Active Deep Learning for Activity Recognition with Context Aware Annotator Selection

H M Sajjad Hossain, Nirmalya Roy
riaj.sajjad@umbc.edu,nroy@umbc.edu
University of Maryland Baltimore County

ABSTRACT

Machine learning models are bounded by the credibility of *ground truth* data used for both training and testing. Regardless of the problem domain, this ground truth annotation is objectively manual and tedious as it needs considerable amount of human intervention. 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. Due to limited amount of ground truth information addressing the variabilities of Activity of Daily Living (ADLs), activity recognition models using wearable and mobile devices are still not robust enough for real-world deployment. In this paper, we first propose an active learning combined deep model which updates its network parameters based on the optimization of a joint loss function. We then propose a novel annotator selection model by exploiting the relationships among the users while considering their heterogeneity with respect to their expertise, physical and spatial context. 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 ≈8% improvement in accuracy in fewer iterations.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**; **Multi-agent planning**; • **Information systems** → *Personalization*.

KEYWORDS

active learning, annotator selection, activity recognition

ACM Reference Format:
H M Sajjad Hossain, Nirmalya Roy. 2019. Active Deep Learning for Activity Recognition with Context Aware Annotator Selection. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330688

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330688

1 INTRODUCTION

Mobile and wearable technologies are exhibiting manifestation of sustained markets and new industries ranging from healthcare, sports analytics to entertainment [24]. These ubiquitous devices armed with plethora of sensing capabilities can capture longitudinal physiological, behavioral and environmental data spurring fundamental understanding of Activities of Daily Living (ADLs). Most of the prior works involving ADL recognition are heavily enlightened based on supervised learning [4]. In order to train a robust predictive model for ADL, multitude of data with possible variations are indispensible. It is important to capture the different variations of a certain activity with respect to different constraints. For example, if we were to build a generalized predictive model for detecting "cooking", several constraints like cultural diversity, living condition, food preferences etc. need to be considered. Noise and variations in the data drive the convergence properties of model parameters. As a result the accuracy of these models are bounded by the inherent quality of the labeled data provided during training phase. In order to tackle this problem, large amount of labeled data originating from diverse environment with context driven constraints are fundamental. Deep learning algorithms have recently become popular because of their intrinsic nature of learning good feature representations given input data with complex patterns [23] [21]. Nevertheless, deep models still require large labeled dataset to tune their parameters and adapt to the instances with distinctive scenarios.

Collecting labeled data information in disparate situations is time-consuming and demands considerable amount of manual labor. In activity recognition domain, researchers have attempted to collect ground-truth information using modalities like video camera, object sensors and visual observations. This data annotation process is usually carried out by domain experts. It is difficult to manage enough domain experts to label large amount of data. Using crowdsourcing, we can manage a large number of annotators but the lack of their domain knowledge influences the quality of the labels. Active learning algorithms have also exploited to collect labels from the users themselves [10] [2]. The key property of active learning is that it enables us to focus only on the data instances which will have significant impact on the model parameters if labels are provided [28]. Traditional active learning is based on single annotator and assumes any number queries can be posed to the annotator at any time. Moreover, the algorithm expects accurate annotation all time which is inappropriate in a real-world settings.

In this paper 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[5] [29] 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. The potential annotators for a certain user are accumulated based on the similarity of their context information. The key insight here is that the potential annotators are those with whom we interact with mostly in our day to day life. As we tend to follow a cognitive routine everyday (having meals, going to work, playing, sleeping etc., at specific times of the day), the people with whom we are connected with may have a general understanding of our habits and timing or can be direct activity witnesses. For this reason, we encompass these people as potential annotators for an user in our model. We then attempt to select the appropriate annotator given the context of the user and the annotators. Hence we propose a novel annotator selection model using deep reinforcement learning algorithm. The main contributions of the paper are summarized below:

- We propose an active learning integrated deep activity recognition model which shows near optimal accuracy in presence of fewer labeled instances.
- We propose a joint loss function which helps us to optimize the parameters of the deep model according to the data distribution of the unlabeled data instances. This enables our model to adapt to outliers and be more sensitive to *true* informative instances.
- We propose a novel annotator selection model using deep reinforcement learning. We acclimate an actor-critic network based approach to learn the reward distribution of the annotators in a continuous state space.
- We develop a smartphone app (SocialAnnotator) to collect a variety of labeled data from 20 users over a month. We evaluate our proposed active-deep reinforcement learning based contextual annotator selection framework using these datasets which attest faster convergence to optimal reward.

2 RELATED WORK

Human activity recognition (HAR) using variety of sensing entities (wearables, mobile, RFID, WiFi, camera etc.) have been investigated extensively in recent years [4][24]. Earlier HAR models based on shallow learning classifiers fail to scale for larger population and diverse environment, as they need handcrafted feature extracted from the experts. Also they require large number of labeled data which is time consuming and most of the time manual process. With the recent advancements of deep models, these limitations have been somewhat mitigated [21]. The authors of [27] proposed a personalized HAR system based on matching networks which can recognize a new class from a few samples of that class. Using matching networks they perform nearest-neighbor classification by reusing the class label of the most similar instances in a provided support set. K-means clustering inspired deep encoding method for feature extraction has been proposed in [11]. A hierarchical deep multi-task learning model to detect simple and complex activities , AROMA [22] utilizes convolutional neural network(CNN) to extract features and then applies LSTM to learn the temporal properties of the activities. Various researchers [20][23][17][23] 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 [8]. The authors of [15] 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 [14][25][13]. Khan et al. have demonstrated a CNN inspired transductive transfer learning model, HDCNN [14] which can attain high accuracy in absence of any labeled information in the target domain. The authors of [32] 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 [26].

Although deep learning and transfer learning models can handle and endure the labeled data scarcity, still they require significant amount of labeled instances. Active learning algorithms has been investigated in online settings to opportunistically collect labels from the users [5][10][29][2][1][11][9]. Gabriele et al. [5] proposed a collaborative active learning method to extract the correlations among sensor events and activity types. However, the effectiveness of active learning algorithms largely depends on the annotators and the quality of label received. [7] [6] [33] [34] have proposed annotator selection models in multi annotator active learning settings. In our earlier work [12], we employed contextual multi armed bandit to model the reward dynamics. However the rewards for individual annotators were sampled from a pre-defined distribution using Monte Carlo simulation. In this paper, we want to make the reward calculation process as functional approximation so that the system becomes dynamic to different context. Existing proposed HAR models based on active learning, treats the active learning process as a separate pipeline. In this paper, we propose an active learning infused deep model which ensures convergence to optimal accuracy with fewer labeled examples. We also propose an annotator selection model based on reinforcement learning to ensure that active learning remains effective.

3 OVERALL

Our model has two major components - an active-deep classifier and an annotator selection model to make active learning more effective. In Figure 1 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 parameters. The posterior probabilities of the class labels are then collected from the final softmax layer. An active learning sampling

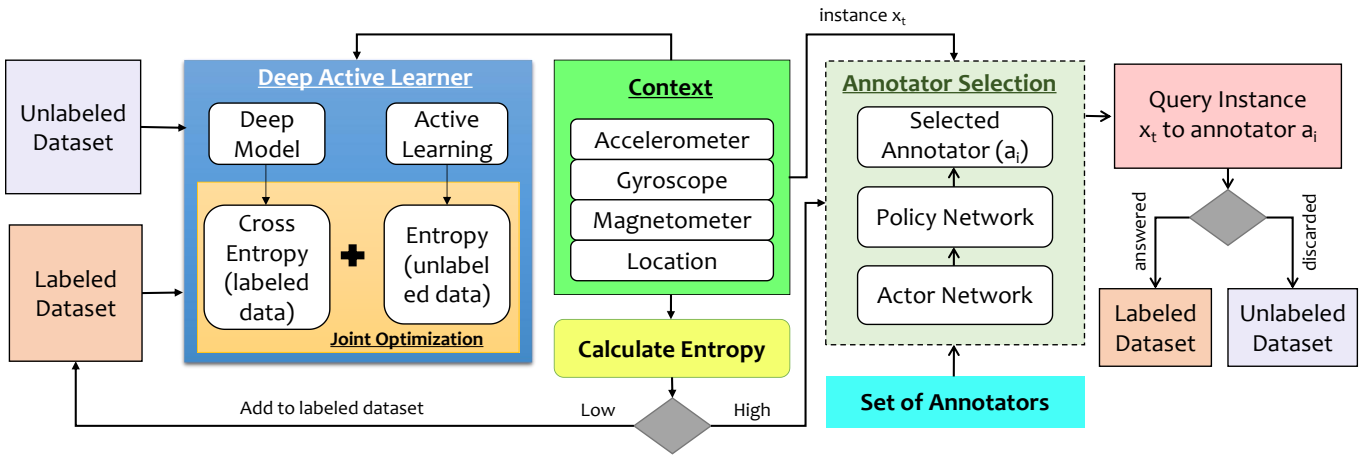


Figure 1: A High level architecture of our proposed model. The left side of the figure illustrates our active learning enabled deep model and in the right side of the figure our annotator selection pipeline is shown.

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. We select appropriate annotator for a certain query from a set of annotators who are connected to the user. The key intuition here is that in our day to day life some activities have direct witnesses. These witnesses are usually the people who we mostly know and are connected or related with. In our annotator selection module we take advantage of these connections and consider them as the potential activity annotators. We employ deep reinforcement learning approach in order to determine the best annotator among the connected users given an instance. Our annotator selection 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 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.

4 DEFINITIONS & PRELIMINARIES

Majority of the real-world problems involving reinforcement learning are modeled in accordance to Markov decision processes (MDP) [31]. 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

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

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:

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi[G_t | s_t = s, a_t = a] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s, a_t = a] \end{aligned} \tag{2}$$

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

$$\begin{aligned} v_*(s_t) &= \max_\pi v_\pi(s) \\ &= \max_{a_i} \sum_{s_{t+1}, r} p(s_{t+1}, r | s_t, a_i)[r + \gamma v_*(s_{t+1})] \end{aligned} \tag{3}$$

$$\begin{aligned} q_*(s, a) &= \mathbb{E}[r_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a') | s_t = s, a_t = a] \\ &= \sum_{s_{t+1}, r | s_t, a} [r + \gamma \max_{a'} q_*(s_{t+1}, a')] \end{aligned} \tag{4}$$

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 4 and 3. Our goal is then to find a good approximate solution given the circumstantial prior encounters.

4.1 Deep Reinforcement Learning

The deep deterministic policy gradient (DDPG) [16] method based on deterministic policy gradient (DPG) [30] 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

$$L(\theta_q) = \mathbb{E}_{s, a \sim \rho(.)}[(y_i - q(s_t, a_t; \theta_q^i)^2] + w_{\theta_q^i} {\theta_q^i}^2 \tag{5}$$

$$y_i = \mathbb{E}[r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a'; \theta_q^{i-1} | s_t, a)] \tag{6}$$

The parameters θ_q are then updated using the batch stochastic gradient descent method in Eqn. 7 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.

$$\theta_q \rightarrow \theta_q - \alpha_q \nabla_{\theta_q} L(\theta_q) \tag{7}$$

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. The average reward received is defined as:

$$\begin{aligned} J(\theta_\pi) &= \lim_{h \rightarrow \infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[r_t | s_0, a_{0:t-1} \sim \pi] \\ &= \lim_{t \rightarrow \infty} \mathbb{E}[r_t | s_0, a_{0:t-1} \sim \pi] \\ &= \sum_s \mu_\pi(s) \sum_a \pi(a | s_t, \theta_\pi) \sum p(s_{t+1}, r | s_t, a) r \end{aligned} \tag{8}$$

In Eqn 8 s_0 is our initial state and μ_π is the steady-state distribution under policy π, $\mu_\pi(s) = \lim_{t \rightarrow \infty} Pr(s_t = s | a_{0:t} \sim \pi)$. 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.

$$\theta_\pi \rightarrow \theta_\pi - \alpha_\pi \nabla_{\theta_\pi} L(\theta_\pi) \tag{9}$$

5 METHODOLOGY

In this section we describe our proposed methodology in details. As mentioned in the Section 3, our algorithm has two fundamental modules, *Deep Active Learner* and *Annotator Selection*. First we will describe our activity classification and instance selection using active learning pipeline.

5.1 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. 10) to optimize the network parameters. In Eqn. 10 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.

$$L_c = -\frac{1}{n_l} \sum_{i=1}^{N} [c_i \log(y_i)] \tag{10}$$

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,

$$H(y_i) = -\sum_{j=1}^{N} y_i^j \log y_i^j \tag{11}$$

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 propose a joint loss function which is defined as following

$$\begin{aligned} L &= \frac{1}{n_l}\sum_{i=1}^{n_l} L(c_i, y_i) + \frac{\lambda}{n_u}\sum_{i=n_l+1}^{n} H(y_i) \\ &= -\frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{i=1}^{N}[c_i \log(y_i)] - \frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{N} y_i \log y_i^j \end{aligned} \tag{12}$$

In Eqn. 12, 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 12 by evaluating the partial derivative of the cross-entropy loss and the entropy. The gradient of the cross-entropy loss is

$$\begin{aligned} \frac{\partial L_c}{\partial h_{qr}} &= -\frac{\partial}{\partial h_{qr}}\left(\frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{i=1}^{N}[c_i \log(y_i)]\right) \\ &= \frac{\partial}{\partial h_{qr}}\left(\frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{i=1}^{N}[c_i \log(\frac{e^{p_i}}{\sum_k^N e^{p_k}})]\right) = \frac{1}{n_l}(p_{qr} - y_q) \end{aligned} \tag{13}$$

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

$$\begin{aligned} \frac{\partial H}{\partial h_{qr}} &= -\frac{\partial}{\partial h_{qr}}\left(\frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{N} x_i \log y_i^j\right) \\ &= -\frac{\partial}{\partial h_{qr}}\left(\frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{N}\frac{e^{p_i}}{\sum_k^N e^{p_k}} \log \frac{e^{p_i^j}}{\sum_k^N e^{p_k}}\right) \\ &= -\frac{\lambda}{n_u}\sum\sum(p_{ij}\log\sum_{j'} e^{h_{ij'}} + p_{ij}p_{ir} - p_{ij}\log\sum_{j'} e^{h_{ij'}}p_{ir}) \\ &= \frac{\lambda}{n_u}\Bigg[- p_{qr}h_{qr} - p_{qr} + p_{qr}\sum_{j=1}^{C} p_{qj}h_{qj} + p_{qr}\log\sum_{j'} e^{h_{qj'}} \\ &+ p_{qj}p_{qr} - \log\sum_{j'} e^{h_{qj'}}p_{qr}\sum_{j=1}^{C} p_{qj}\Bigg] \\ &= \frac{\lambda}{n_u}p_{qr}\left[\sum_{j=1}^{C} p_{qj}h_{qj} - h_{qr}\right] \end{aligned} \tag{14}$$

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

$$\frac{\partial L}{\partial h_{qr}} = \frac{1}{n_l}(p_{qr} - y_q) + \frac{\lambda}{n_u}p_{qr}\sum_{j=1}^{C} p_{qj}h_{qj} - h_{qr} \tag{15}$$

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 1. 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.

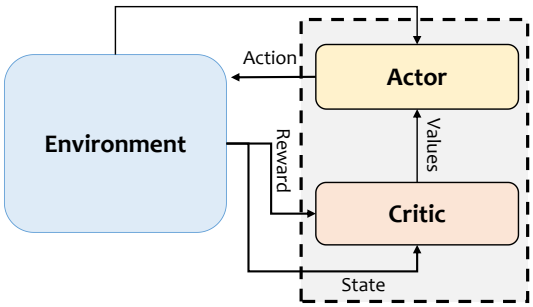


Figure 2: Actor critic network flow

Algorithm 1 Deep Active Learning

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

5.2 Annotator Selection

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 2. Taking an action is analogous to selecting an annotator from a finite set 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. 8. The gradient is defined as following:

$$\begin{aligned} \nabla_\pi J(\theta_\pi) &= \nabla_\pi\left[\sum_s \mu_\pi(s)\sum_a \pi(a|s_t, \theta_\pi)\sum p(s_{t+1}, r|s_t, a)r\right] \\ &= \sum_s \mu(s)\sum_a \nabla\pi(a|s)q_\pi(s, a) \end{aligned} \tag{16}$$

While training the $actor - critic$ networks, the loss function defined Eqn. 6 and the gradient calculation in 16 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. 7 and 9. We exploit the method proposed by [16] to fix the problem of divergence when directly setting the parameters of the target function in Eqn. 6 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.

Algorithm 2 Annotator Selection Policy using DDPG

Input: activity fragments F_k, $k = 1 \ldots K$ with associated annotator fragments (s_t, a_i^t)
Input: parameterized policy $\pi(a_i^t | s_t, \theta_\pi)$
Input: parameterized state-value function $v(s_t, \theta_q)$
initialize state-value parameter θ_q and policy parameter θ_π to 0
initialize target function $\hat{\theta}_q \leftarrow \theta_q$ and $\hat{\theta}_\pi \leftarrow \theta_\pi$
initialize starting state s_0
for each time step t **do**
  Select annotator a_j^t using current policy π
  Evaluate state-value function $q(s_t, a_j^t, \theta_q)$
  Evaluate expected state-value y_i using optimum annotator a_i^t
  Calculate the loss of $critic$ network $L(\theta_q)$
  Calculate the average reward rate $J(\theta_\pi)$ for state s_t
  Evaluate the gradients $\nabla L(\theta_q)$ and $\nabla J(\theta_\pi)$
  Update θ_q and θ_π
  $\hat{\theta_q} \leftarrow (1-\eta)\hat{\theta_q} + \eta\hat{\theta_q}$
  $\hat{\theta_\pi} \leftarrow (1-\eta)\hat{\theta_\pi} + \eta\hat{\theta_\pi}$
  Update state $s_t \leftarrow s_{t+1}$
end for

6 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?

6.1 System Implementation

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 on both the platforms are shown in Figure 3. 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"

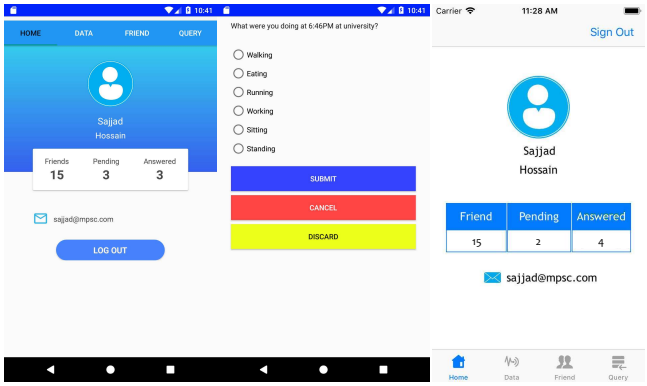


Figure 3: SocialAnnotator application interface.

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.

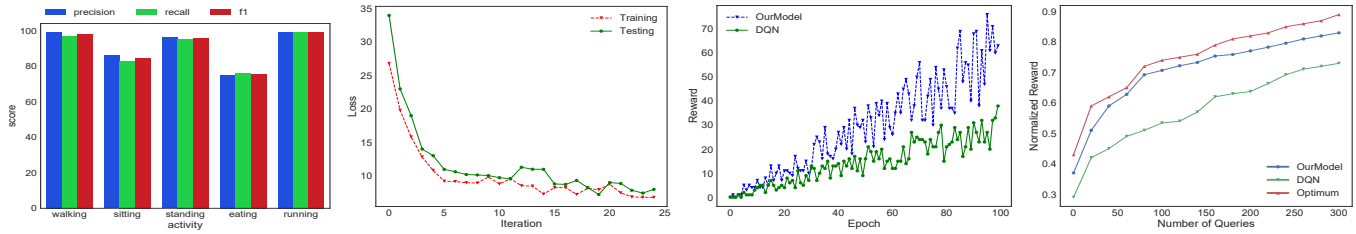
6.2 Data Collection & Preprocessing

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 app 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.

6.3 Network Architecture

6.3.1 Deep Active Learner. 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.



(a) (b) (c) (d)

Figure 4: (a) shows precision, recall and F1-score of our classifier for different activities. (b) the trend of loss function during training and testing. (c) illustrates average reward received with respect to number of epochs while training. (d) shows the progression of normalized reward with respect to number of posed queries.

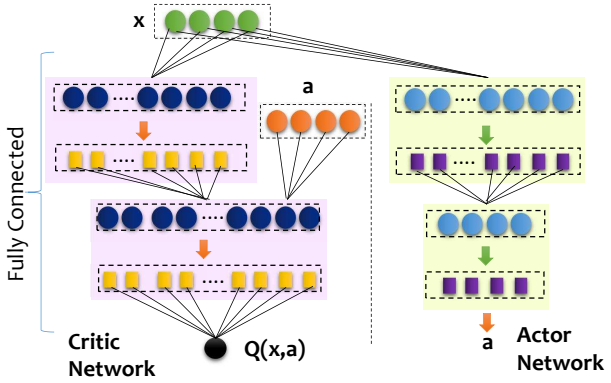


Figure 5: Architecture of actor-critic network.

6.3.2 Actor-Critic Netowrk. Two separate neural networks are trained to approximate the actor and critic functions. In figure 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.

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 $\alpha_\pi = 0.0001$.

6.4 Classifier Performance

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 1. 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 4a. In Figure 4b we demonstrate the trend of our loss function during training and testing.

Table 1: 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%

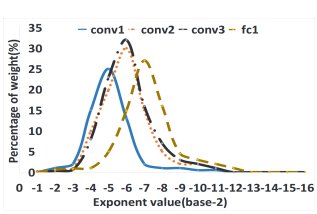


Figure 8: Weight distribution in log scale of different layers using cross-entropy loss.

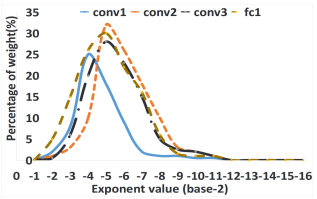


Figure 9: Weight distribution in log scale of different layers using our joint loss function.

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 8 and 9 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

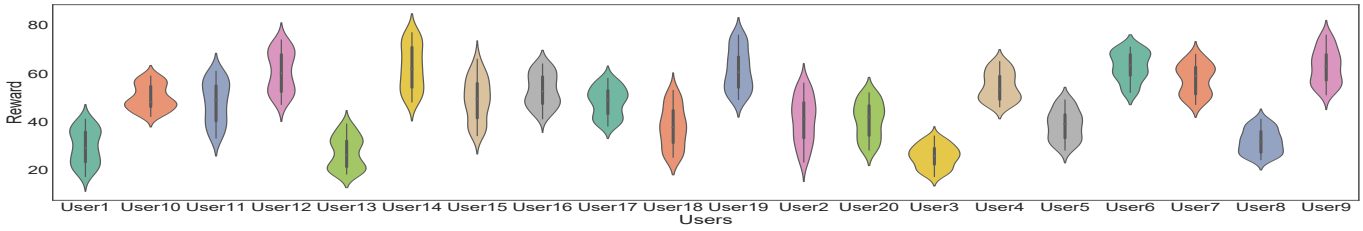


Figure 6: The plot illustrates the reward distribution of 20 users during our experiment.

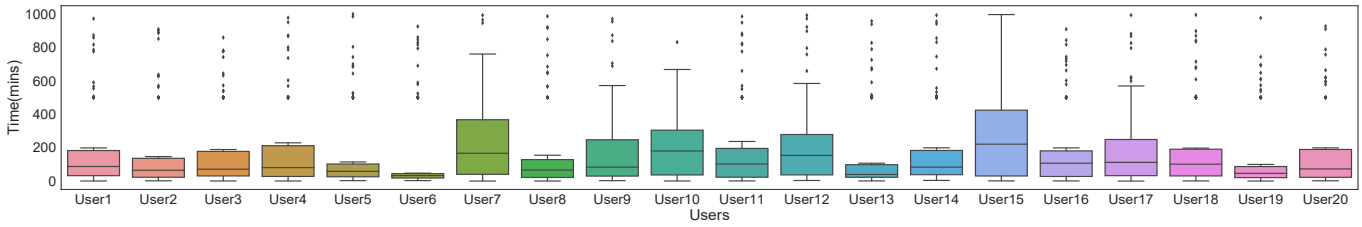


Figure 7: The figure illustrates the labeling time distribution of 20 users.

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.

6.5 Annotator Selection

To demonstrate the effectiveness of our annotator selection algorithm, we compared our algorithm with other popular deep reinforcement learning algorithm - Deep Q-Network (DQN) [18]. In Figure 4c 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 4d 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. 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 7 the distributions of annotation time for individual users are presented.

6.6 Impact of Active Learning

After training our 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 2. In Table 2 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 [35] 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 2: Comparison of our algorithm with other existing approaches with varying number of queries.

AR System	Number of Queries				
	0	100	200	300	400
Francisco [19]	81.2	84.14	87.87	88.57	89.47
Ming Zeng et al. [35]	80.39	82.47	86.33	87.59	89.25
Mohammad et al. [3]	81.01	83.17	84.11	87.89	88.32
Our model	83.26	85.65	88.39	90.48	91.64

7 CONCLUSION AND FUTURE WORK

In this paper we have proposed a deep model for activity recognition which integrates active learning in its hyperparameter tuning while prior works have only focused on calculating the most informative instance using active learning. We proposed to optimize the network parameters using a joint loss function consisting of cross-entropy loss of the deep model and the entropy function of the active learning pipeline. We have validated the performance of our trained classifier with data collected in real-world settings using our own mobile application. The results show that the joint loss function helps the deep model to have lower weight value and generalize well. Our model demonstrated better performance by selecting true informative instances in presence of outliers. We have also proposed a novel annotator selection

model using the contextual similarity between the annotators and the users. Our annotator selection model ensures faster convergence to the optimal award compared to other algorithm. Our experimental results show that our model converges faster to the optimal accuracy and after only 400 queries, the model exhibited ≈ 8% improvement in accuracy. In our current work, we have not considered any budget on the number of queries the system can pose to the users. In future we want to investigate more constraints like query budget and availability of the annotators in our annotator selection model.

ACKNOWLEDGMENTS

This research is partially supported by the NSF CAREER Award # 1750936, ONR under grant N00014-15-1-2229, and Alzheimer's Association, Grant/ Award # AARG-17-533039.

REFERENCES

[1] Mohammad Arif Ul Alam and Nirmalya Roy. 2017. Unseen Activity Recognitions: A Hierarchical Active Transfer Learning Approach. In *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*. 436–446.

[2] Hande Özgür Alemdar, T. L. M. van Kasteren, and Cem Ersoy. 2017. Active learning with uncertainty sampling for large scale activity recognition in smart homes. *JAISE* 9, 2 (2017), 209–223.

[3] Mohammad Abu Alsheikh, Ahmed Selim, Dusit Niyato, Linda Doyle, Shaowei Lin, and Hwee-Pink Tan. 2016. Deep Activity Recognition Models with Triaxial Accelerometers. In *Artificial Intelligence Applied to Assistive Technologies and Smart Environments, Papers from the 2016 AAAI Workshop*.

[4] Liming Chen, Jesse Hoey, Chris D. Nugent, Diane J. Cook, and Zhiwen Yu. 2012. Sensor-Based Activity Recognition. *Trans. Sys. Man Cyber Part C* 42, 6 (2012), 790–808.

[5] Gabriele Civitarese, Claudio Bettini, Timo Sztyler, Daniele Riboni, and Heiner Stuckenschmidt. 2018. 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*. 1–10.

[6] Pinar Donmez and Jaime G. Carbonell. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*. 619–628.

[7] Pinar Donmez, Jaime G. Carbonell, and Jeff G. Schneider. 2010. A Probabilistic Framework to Learn from Multiple Annotators with Time-Varying Accuracy. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*. 826–837.

[8] Yu Guan and Thomas Plötz. 2017. Ensembles of Deep LSTM Learners for Activity Recognition using Wearables. *IMWUT* 1, 2 (2017), 11:1–11:28.

[9] Mahmudul Hasan and Amit K. Roy-Chowdhury. 2015. Context Aware Active Learning of Activity Recognition Models. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 4543–4551.

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

[11] H. M. Sajjad Hossain, M. D. Abdullah Al Haiz Khan, and Nirmalya Roy. 2018. DeActive: Scaling Activity Recognition with Active Deep Learning. *IMWUT* 2, 2 (2018), 66:1–66:23.

[12] H. M. Sajjad Hossain and Nirmalya Roy. 2018. SocialAnnotator: Annotator Selection Using Activity and Social Context. In *Proceedings of the AAAI Fall Symposium on Reasoning and Learning in Real-World Systems for Long-Term Autonomy*. 30–37.

[13] Md Abdullah Al Hafiz Khan and Nirmalya Roy. 2017. 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*. 545–550.

[14] Md Abdullah Al Hafiz Khan, Nirmalya Roy, and Archan Misra. 2018. 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*. 1–9.

[15] Kundan Krishna, Deepali Jain, Sanket V. Mehta, and Sunav Choudhary. 2017. An LSTM Based System for Prediction of Human Activities with Durations. *IMWUT* 1, 4 (2017), 147:1–147:31.

[16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* (2015).

[17] Akhil Mathur, Tianlin Zhang, Sourav Bhattacharya, Petar Velickovic, Leonid Joffe, Nicholas D. Lane, Fahim Kawsar, and Pietro Liò. 2018. 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*. 200–211.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013).

[19] Francisco Javier Ordóñez Morales and Daniel Roggen. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* 16, 1 (2016), 115.

[20] Sebastian Münzner, Philip Schmidt, Attila Reiss, Michael Hanselmann, Rainer Stiefelhagen, and Robert Dürichen. 2017. CNN-based sensor fusion techniques for multimodal human activity recognition. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers, ISWC 2017, Maui, HI, USA, September 11-15, 2017*. 158–165.

[21] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. 2018. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Syst. Appl.* 105 (2018), 233–261.

[22] Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. 2018. AROMA: A Deep Multi-Task Learning Based Simple and Complex Human Activity Recognition Method Using Wearable Sensors. *IMWUT* 2, 2 (2018), 74:1–74:16.

[23] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D. Lane, Cecilia Mascolo, Mahesh K. Marina, and Fahim Kawsar. 2017. Multimodal Deep Learning for Activity and Context Recognition. *IMWUT* 1, 4 (2017), 157:1–157:27.

[24] Sreenivasan Ramasamy Ramamurthy and Nirmalya Roy. 2018. Recent trends in machine learning for human activity recognition - A survey. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery* 8, 4 (2018).

[25] Daniele Riboni, Timo Sztyler, Gabriele Civitarese, and Heiner Stuckenschmidt. 2016. 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*. 1–12.

[26] Seyed Ali Rokni, Marjan Nourollahi, and Hassan Ghasemzadeh. 2018. 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*.

[27] Sadiq Sani, Nirmalie Wiratunga, Stewart Massie, and Kay Cooper. 2018. Matching networks for personalised human activity recognition. In *Proceedings of the First Joint Workshop on AI in Health organized as part of the Federated AI Meeting (FAIM 2018)*. 61–64.

[28] Burr Settles. 2012. *Active Learning*. Morgan & Claypool Publishers.

[29] Farhad Shahmohammadi, Anahita Hosseini, Christine E. King, and Majid Sarrafzadeh. 2017. 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*. 321–329.

[30] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*.

[31] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press.

[32] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, and Philip S. Yu. 2018. 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*. 1–10.

[33] Chirine Wolley and Mohamed Quafafou. 2012. Learning from Multiple Annotators: When Data is Hard and Annotators are Unreliable. In *12th IEEE International Conference on Data Mining Workshops, ICDM Workshops, Brussels, Belgium, December 10, 2012*. 514–521.

[34] Ou Wu, Weiming Hu, and Jun Gao. 2011. Learning to Rank under Multiple Annotators. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. 1571–1576.

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