Proximity-Based Active Learning for Eating Moment Recognition in Wearable Systems

Marjan Nourollahi
Washington State University
Pullman, WA, USA
m.nourollahidarabad@wsu.edu

Seyed Ali Rokni
Washington State University
Pullman, WAcountryUSA
alirokni@eecs.wsu.edu

Parastoo Alinia
Washington State University
Pullman, WA, USA
parastoo.alinia@wsu.edu

Hassan Ghasemzadeh
Washington State University
Pullman, WA, USA
hassan@eecs.wsu.edu

ABSTRACT

Detecting when eating occurs is an essential step toward automatic dietary monitoring, medication adherence assessment, and diet-related health interventions. Wearable technologies play a central role in designing unobtrusive diet monitoring solutions by leveraging machine learning algorithms that work on time-series sensor data to detect eating moments. While much research has been done on developing activity recognition and eating moment detection algorithms, the performance of the detection algorithms drops substantially when the model is utilized by a new user. To facilitate the development of personalized models, we propose PALS[1], Proximity-based Active Learning on Streaming data, a novel proximity-based model for recognizing eating gestures to significantly decrease the need for labeled data with new users. Our extensive analysis in both controlled and uncontrolled settings indicates F-score of PALS ranges from 22% to 39% for a budget that varies from 10 to 60 queries. Furthermore, compared to the state-of-the-art approaches, off-line PALS achieves up to 40% higher recall and 12% higher F-score in detecting eating gestures.

CCS CONCEPTS

• **Computing methodologies → Active learning settings**; • **Human-centered computing → Ubiquitous and mobile computing design and evaluation methods**.

KEYWORDS

Machine learning, mobile health, eating detection, active learning, wearable computing

[1]Software code for PALS is available online at https://github.com/marjan-nourollahi/PALS

ACM Reference Format:
Marjan Nourollahi, Seyed Ali Rokni, Parastoo Alinia, and Hassan Ghasemzadeh. 2020. Proximity-Based Active Learning for Eating Moment Recognition in Wearable Systems. In *Toronto '20: ACM Workshop on Wearable Systems and Applications, June 19–21, 2020, Toronto, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Eating habits are highly correlated with human health [6]. Not only what people eat, but also when and how often eating events occur contributes to their health [9]. An automatic health monitoring system helps with monitoring eating habits and accommodating users with special health conditions such as diabetes [7] and those with certain dietary plans [3]. Therefore, eating moment detection is an important factor in automatic health monitoring.

Recent eating moment recognition approaches require multiple on-body sensors or specialized devices [1, 2], which make them impractical for everyday use. This research develops a machine learning model that uses easy-to-wear and prevalent wearable devices such as smartwatches for eating moment detection.

Cross-subject pattern variations while performing the same activity causes pre-trained machine learning models not to achieve desirable accuracy when used on the new subjects without collecting large labeled training data. This problem becomes even more challenging in real-life scenarios as the user's pattern in performing activities deviates significantly from the ones performed in the lab settings. A potential approach is to collect ground truth labels in real-life scenarios is to continuously record user's activities using body-worn cameras. However, deploying cameras in uncontrolled settings impose privacy concerns. Therefore, it is critical to develop strategies that allow for collecting ground truth labels outside laboratory settings.

Active learning is an approach to query sensor data for ground truth labels in end-user settings. It allows us to query a small subset of sensor data based on an informativeness measurement [10] and yet achieve an acceptable accuracy level. However, in mobile health using streaming data, the sensors are sampled in real-time and a decision about querying or skipping a data segment needs to be made instantaneously. To address the problem of activity learning with streaming sensor data, we propose PALS as a proximity-based active learning approach for eating moment recognition. We introduce PALS to improve the performance of the model with less labeled data while leveraging unlabeled data for model training. Inspired by graph-based semi-supervised learning research [12, 13], our approach utilizes unlabeled data to improve the quality of the model. To the best of our knowledge, PALS is the first attempt to develop a practical approach for eating moment detection using an active learning framework for human-in-the-loop learning on streaming sensor data.

2 PROBLEM STATEMENT

Let $\mathcal{X}$ denote a large set of collected sensor data. An observation $\mathbf{x_i}$ made by a wearable sensor at time i can be represented as a D-dimensional feature vector, $\mathbf{x_i} = \{w_{i1}, w_{i2}, \ldots, w_{iD}\}$. Each feature is computed from a given time window and a marginal probability distribution over all possible feature values. The activity recognition task is composed of a label space $\mathcal{A}$=$\{a_1, a_2, \ldots, a_m\}$ consisting of the set of labels for activities of interest, and a conditional probability distribution $P(\mathcal{A}|\mathbf{x_i})$ which is the probability of assigning a label $a_j \in \mathcal{A}$ given an observed instance $\mathbf{x_i}$. Subsequently, the final predicted label for observation $\mathbf{x_i}$ is defined as

$$f(\mathbf{x_i}) = \mathbf{arg}\max_{\mathbf{a_j} \in \mathcal{A}} \mathbf{P}(\mathbf{a_j}|\mathbf{x_i}) \tag{1}$$

Although, given the growing ubiquity of Internet-of-Things (IoT) sensors, collecting a large pool of unlabeled sensor data is attainable, labeling such a huge amount of data using human supervision is time-consuming, burdensome, and expensive. Therefore, it is important to devise an efficient approach for selecting informative instances taking into account the constraint of a limited budget to query an expert for ground truth labels. Furthermore, because the sensors are sampled continuously as the user performs various daily activities, the active learning algorithm needs to select sensor data for query in real-time. The reason for such a constraint is that expecting the user/expert to provide true labels for activities that occurred in the past is subject to human memory and bias errors. Therefore, it is desirable to decide if a query needs to be issued for the currently occurring activity. In this section, we formally define active learning as an optimization problem.

2.1 Limited Budget Training

To approach the problem of active learning given both budget and real-time decision making constraints, we first relax the second constraint by assuming that a human expert can label a pool of sensor data collected in the past by either remembering the activities or watching a video recording of the activities. This allows us to develop a basic pool-based active learning algorithm that selects most informative instances from a large pool of the collected sensor data. In the next step, we show how the pool-based algorithm can be modified for realizing real-time active learning scenarios where a decision about querying the expert is made instantaneously. In the following, we formulate each of the problems and present our solution to solve those problems. Problem 1 formally defines the limited budget active learning problem.

Problem 1 (Limited Budget Training (LBT)). *Assume an active learning algorithm splits the instances in $\mathcal{X}$ into two disjoint subsets l and U where the instances in l are used to query the oracle to obtain their true labels and those in U remain unlabeled. The Limited Budget Training (LBT) problem is to efficiently construct the small subset l and train a classifier such that the error of classifying instances in $\mathcal{X}$ is minimized and the size of l is bounded by a given query budget of Δ.*

The LBT problem described in Problem 1 can be formulated as follows.

$$Minimize \quad \sum_{i=1}^{|\mathcal{X}|} (|f(\mathbf{x_i}) - \mathbf{y_i}|) \tag{2}$$

$$|l| \leq \Delta \tag{3}$$

$$l \cup U = \mathcal{X} \tag{4}$$

$$l \cap U = \emptyset \tag{5}$$

The objective function in (2) aims to minimize the amount of misclassification error given the budget constraint in (3). The constraints in (4) and (5) are based on the definition where l and U are considered a perfect partitioning of set $\mathcal{X}$.

As described in Problem 1, due to limited budget constraints, designing an efficient method to cherry-pick instances to feed the training process is essential. Here, Definition 1 formally defines the instance selector function.

Definition 1 (Instance Selector). *An instance selector $\mathcal{I}$ is a function $\mathcal{I} : \mathcal{X} \rightarrow \{0, 1\}$ such that*

$$\mathcal{I} = \begin{cases} 1, & if\, \mathbf{x_i} \in \mathbf{l} \\ 0, & otherwise \end{cases} \tag{6}$$

where x_i refers to the instances selected for the query. Considering that the active learning algorithm uses the instance selector $\mathcal{I}$, the Problem 1 could be re-formulate as an Integer Linear Programming problem as follows.

$$Minimize \quad \sum_{i=1}^{|\mathcal{X}|} (1 - \mathcal{I}(i))\,|f(\mathbf{x_i}) - \mathbf{y_i}| \tag{7}$$

$$\sum_{i=1}^{|\mathcal{X}|} \mathcal{I}(i) = \Delta \tag{8}$$

The objective function in (7) aims to minimize the amount of misclassification error on unknown instances while (8) states the budget constraint.

A major limitation of the LBT problem described above is that it assumes perfect memory retention for the oracle. That is, the oracle can remember past events reliably. In reality, however, mobile health technologies monitor end users continuously and the user may not remember past events. Therefore, it is more realistic to design an active learning approach for streaming sensor data. In the following, we reformulate Problem 1 taking into account that the oracle provides labels for current activity. Problem 2 formally defines the problem of training with a limited budget on a stream of data.

Problem 2 (Limited Budget Training on Data Stream (LBTS)). *Let $\mathcal{X}$=$[x_1, x_2, \ldots, x_t, \ldots, x_T]$ be a sequence of sensor instances that are being produced during time frame t= $\{1, \ldots, T\}$. An active learning algorithm on stream splits the instances in $\mathcal{X}$ into two disjoint subsequences l and U where the instances in l are used to query the oracle to obtain their true label and update the model as they become available in real-time while U remain unlabeled. The Limited Budget Training on Stream(LBTS) is to efficiently decide whether to query the true label for the instance at time t and update the classifier as* *it becomes available in real-time such that the error of classifying instances in U is minimized.*

Using Linear Programming framework in (7), the problem of limited budget training on data stream could be formulated as follows.

$$Minimize \quad \sum_{t=1}^{T}(1 - \mathcal{I}(t))\,|f_t(\mathbf{x_t}) - \mathbf{y_t}| \tag{9}$$

$$\sum_{t=1}^{T}\mathcal{I}(t) = \Delta \tag{10}$$

where f_t is the classification function at time t. The objective function in (9) aims to minimize the amount of misclassification error given the budget constraint in (10).

3 PALS FRAMEWORK DESIGN

PALS framework focuses on two characteristics of everyday living situations: (1) the ubiquity of data and the ability to obtain huge amounts of unlabeled data with mobile devices and wearable sensors; and (2) realistic assumption that the user/expert has a limited capability or interest in providing ground truth labels for the massive amounts of data that are being collected in continuous health monitoring applications. Therefore, the general goal of the PALS framework is to leverage the unlabeled data to construct an efficient model while choosing a small subset of instances of the unlabeled data to query the user/expert for label/annotation. In the following, we described our approach for leveraging unlabeled data through a proximity graph model and selecting informative data instances in preparation to query the expert.

3.1 Proximity-Based Modeling

Inspired by graph-based semi-supervised learning research, we propose to construct a proximity-based model to quantify similarity among data instances. The intuition behind a proximity-based modeling and label inference is *smoothness assumption.* The *smoothness assumption* suggests that the instances that are close in the feature space should have similar labels [12]. The process of constructing a proximity-based model includes two phases. The first phase aims to build a proximity graph using both labeled and unlabeled data. Leveraging unlabeled data could potentially improve the model. As suggested by prior research [12], in the absence of sufficient labeled data, using both labeled and unlabeled data can lead to a more accurate decision boundary for the learned model. The second phase is label inference, which focuses on generating labels for unlabeled instances through an iterative label propagation method.

Definition 2 (Proximity Graph). *A proximity graph $G(V, E)$ is a weighted graph where each node in V represents an instance in $X = l \cup U$. Each node in the graph maintains a vector of its feature values and the probability distribution of its labels. An edge $e_{ij} \in E$ represents the amount of similarity between instances $\mathbf{x_i}$ and $\mathbf{x_j}$.*

We denote the similarity between $\mathbf{x_i}$ and $\mathbf{x_j}$ by $\eta(\mathbf{x_i}, \mathbf{x_j})$ and compute its value by their Euclidean distance:

$$\eta(i, j) = \|\mathbf{x_i} - \mathbf{x_j}\| \tag{11}$$

To avoid the confusion of far away instances, we build similarity graph using k-NN schema which is one of the most popular approaches in similarity graph construction [8]. Therefore, we measure edge weights in the similarity graph using the following equation:

$$e_{ij} = \begin{cases} \eta(i, j) & \text{if } i \in \kappa(j) \text{ or } j \in \kappa(i) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $\kappa(i)$ is the set of k-nearest-neighbors of instance $\mathbf{x_i}$ based on the defined similarity function.

In practice, we will show that using the k-NN schema improves the performance of the trained model in detecting eating moments.

3.2 Instance Selector

To maximize the labeling accuracy while taking into account the constraint in (3), we need an effective instance selector function to select the most informative instances from $\mathcal{X}$ to add to the training data used to learn a final model. To quantify the informativeness of the instances, in this article, we use an entropy-based method, which generates a score for a given instance based on Information Gain (IG) from that instance. Recall that entropy indicates the certainty of the model in classifying an instance. An entropy of zero means pure certainty with one of the classes receiving a probability of one. Therefore, low values of entropy suggest that the model is confident about how to classify the input instance. The instance selector $\mathcal{I}$ sorts the instances by their information gain and selects the instance with the highest information gain to add to the labeled pool l.

3.3 Off-line PALS

As described previously in Section 2, in the off-line version of PALS, we assume that a pool of unlabeled sensor instances is available to the oracle. The oracle is then able to label any of the instances and to assign the correct activity label upon request. In this off-line approach, we assume that the provided label is correct. This assumption is based on the fact that either the oracle's memory is perfect that they can remember past events or there is a video recording of the activities that the oracle can navigate to find the correct label for a queried activity.

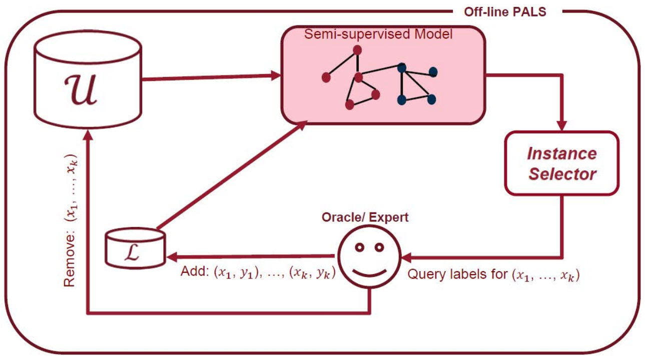
Off-line PALS
Semi-supervised Model
$\mathcal{U}$
Instance Selector
Oracle/ Expert
$\mathcal{L}$
Add: $(x_1, y_1), \ldots, (x_k, y_k)$
Query labels for $(x_1, \ldots, x_k)$
Remove: $(x_1, \ldots, x_k)$

Figure 1: Overall architecture of PALS for off-line active learning.

Fig 1 shows the overall architecture of our off-line proximity-based active learning approach. Initially, among all of the recorded activities, there is no or a small set of labeled instances l along with a large pool of unlabeled instances U. Our algorithm constructs a proximity-based graph on the entire dataset using both l and U.

Following the graph construction phase, the model aims to infer the actual label of the instances in U in multiple iterations of the label propagation procedure. In the next step, the instance selector $\mathcal{I}$ searches through the unlabeled instances to find the most informative instance in U, to date, to request for a label. The process concludes by adding the labeled instance to the model.

Algorithm 1 Algorithm for Off-line PALS

```
    Input: labeled data $l$, unlabeled pool $U$, number of iterations $k$, budget $\Delta$
    Output: Proximity-based model $f$
    Initialize: $\delta \leftarrow \frac{\Delta}{k}$ , $itr \leftarrow k$
 1: procedure Offline PALS
 2:     $f \leftarrow$ construct proximity-based model on $l \cup U$
 3:     while $itr > 0$ do
 4:         $L_u \leftarrow$ inferred labels on $U$ using model $f$
 5:         $IGs \leftarrow IG(L_u)$
 6:         $X_{sel} \leftarrow \delta$ instances with highest $IGs$
 7:         $l_{sel} \leftarrow$ labels provided by oracle for $X_{sel}$
 8:         $l \leftarrow l \cup (X_{sel}, l_{sel})$
 9:         $f \leftarrow$ update model $f$ with new instances in $l$
10:         $itr \leftarrow itr - 1$
```

As illustrated in Fig 1, the process continues iteratively by obtaining new labeled instances and adding them to the labeled set l. The model is then updated and the process of label inference and instance selection is repeated. The algorithm finishes when all the allowed queries are exhausted (i.e. $|l| = \Delta$). Algorithm 1 shows the off-line active learning approach in PALS.

3.4 Real-time PALS

To realize real-time active learning on streaming data, we develop real-time PALS. Development of real-time PALS is motivated by the fact that both non-stop video recording of user's activities in naturalistic settings and assuming perfect memory for the user to accurately remember all activities performed in a given time-frame in the past are unrealistic for activity recognition in free-living situations. Therefore, to develop a personalized model in real-life scenarios, we cannot solely rely on pool-based active learning. Yet, we develop our real-time PALS algorithms based on the foundations established in our off-line PALS.

The main challenge in real-time active learning is to decide whether or not to query each sensor instance as it becomes available in real-time. In particular, because the model does not have access to future instances, it needs to determine whether the current instance is informative enough for which to request a label. We define a threshold on the informativeness of a given instance to make such determination in real-time. Such a threshold, if defined appropriately, will allow us to make real-time active learning decisions.

Definition 3 (Informativeness Threshold). *Let X^{sort} be the entire stream X sorted in informativeness score given by IG. An informativeness threshold λ is a value such that $IG(X^{sorted}_{\Delta}) = \lambda$ where Δ is the query budget.*

As shown in Fig 2, real-time PALS assumes that the user can provide labels only for the current or very recent activities. In this approach, each instance is evaluated only once. As a result of this

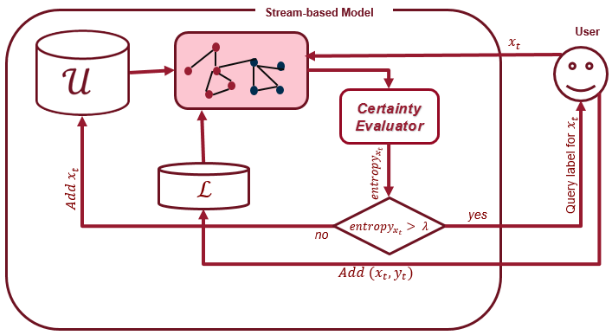

Figure 2: Overall architecture of PALS for real-time active learning on streaming sensor data.

evaluation, the instance is either discarded from further analysis or used to query the oracle. If the system receives a label from the oracle, the next step is to update the model with the new instance to obtain a more personalized model. This is accomplished by adding the newly labeled instance to the labeled pool.

Algorithm 2 Algorithm for real-time PALS.

```
    Input: current model $f_c$, new instance $x$, threshold $\lambda$, budget $\Delta$
    Output: $f$
 1: procedure Real-time PALS
 2:     $p \leftarrow$ make a prediction on $x$ using model $f_c$
 3:     $e \leftarrow$ calculate entropy of $p$
 4:     if $e \geq \lambda$ and $\Delta > 0$ then
 5:         $\Delta \leftarrow \Delta - 1$
 6:         $y \leftarrow$ query oracle to provide a true label for $x$
 7:         $f \leftarrow$ update model $f_c$ with $(x, y)$
```

We need an algorithm to adjust the value of the informativeness threshold to balance labeling over the instance space. The algorithm needs to avoid both high and low values of λ. High values of λ will translate into a highly conservative approach where a very small number of questions are asked. Therefore, the algorithm can fail in personalizing the model for the current user due to a lack of sufficient input from the user. On the other hand, low values of λ will result in the algorithm exhausting the budget very quickly rather than generating queries that are distributed in time. Therefore, we need an adaptive algorithm to adjust λ to create a balance between prompting time and query budget.

3.5 Adaptive Threshold Setting

An adaptive algorithm for adjusting λ needs to address concerns of when and how to update λ to achieve effective performance. Our strategy is to update λ after receiving a new instance to a value that ensures a uniform distribution of queries over a given time interval. Suppose N denotes the number of instances over a given time interval. Also, assume that we have seen k instances so far. To uniformly distribute queries over the time interval, we need to adjust λ taking into account the fact that $\frac{k}{N}$ percentage of the budget has been already exhausted. Here we describe how λ can be adjusted for a stream of data to ensure a uniform distribution of queries over a given time interval of T.

Let T denote a given expected execution time interval of the active learning process. Also, let $\mathcal{X}_t$ represent the data stream generated up to time t and $\mathcal{X}_t^{sort}$ be $\mathcal{X}_t$ sorted in non-decreasing order by informativeness score given by IG. Furthermore, let Δ_t be $(t/T) \times \Delta$. An informativeness threshold at time t is denoted by λ_t is a value such that $IG((\mathcal{X}_t)^{sorted}_{\Delta_t}) = \lambda_t$. The threshold value λ_t ensures a uniform distribution of queries over the time interval T. This process for obtaining an adaptive λ is shown in Algorithm 3.

Algorithm 3 Algorithm for adaptive adjustment of informativeness threshold, λ

```
Input: current model $m$, new instance $x$, time $t$, time interval $T$, Entropy of instances up to current time $E$, budget $\Delta$
Output: $\lambda_t$
1: procedure ADAPTIVE$\lambda$
2:     $p \leftarrow$ use model $m$ to make predictions about $x$
3:     $e \leftarrow$ calculate the entropy of $p$
4:     $E \leftarrow (E + e)^{sorted}$
5:     $index \leftarrow (t/T) \times \Delta$
6:     $\lambda_t \leftarrow E_{index}$
```

4 RESULTS

This section presents experimental results for PALS on two public datasets SW6S and SW3S [11]. Both datasets contain 3D accelerometer data collected from a wristband worn on the dominant hand while doing eating-related activities. SW3S was collected in a semi-lab setting from 20 participants performing different activities including eating, watching a movie trailer, chatting, taking a walk, placing a phone call, brushing teeth, and combing hair. The second dataset was collected in free-living settings with seven participants. The participants performed various daily activities such as taking, commuting, reading, walking, working with a computer, and eating. We extract 15 features such as *median* and *mean* capture intensity of the signal, *variance* and *zero crossing* intend to capture morphology of the signal from signal segments for each axis of sensor data. We use the χ^2 feature selection method to eliminate irrelevant features. Since the majority of the samples in the datasets are non-eating activities we use Synthetic Minority Over-sampling Technique (SMOTE) [4] to up-sample the selected most informative instances in each iteration of offline PALS. To avoid biased results while testing the model on the unbalanced data, we report the *f-score* value, which is a metric to measure the quality of the model based on the balance between *Precision* and *Recall*.

4.1 Performance of Offline PALS

We compare our algorithm with two prior research in the area of eating moment detection as well as state-of-the-art machine learning methods. (1) RFA, a Random Forest-based food intake gesture recognition algorithm [11], and, (2) XGBoost, which is an optimized and distributed implementation of Gradient Boosting. It provides a parallel tree boosting method to effectively solve machine learning problems in the industrial scale [5]. We assume that each algorithm has access to 20% of in-lab data as its training set and we use the remaining 80% of the data as a test set for validation. As shown in Table 1, offline PALS outperforms other approaches; it achieves 41% and 48% f-scores detecting eating moments for SW3S and SW6S datasets, respectively, which is more improvement than XGBoost and RFA. Also, low recall for eating class refers to the classifier having a high bias in classifying all instances as not-eating. This again emphasizes the importance of selecting appropriate metrics while working with skewed datasets. As presented in Table 1, offline PALS achieves a 62% and 64% recall when running on SW3S and SW6S datasets, respectively, which demonstrates significant improvements over RFA and XGBoost.

Table 1: Performance of offline PALS vs. other approaches.

Dataset	Methods	Recall	F-score
SW3S	Offline PALS	0.62	0.41
	XGBOOST	0.25	0.35
	RFA	0.22	0.34
SW6S	Offline PALS	0.64	0.48
	XGBOOST	0.32	0.40
	RFA	0.10	0.18

4.2 Performance of Real-Time PALS

We conducted two experiments highlighting the effect of query budget and decision threshold estimation on the performance of the real-time PALS algorithm.

4.2.1 Query Budget. We assess the effect of query budget, Δ, on the performance of real-time PALS approach in classifying eating moments by examining twelve different values of query budget per hour for different subjects in in-the-wild setting on SW3U dataset.

Fig 3 shows the f-score value averaged over all the subjects at the end of training cycles. Limited query budget to query the user in real-time, the model cannot adapt itself from the lab-setting to real-world setting (less than 1% f-score for 5 queries per hour). First, the distribution of eating vs. non-eating activities is very different from a lab setting to a real-world setting. Second, in the real-world setting, real-world settings and without people tend to perform eating activities with higher variations than the lab setting. This result again highlights the importance of designing adaptive models for real-world settings. However, increasing the value of the budget creates a more personalized classifier for each participant with a higher performance measure. Increasing the query budget to 10 queries, the average f-score of detecting eating-moments increases by around 23.1%, 29.8%, and 39% having the query budget of 10, 20, and 60, respectively. There is always a trade-off between the query budget and user convenience. While by increasing the query budget, we increase the performance of the model, we may also increase user inconvenience.

4.2.2 Comparison of Thresholding Methods. To verify the effectiveness of our approach in updating the decision threshold, λ, we designed two different methods. In the first method, the value of the λ is learned from the in-lab training data which is derived based on the ratio of budget to the size of the dataset. Since this value extracted from the in-lab data and remains unchanged during the real-time training, we refer to the decision threshold obtained in this approach as *static* λ. The second method uses the knowledge of the best possible value for the threshold in a time interval to select the most informative instances based on the entropy of the classification decision. This experiment provides an experimental

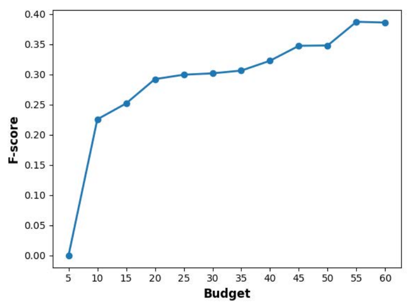

Figure 3: Performance of the learned model in terms of f-score as a function of query budget on SW3U dataset.

upper-bound for the adaptive lambda because it has unlimited access to the future data and can extract the most accurate value of λ that the adaptive lambda algorithm attempts to estimate. We refer to the decision threshold obtained by this approach as *best* λ.

We compared the performance of the eating moment detection models trained on real-time data of the SW3U dataset using *best* λ, *adaptive* λ, and *static* λ. The x-axis refers to different subjects and the y-axis shows the binary f-score. The query budget is set to 60 queries per hour. As Fig 4 shows, the adaptive lambda algorithm achieves performance values close to *best* λ while using a static value for λ performs poorly across different subjects. Specifically, *adaptive* λ on average achieves 7% less f-score compared to *best* λ and 12% better f-score compared to *static* λ. Also, to evaluate the extreme cases, *adaptive* λ achieves to 13% less f-score compared to *best* λ for subject number 5 while it works better for other subjects. Furthermore, *adaptive* λ works, in worst case, slightly better than *static* λ with 1.6% better f-score for subject number 6 while it outperforms *static* λ for other subjects specifically subject 7 with 28% higher f-score. To summarize the results of this experiment, *best* λ, *adaptive* λ, and *static* λ on average can provide 47%, 39%, and 28% average f-score for all subjects of SW3U dataset.

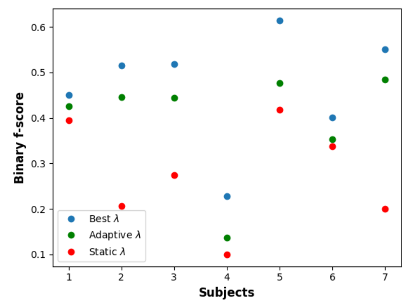

Figure 4: Comparison of *best* λ, *adaptive* λ, and *static* λ for decision threshold in terms of f-score on SW3U dataset.

5 CONCLUSIONS

Most approaches to detect eating moment require multiple on-body sensors or specialized devices such as neck-collars for swallow detection that are impractical for everyday usage. This research developed a practical solution for eating moment detection using wearable sensors. We designed a non-intrusive detection system with machine learning algorithms personalized for the end-user.

Because people perform the same activity differently, relying on a model trained based on in-lab data leads to a significant performance drop for the new users. We proposed a novel proximity-based model to recognize eating gestures. We showed that our approach significantly decreases the need for labeled data with new users leveraging active learning under a limited query budget. Our extensive analysis of data collected from real subjects showed that PALS achieves up to 40% higher recall and 12% higher F-score in detecting eating events. Furthermore, we showed the effectiveness of our adaptive thresholding method and how the online PALS algorithm could be adapted in real-world settings with a limited query budget.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation, under grants CNS-1750679 and CNS-1932346. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

REFERENCES

[1] Abdelkareem Bedri, Richard Li, Malcolm Haynes, Raj Prateek Kosaraju, Ishaan Grover, Temiloluwa Prioleau, Min Yan Beh, Mayank Goel, Thad Starner, and Gregory Abowd. 2017. EarBit: using wearable sensors to detect eating episodes in unconstrained environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 37.
[2] Abdelkareem Bedri, Apoorva Verlekar, Edison Thomaz, Valerie Avva, and Thad Starner. 2015. Detecting mastication: A wearable approach. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, 247–250.
[3] Samir Chatterjee and Alan Price. 2009. Healthy living with persuasive technologies: framework, issues, and challenges. *Journal of the American Medical Informatics Association* 16, 2 (2009), 171–178.
[4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.
[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 785–794.
[6] Katherine M Flegal, Margaret D Carroll, Cynthia L Ogden, and Clifford L Johnson. 2002. Prevalence and trends in obesity among US adults, 1999-2000. *Jama* 288, 14 (2002), 1723–1727.
[7] Abdelsalam Helal, Diane J Cook, and Mark Schmalz. 2009. Smart home-based health platform for behavioral monitoring and alteration of diabetes patients. *Journal of diabetes science and technology* 3, 1 (2009), 141–148.
[8] Markus Maier, Matthias Hein, and Ulrike Von Luxburg. 2007. Cluster identification in nearest-neighbor graphs. In *Algorithmic Learning Theory*. Springer, 196–210.
[9] Theresa A Nicklas, Tom Baranowski, Karen W Cullen, and Gerald Berenson. 2001. Eating patterns, dietary quality and obesity. *Journal of the American College of Nutrition* 20, 6 (2001), 599–608.
[10] Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.
[11] Edison Thomaz, Irfan Essa, and Gregory D Abowd. 2015. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1029–1040.
[12] Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison* 2, 3 (2006), 4.
[13] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.