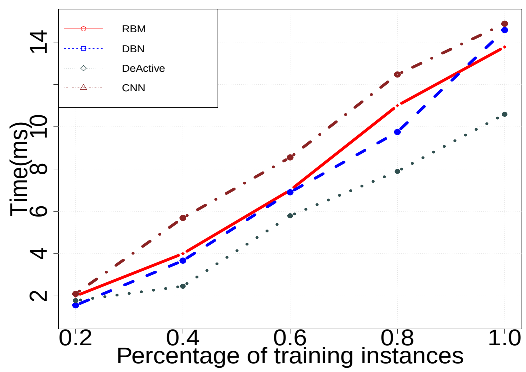


Fig. 9. The execution time of DeActive in SenseBox architecture.

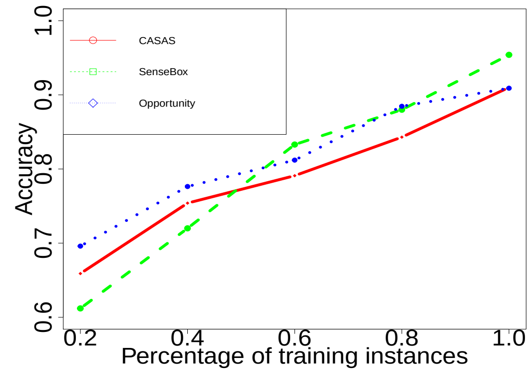


Fig. 10. Convergence of Accuracy with respect to percentage of data instances.

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 (≈ 92%) than other datasets as the activities considered have distinct signature pattern in the accelerometer data. In Table 2 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.

8.3 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 7a 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 7b 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 8b. 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 8a we show the progression of instance selection time for the first 100 iterations for all active learning strategies.

8.4 Device Performance

We investigate the performance of our *DeActive* model in *SenseBox* architecture. In Figure 9 we see that our algorithm executes much faster than other algorithms. In [25], 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 [25] used Snapdragon 400 quad core CPU whereas we used single core CPU.

9 CONCLUSION

Scaffolding the activity recognition to many emerging smart environment applications is a pressing societal need. While advanced machine learning approaches albeit help achieve that objective, in this paper, we envision that simple algorithms can be considered as viable and sometime competitive alternatives along that pathway. Motivated by this, we presented an activity recognition model using a simple K-means clustering assisted deep architecture which help scale activity recognition in large. Our proposed model also consolidated active learning to mitigate the amount of human effort needed for collecting ground truth information. We compared our activity model with other deep and active learning algorithms and validated that our model can outperform them. We built a custom-made smart home system, *Sensebox* and demonstrated that the competence and viability of our model through real deployment in retirement community center. We believe our work is the first step towards enabling the vision that *Simplicity is the Scalability*.

REFERENCES

[1] 2015. UMBC-UMB Partnership Awards a Catalyst for Collaboration. http://research.umbc.edu/umbc-research-news/?id=49901. (2015).
[2] 2017. Torch: A scientific computing framework for LuaJIT. http://torch.ch/. (2017).
[3] Hande Alemdar, TLM van Kasteren, and Cem Ersoy. 2017. Active learning with uncertainty sampling for large scale activity recognition in smart homes. *Journal of Ambient Intelligence and Smart Environments* 9, 2 (2017), 209–223.
[4] Moez Baccouche and Franck Mamalet. Sequential Deep Learning for Human Action Recognition. In *Proc. of IEEE HBU, 2011.*
[5] Marc Bächlin and et al. 2010. Wearable assistant for Parkinson's disease patients with the freezing of gait symptom. *IEEE Trans. Information Technology in Biomedicine* (2010).
[6] Ling Bao and Stephen S. Intille. 2004. Activity Recognition from User-Annotated Acceleration Data. In *Proc.of PERVASIVE.*
[7] Sourav Bhattacharya and Nicholas D. Lane. 2016. 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).* 176–189.
[8] Sourav et al. Bhattacharya. 2014. Using Unlabeled Data in a Sparse-coding Framework for Human Activity Recognition. *Pervasive Mob. Computing.* (2014).
[9] Monica Bianchini and Franco Scarselli. 2014. On the Complexity of Neural Network Classifiers: A Comparison Between Shallow and Deep Architectures. *IEEE Trans. Neural Netw. Learning Syst.* (2014).
[10] Yu chen ho and Ching hu lu. 2009. Active-learning assisted self-reconfigurable activity recognition in a dynamic environment. In *ICRA.*
[11] Diane J Cook and Maureen Schmitter-Edgecombe. 2009. Assessing the quality of activities in a smart environment. *Methods of information in medicine* (2009).
[12] Sanjoy Dasgupta. 2011. Two faces of active learning. *Theor. Comput. Sci.* (2011).
[13] Tom Diethe, Niall Twomey, and Peter Flach. 2015. Bayesian Active Transfer Learning in Smart Homes. In *Proc. of ICML.*
[14] Tushar Dobhal and Vivswan Shitole. 2015. Human Activity Recognition using Binary Motion Image and Deep Learning. *Procedia Computer Science* (2015).
[15] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proc. of ICML.*
[16] Andreas Bulling et al. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.* (2014).
[17] Adam Coates et al. Learning Feature Representations with K-Means. In *Neural Networks: Tricks of the Trade - Second Edition, 2012.*
[18] Francisco Javier et al. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* (2016).
[19] Gina Sprint et al. 2016. Unsupervised detection and analysis of changes in everyday physical activity data. *Journal of Biomedical Informatics* (2016).
[20] H. M. Sajjad Hossain et al. 2016. Active learning enabled activity recognition. In *IEEE PerCom.*
[21] Mohammad Abu Alsheikh et al. 2016. Deep Activity Recognition Models with Triaxial Accelerometers. *AAAI Workshop* (2016).
[22] Michalis Vrigkas et al. 2015. A Review of Human Activity Recognition Methods. *Front. Robotics and AI* (2015).
[23] Ming Zeng et al. 2014. Convolutional Neural Networks for human activity recognition using mobile sensors. In *MobiCASE.*

[24] Raj Kumar Gupta et al. 2013. Human activities recognition using depth images. In *Proc. of ACM Multimedia Conference.*

[25] Sourav Bhattacharya et al. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *Proc. of PerCom Workshops 2016.*

[26] Scott E. Reed et al. 2014. Training Deep Neural Networks on Noisy Labels with Bootstrapping. *CoRR* (2014).

[27] Tianqi Chen et al. 2016. Training Deep Nets with Sublinear Memory Cost. *CoRR* (2016).

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

[29] Kyle D. Feuz and Diane J. Cook. 2015. Transfer Learning Across Feature-Rich Heterogeneous Feature Spaces via Feature-Space Remapping (FSR). *ACM Trans. Intell. Syst. Technol.* (2015).

[30] Yu Guan and Thomas Ploetz. 2017. Ensembles of Deep LSTM Learners for Activity Recognition using Wearables. *arXiv preprint arXiv:1703.09370* (2017).

[31] Nils Y. Hammerla and Shane Halloran. Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. In *Proc. of IJCAI 2016.*

[32] Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. *CoRR* abs/1606.03476 (2016).

[33] Yu-chen Ho, Ching-hu Lu, I-han Chen, Shih-shinh Huang, Ching-yao Wang, and Li-chen Fu. 2009. 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).* IEEE Press, Piscataway, NJ, USA, 1567–1572. http://dl.acm.org/citation.cfm?id=1703435.1703688

[34] E. Hoque and J. Stankovic. 2012. AALO: Activity recognition in smart homes using Active Learning in the presence of Overlapped activities. In *Proc. of PervasiveHealth.*

[35] Tâm Huynh and Bernt Schiele. 2006. Unsupervised Discovery of Structure in Activity Data Using Multiple Eigenspaces. In *Proc. of LoCA.*

[36] Wenchao Jiang and Zhaozheng Yin. Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks. In *Proc. of ACM MM 2015.*

[37] H M Sajjad Hossain et al Joseph Taylor. 2017. SenseBox: A Low-Cost Smart Home System. In *PerCom Demo Workshop.*

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

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

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

[41] Jennifer R. Kwapisz and et al. 2010. Activity recognition using cell phone accelerometers. *SIGKDD Explorations* (2010).

[42] N. D. Lane and S. Bhattacharya. 2016. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In *Proc. of IPSN.*

[43] Nicholas D. Lane and Petko Georgiev. Can Deep Learning Revolutionize Mobile Sensing?. In *Proc. of HotMobile '15.*

[44] Nicholas D. Lane and Petko Georgiev. Can Deep Learning Revolutionize Mobile Sensing?. In *Proc. of HotMobile 2015.*

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

[46] Nicholas D. etl al. Lane. 2015. DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments Using Deep Learning. In *Proc. of ACM UbiComp.*

[47] Walter S. Lasecki and Young Chol Song. 2013. Real-time Crowd Labeling for Deployable Activity Recognition. In *Proc. of CSCW.*

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

[49] Xinyu et al. Li. Deep Learning for RFID-Based Activity Recognition *(SenSys '16).*

[50] Rong Liu, Ting Chen, and Lu Huang. 2010. Research on human activity recognition based on active learning. In *Proc. of ICMLC.*

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

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

[53] Francisco Javier Ordóñez Morales and Daniel Roggen. 2016. 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).* 92–99.

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

[55] Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. 2011. Feature Learning for Activity Recognition in Ubiquitous Computing. In *Proc. of IJCAI 2011.*

[56] David Martin Ward Powers. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology* (2011).

[57] Valentin Radu and Nicholas D. Lane. Towards Multimodal Deep Learning for Activity Recognition on Mobile Devices. In *Proc. of UbiComp '16.*

[58] Daniel Roggen and Alberto Calatroni. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *Proc. of INSS*.
[59] Paul E. Rybski and Manuela M. Veloso. 2005. Robust Real-Time Human Activity Recognition from Tracked Face Displacements. In *Proc. of EPIA, 2005*.
[60] Fabian Stark and Rudolph Triebel. 2015. CAPTCHA Recognition with Active Deep Learning. (2015).
[61] M. et al. Stikic. 2008. Exploring semi-supervised and active learning for activity recognition. In *Proc. of ISWC*.
[62] Dan Wang and Yi Shang. A new active labeling method for deep learning. In *Proc. of IJCNN 2014*.
[63] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. 2017. Cost-Effective Active Learning for Deep Image Classification. *CoRR* abs/1701.03551 (2017).
[64] Shusen Zhou and Qingcai Chen. Active Deep Networks for Semi-Supervised Sentiment Classification. In *Proc. of COLING*.