Automatic Class Discovery and One-Shot Interactions for Acoustic Activity Recognition

Jason Wu¹,² Chris Harrison¹ Jeffrey P. Bigham²,¹ Gierad Laput²,¹

¹Carnegie Mellon University, HCl Institute
5000 Forbes Avenue, Pittsburgh, PA, USA
{janwu2, chris.harrison}@cs.cmu.edu

²Apple Inc.
One Apple Park Way, Cupertino, CA, USA
{gierad, jbigham}@apple.com

ABSTRACT
Acoustic activity recognition has emerged as a foundational element for imbuing devices with context-driven capabilities, enabling richer, more assistive, and more accommodating computational experiences. Traditional approaches rely either on custom models trained in situ, or general models pre-trained on preexisting data, with each approach having accuracy and user burden implications. We present Listen Learner, a technique for activity recognition that gradually learns events specific to a deployed environment while minimizing user burden. Specifically, we built an end-to-end system for self-supervised learning of events labelled through one-shot interaction. We describe and quantify system performance 1) on preexisting audio datasets, 2) on real-world datasets we collected, and 3) through user studies which uncovered system behaviors suitable for this new type of interaction. Our results show that our system can accurately and automatically learn acoustic events across environments (e.g., 97% precision, 87% recall), while adhering to users’ preferences for non-intrusive interactive behavior.

Author Keywords
Automatic class discovery; Acoustic activity recognition

CCS Concepts
Human-centered computing—Ubiquitous and mobile computing systems and tools.

INTRODUCTION
Smart devices are becoming more prevalent in peoples’ living environments, accelerating the vision of ubiquitous computing and the Internet-of-Things (IoT). However, these devices still lack contextual sensing capabilities— they have minimal understanding of what is happening around them, therefore limiting their potential to enable truly assistive computational experiences. In response, acoustic activity recognition has emerged as a practical modality for contextual sensing. This is chiefly due to the prevalence of microphones and their robustness to occlusion, as well as the availability of hardware that can process high-fidelity acoustic information on-device (i.e., at the edge).

A key challenge for acoustic activity recognition is building classifiers that can recognize highly localized events with minimal user intervention or in situ training. To train such classifiers, two predominant approaches have been proposed, with particular accuracy and user burden implications (Figure 1). First is to train a system manually, after it is deployed, most often by demonstrating different activities and having a user provide class labels (Figure 1, top-right). Because data is collected in-situ, accuracy tends to be quite high. However, the burden to the user is also high. The other approach is to provide users with pre-trained general classifiers that work “out of the box” (Figure 1, bottom-left). This technique is achieved by training a classifier on a large, pre-existing corpus of acoustic data. However, because the classifier has no data for a user’s specific environment, it tends to be less accurate, but the burden to the user is very low.

We propose and evaluate a middle-ground approach that seeks to provide high classification accuracy, while minimizing user burden. Our approach (Figure 2) requires no up-front data, and instead, learns acoustic events over time, requiring no manual demonstration. Instead, our system learns events in situ, and thus it is highly tuned to its local environment and events of interest, offering superior accuracy than pre-trained classifiers.
LISTEN LEARNER

In this paper, we characterize an operational space for personalized Human Activity Recognition (HAR) systems using two factors that significantly impact practicality—classification accuracy and user burden. We contribute an approach that optimizes this tradeoff with an interactive, low-burden approach, along with an end-to-end hardware and software implementation supporting several virtual assistant-driven interaction techniques. Finally, we evaluate both quantitative (i.e., benchmarks and comparisons of our system’s performance in a variety of contexts) and qualitative (i.e., users’ preferences for interactive ML systems) aspects of our system.

Example Interaction

To illustrate the utility of Listen Learner, we describe the following vignette:

Setup. Lisa deploys a smart speaker, equipped with Listen Learner, on her kitchen countertop. The system starts with no data or knowledge about its environment. As sounds in Lisa's kitchen occur, the device clusters similar acoustic events. No raw audio is saved to the device or to the cloud.

One-Shot Labeling. Eventually, the system becomes confident that an emerging cluster of data is a unique sound, at which point, it prompts Lisa for a label the next time it occurs. The system asks: “what sound was that?” and Lisa responds with: “that is my faucet.” As time goes on, the system can continue to intelligently prompt Lisa for labels, thus slowly building up a library of recognized events.

Verification and Refinement. Instead of asking an open-ended question, the system can make an initial guess (using a general pretrained model). The system might ask: “was that a blender?” in which Lisa responds: “no, that was my coffee machine.” In other cases (e.g., ambiguous cluster boundaries), the system can ask refinement questions such as: “was that a faucet or a microwave?” in which Lisa responds: “it’s a microwave.” The library of sounds that the system builds over time can then be used to power new assistive and smart applications.

Unlike traditional supervised learning methods that require numerous user-labelled examples in the training phase, our approach inverts the annotation and training process. Our system learns an ensemble of classifiers without any user intervention, and only later is the user queried in situ to provide a label for the model the next time it is triggered.

RELATED WORK

We situate our system in the literature of contextual sensing for activity recognition and machine learning methods for real-world activity recognition.

Audio Event Classification

Optical sensors, such as RGB cameras [23][32][35][36] or depth sensors [38][82] are popular approaches for human activity recognition [14], but these systems are susceptible to occlusion. In response, audio-based sensing has emerged as a complementary modality, deployed in both localized [44][45][68][80] and wide-area applications [34][62][69]. Approaches for distilling acoustic information include computing statistical features on time-domain [70], frequency [46][70] or wavelet representations [70][72]. More recently, deep-learning architectures have been used to model the inherent non-linearities in acoustic data. Here, audio signals are treated as one-dimensional signals [8][51], or two-dimensional spectrograms [24] that serve as input to convolutional neural networks (CNNs), previously used for image classification [24][34]. Listen Learner uses the “bottleneck” embedding representation of a CNN (similar to those previously used for audio event discovery and activity recognition [28][41]), but fine-tuned on a library of sound effects [34].

Generalizable Machine Learning Methods for HAR

A major challenge of HAR is training highly robust machine learning models (i.e., accurate classification, sparse false positives). One approach is to employ semi-supervised learning techniques such as Positive Unlabeled Learning (PUL) [20] to learn from a small number of positively labelled samples. In the context of HAR, Nguyen et al. propose using a specific form of PUL, called mPUL to decrease the amount of training data and reduce false-positives by assuming “open-world conditions”[49]. Others have focused on active learning to intelligently scaffold the training process [26][43].
A related strategy is to model user activity using a set of semantic attributes—allowing activities to be defined in a more generalizable way [3][50][75].

**Relevant Machine Learning Approaches**

One of the goals of our system is to require limited training labels from users. Relevant approaches include co-training [13], a semi-supervised learning method [16] for leveraging a small number of examples and a large unlabeled set to create a model with better classification performance. Likewise, one-shot and zero-shot learning approaches allow models to recognize previously unseen classes with very few (or zero) labelled training instances [79]. Our system most closely resembles *incremental learning*, a learning approach that accommodates new data to continuously improve and extend a model’s knowledge without fully retraining the model [56]. This approach to learning has been explored in many domains, including computer vision [40][57], audio event recognition [19], natural language processing [15][81] and activity recognition [47][61].

SoundSense [44] is most similar to Listen Learner, in that it also provides a platform for audio event discovery. SoundSense was implemented on a mobile platform and starts with a pre-existing set of classes that it uses to bootstrap recognizers for new classes. SoundSense uses a Bayesian classifier and Hidden Markov Model (HMM) to learn new audio events, which results in assumptions about the distribution of audio events (*e.g.*, can be modeled by a Gaussian). The model also requires a number of parameters to be set that can be difficult without *a priori* knowledge. In our evaluation, we show that Listen Learner has superior performance to a baseline Gaussian Mixture Model (GMM), another type of model that assumes a Gaussian distribution of data.

**IMPLEMENTATION**

We implemented Listen Learner as an end-to-end system that automatically generates acoustic event classifiers over time. Here, we describe our sensing hardware, data processing pipeline, and self-supervised learning algorithm.

**Hardware**

Our prototype consists of both a deployed sensor device (analogous to a smart speaker; Figure 3), and a processing server (on which the self-supervised learning algorithm is executed). Specifically, we use a Raspberry Pi 3 Model B+ with a 4-microphone array (seeedstudio.io), which we use to compute acoustic direction-of-arrival (part of our feature set). We also connect a speaker using its 3.5 mm audio jack. We set the microphone sampling rate to 16 kHz 16-bit integer linear PCM. The device is configured to connect to WiFi and upload featurized audio data to our data processing server (12-core Mac Pro, 64GB RAM).

**Cluster-Classify Algorithm**

We designed a self-supervised algorithm that identifies salient acoustic events, generating corresponding classifiers for activity recognition, while minimizing user effort (Figure 4). More specifically, the algorithm learns an ensemble model by iteratively clustering unknown samples, and then training classifiers on the resulting cluster assignments. This allows for a “one-shot” interaction with the user to label portions of the ensemble model when they are activated.

**Segmentation**

First, we segment audio events using an adaptive threshold that triggers when the microphone input level (d BFS) is 1.5 standard deviations higher than the mean of the past minute. We employ hysteresis techniques (*i.e.*, for debouncing) to further smooth our thresholding scheme. While many environments have persistent and characteristic background sounds (*e.g.*, HVAC), we ignore them (along with silence) for computational efficiency. Note that incoming samples were discarded if they were too similar to ambient noise, but silence within a segmented window is not removed.

**Featureization**

Next, we convert audio segments into feature embeddings extracted from the last hidden layer of a VGG-ish [67] deep CNN audio model [24]. This model was initially trained on the YouTube-8M dataset, and further augmented with a library of professional sound effects [34]. We construct 96×64 log-mel spectrogram patches as input to the CNN using a non-overlapping 960 ms sliding window over audio input. For example, an audio clip of a faucet running for 9.6 seconds would produce 10 featurized embeddings. In our prototype hardware, this computation takes an extra one second per 960 ms of audio. While this causes some input frames to be dropped, we do not find this limiting (*i.e.*, due to the sustained nature of most human activities). The choice of using deep neural network embeddings, which can be seen as learned low-dimensional representations of input data [12], is consistent with the manifold assumption (*i.e.*, that high-dimensional data roughly lie on a low-dimensional manifold [16]). By performing clustering and classification on this low-dimensional learned representation, our system is able to more easily discover and recognize novel sound classes.

**Clustering**

Next, we infer the location of class boundaries from our low-dimensional learned representations using unsupervised clustering methods. Our approach is supported by the cluster assumption, which states that if points are in the same cluster, they are likely to belong to the same class and that the decision boundary between classes should lie in a low-density region [16]. For our implementation, we use a hierarchical agglomerative clustering (HAC) algorithm known as Ward’s.
As a practical compromise, of course, there are computational and data storage limits. Earlier, the data pool grows as more audio is captured from more data is revealed to our system over time. As described earlier, the data pool grows as more audio is captured from the environment. When a new batch of data is added, we re-run our algorithm.

Of course, there are computational and data storage limits. As a practical compromise, in our current implementation, we only store audio samples within a fixed time window (e.g., one-week’s worth of data). When new data is received beyond this threshold, the oldest samples are discarded. Other methods are also possible, depending on the hardware or desired behavior (e.g., random subsampling of the data pool, or a replacement scheme that discards data points based on classification accuracy rather than age).

Data Management
For research purposes, we chose to aggregate data collected from the sensing devices on a central server, which also permitted us to process data more efficiently. In our study deployment, we only transmitted and stored featurized data, which is computed on the sensing board. The algorithm has access to a data pool, which contains featurized data that is not yet recognized by our system. Data is added to the pool in batches (e.g., after the end of each day, or after n samples).

Audio Directionality
Our sensing hardware includes a microphone array, allowing additional directionality information, which serves as a complementary sensing modality. We chose to represent this using the x and y components of a normalized unit vector. During the clustering step, we use a late-fusion approach [9] that performs clustering on two sources of data: 1) clusters based on directional information, and 2) clusters from audio embeddings. Clusters from both sources are provided to the classify step, with directional information considered first. During the classification stage, we allow the classifier to learn the relative weights of each modality by using an early-fusion approach (i.e., concatenating audio embeddings with directionality vectors).

User Interaction
Once our system has generated classifiers for the model, the last step is to seek labels from users. Numerous approaches are possible, depending on the platform. Examples include voice-based conversation agents [53][59][60], text response for screen-based hardware, and push notifications for mobile devices. As a proof-of-concept prototype, we programmed our device to act like a smart speaker that queries the user using a simple speech interface. When an unlabeled class in the model is activated, the system asks the user, “what was that sound?” immediately after the sound event. We use a commercial voice transcription web service to recover the utterance text and extract the last noun chunk using an off-the-shelf NLP package [25]. If no noun chunks are detected, the entire response is used as a label.

Privacy Preservation
While our acoustic approach to activity recognition affords benefits such as improved classification accuracy and incremental learning capabilities, the capture and transmission of audio data, especially spoken content, should raise privacy concerns. In an ideal implementation, all data would be retained on the sensing device (though significant compute would be required for local training). Alternatively, compute could occur in the cloud with user-anonymized labels of model classes stored locally.
HYPERPARAMETER TUNING
We adjust our system’s behavior using the following parameters: $\theta_{clf}$ (classifier acceptance threshold), $n_{\text{min}}$ (min. cluster size), $n_{\text{max}}$ (max. cluster size), and $\nu$ (number of support vectors) [64] of the one-class SVM. We conducted a series of preliminary studies to inform the design of our algorithm and for hyperparameter tuning.

Tuning Metrics
Traditional clustering metrics include cluster purity [28], conditional entropy [58], and other information-based approaches [73]. By running our generated model on a dataset, we can create cluster assignments conditioned on these metrics. We use an objective function that takes into account the classification performance on unseen data, based on the following equation:

$$\mu = \alpha \cdot F_1_{\text{accept}} + (1 - \alpha) \cdot r_{\text{accept}}$$

where $\alpha$ represents the weighted average between the F1 score of accepted samples and the models’ acceptance rate. As we later describe, $\alpha$ can be adjusted to influence characteristic system behaviors (e.g., relaxed or conservative).

Tuning Setup
We tuned our system’s hyperparameters empirically, using two datasets for audio events and environmental sounds. Specifically, we use the ESC-10 subset of the ESC-50 dataset (10 classes, 400 clips) [55] and UrbanSound8K (10 classes, 8732 clips) [63]. These datasets contain ground-truth labels, thereby allowing us to find optimal parameters for our system. Further, we randomly subsample 3000 0.96-second windows from UrbanSound8K to simulate a real-world usage period (10 event classes $\times$ 10 events per day $\times$ 30 days). Each dataset is shuffled and split into a tuning set (25%), used for hyperparameter tuning, and an evaluation set (75%), used later for our formal evaluation. We further divide our tuning set into three partitions for training (60%), holdout (20%), and testing (20%).

Tuning Procedure
The algorithm is initially executed using hyperparameters derived from the training set. The extracted unlabeled classifiers are labelled by randomly selecting samples from the holdout set (without replacement) and taking the ground truth label of the first instance that was recognized. This simulates our system’s labelling strategy of soliciting labels from the user the first time it is triggered. The ensemble model is then evaluated on the test set to calculate $F_1_{\text{accept}}$ and $r_{\text{accept}}$. This process is repeated 10 times (per hyperparameter combination), and the mean is used by the objective function. Finally, for our hyperparameter search, we use a parameter-free black-box optimizer [33] to maximize the objective function, and we run the optimizer for 50 iterations.

Algorithm Behavior & Results
We now describe three example system behaviors derived from characteristic hyperparameters:

- **Relaxed** (low $\alpha$) - This setting aims to cluster and classify as many sound events as possible, even when confidence is low. Although more sounds are recognized, accuracy is generally lower.

- **Balanced** (medium $\alpha$) – This setting produces an intermediate behavior that seeks to accept a moderate number of samples with usable levels of accuracy.

- **Conservative** (high $\alpha$) – This setting accepts new classes only when confidence is extremely high. This results in more events being unclustered (and thus ignored), but recognized sounds are more accurate.

These settings were acquired by manually fine-tuning the output of the black-box hyperparameter optimizer. For the ESC-10 dataset, we use $\alpha$ values of 0.4, 0.75, and 0.9 for Relaxed, Balanced, and Conservative, respectively. For the UrbanSound8K dataset, we use $\alpha$ values of 0.6, 0.8, and 0.9 for Relaxed, Balanced, and Conservative, respectively. The inverse relationship between F1 accuracy and acceptance rate is shown in Table 1.

In-The-Wild Data Collection
**Setup.** In addition to our preliminary experiments for hyperparameter tuning, we also performed a 10-day-long in-the-wild data collection. Because the primary motivation of Listen Learner is to support low-burden personalized acoustic activity recognition *in situ*, we wanted to run our system under real-world conditions to better characterize sound events present in entirely uncontrolled environments as a compliment to datasets like ESC-50 and UrbanSound8K.

**Procedure.** Our in-the-wild investigation was conducted across a period of one and a half weeks at seven locations (seven rooms, five buildings). Specifically, they include a mix of high-activity and low-activity environments: office, basement, kitchen 1, bathroom 1, living room, kitchen 2, and

<table>
<thead>
<tr>
<th>Location</th>
<th>Relaxed</th>
<th>Balanced</th>
<th>Conservative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office</td>
<td>38</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>Basement</td>
<td>10</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Kitchen 1</td>
<td>28</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Bathroom 1</td>
<td>6</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>Living Room</td>
<td>176</td>
<td>93</td>
<td>28</td>
</tr>
<tr>
<td>Kitchen 2</td>
<td>13</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>Bathroom 2</td>
<td>22</td>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 1. F1 scores and acceptance rates on the ESC-10 (ESC) and UrbanSound8K (U8K) datasets, based on different hyperparameter behaviors.

<table>
<thead>
<tr>
<th>Behavior</th>
<th>F1 Score</th>
<th>Accept. Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ESC U8K</td>
<td>ESC U8K</td>
</tr>
<tr>
<td>Relaxed</td>
<td>0.40</td>
<td>0.48</td>
</tr>
<tr>
<td>Balanced</td>
<td>0.97</td>
<td>0.79</td>
</tr>
<tr>
<td>Conservative</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 2. Number of discovered classes by room and their hyperparameter profiles. Living room had the most varied acoustic profile, primarily due to edge-case appliances e.g., TV.
bathroom 2. Recording consent was obtained from both the owners of the spaces and any visitors.

Results. For each behavior setting, we take the mean value between the ESC-10 and UrbanSound8K hyperparameter value for use on our in-the-wild collected dataset. An average of 41.9 (SD=55.7), 27.9 (SD=27.3), 9.9 (SD=7.5) classes were discovered by the system using the Relaxed, Balanced, and Conservative settings, respectively. It is possible that multiple classes are generated by the same object and would be given the same label by the user (e.g., microwave running and microwave door closing both being labeled as “microwave”). We report room-specific results in Table 2, which shows that our system can effectively discover classes in real-world environments, and hyperparameters tuned using our objective function produces consistent behaviors across datasets.

Evaluation
Our in-the-wild investigation was useful for characterizing key aspects of our system, but without reliable ground truth, it was impossible to quantify its classification and discovery performance. In response, we conducted a secondary evaluation wherein we collected acoustic data along with ground truth labels. We quantified system performance over time by categorizing model output into three possibilities: 1) correct (recognized sound belongs to a cluster and is classified correctly), 2) incorrect (recognized sound belongs to a cluster but is misclassified), and 3) ignored (sound event is ignored by the system).

Datasets
We conducted our evaluation with three datasets of varying sizes and class counts.

Preexisting Datasets. The first two datasets (ESC-10 and UrbanSound8K) were previously used for hyperparameter tuning. In this evaluation, we utilized unused data we specifically held out for this evaluation. However, we note these downloaded datasets were recorded with different microphones in different environments, and thus less representative of the type of data we envision for our system.

Environment-Deployed Dataset. In addition to the two existing datasets, we also collected real-world data from six environments over a one-week period using our sensing hardware. Following previous audio-based HAR work [34][41], we selected the following environments: an apartment bathroom, an apartment kitchen, a detached house bathroom, a woodworking shop, an electronics workshop, and a commercial office. For each environment, we selected 5-7 events of interest and recorded five 20-second clips of each event per day. Data collection was performed in a controlled setting (i.e., minimal competing events) and in the absence of any bystanders, to preserve their privacy. Mobile objects or actions that could be performed in different locations (e.g., speech, electric toothbrush) had their locations randomized in the room across recording sessions. The position and orientation of the recording device was kept constant across sessions.

We repeated this data collection process for one week. In total, we collected 1295 audio clips (432 minutes), resulting in 26,970 featurized samples.

Procedure
We followed an evaluation procedure similar to our hyperparameter tuning experiments. Specifically, we divided each dataset (two preexisting, one real-world) into training (60%), holdout (20%), and test sets (20%). We simulate the passage of time by gradually expanding the portion of the training set used by our system (i.e., using the next 100 samples for the offline datasets, or using timestamps for the real-world dataset). We further analyzed our system’s accuracy on the portion of accepted samples, breaking it down by class and comparing against two baseline models. Specifically, we measure the % of Correctly classified instances, % of Incorrectly classified instances, and % of Ignored instances (e.g., low confidence). It is possible to interpret our results using traditional metrics such as Precision (% Correct / % Accepted) and Recall (% Correct).

Results
In this section, we discuss evaluation results for key metrics, including accuracy, number of events recognized, the effect of directional data, performance across classes, and comparisons against baseline models.

Accuracy & Accept Rate
Figures 5 depicts our system’s accuracy over time. On the real-world dataset, our system achieved F1 scores of 0.59, 0.84, and 0.88 (for the Relaxed, Balanced, and Conservative settings, respectively) between accepted samples. Specific to the real-world dataset, the apartment kitchen environment achieved the highest accuracy after the one-week period (F1 accept=1.0). Both the Balanced and Conservative settings achieved F1 accept scores of 1.0, with accept rates of 0.39 and 0.16, respectively. The lowest accuracy on the real-world dataset occurred in the apartment bathroom using the Relaxed setting (F1 accept=0.42 and r accept=0.73).

The accept rates for our various hyperparameters remained consistent with their tuned behaviors. On the real-world dataset, our system reached accept rates of 0.66, 0.38, and 0.20 (for the Relaxed, Balanced, and Conservative settings, respectively). The Relaxed setting run on the apartment kitchen environment achieved a 0.85 accept rate with a F1 score of 0.64. On the other hand, the apartment bathroom environment had the lowest accept rate on the Conservative setting (F1 accept=1.0, r accept=0.03). At the dataset level, the highest accept rate was reached by the Relaxed setting on UrbanSound8K (0.67), while the Conservative setting for the same dataset had the lowest rate (0.16).

We observed key trends with our system’s performance over time. In general, the addition of more audio samples led to higher accuracy, due to the ability to form larger clusters and thus train more robust classifiers. Occasionally, novel outlier events caused accuracy to temporarily decrease (e.g., 4th day of ESC-10, 13th day of UrbanSound8K), but the system learns and accommodates this quickly. In all tested datasets,
there is a convergence point at which processing more data does not lead to significant improvements in either accuracy or accept rate.

Baseline Comparisons
To further contextualize the accuracy of our system, we also compared its performance against two baselines implemented using Scikit-Learn [52]. We selected two standard clustering algorithms for comparison: a soft-labelling clustering algorithm (GMM with full covariance) and a hard-labelling clustering algorithm (K-Means). The number of clusters for both algorithms were chosen by maximizing an objective score using brute-force search from 1 to 42 clusters (i.e., the average number of clusters from our in-the-wild data collection). The objective score of the GMM was the Bayesian Information Criterion (BIC), and the objective score of the K-Means algorithm was the gap statistic [71]. Other parameters were left at default values, and we found that varying them did not lead to significant changes.

In comparing Listen Learner’s performance to baseline clustering algorithms, we considered different types of errors and performance tradeoffs. The results of our experiments are shown in Table 3. Note that unlike the F1_accept metric, which is computed only for discovered classes, we computed Precision and Recall for all samples, with a caveat that our system may not discover all classes automatically. From a practical standpoint, we believe that fewer, more reliable classifiers are more useful for HAR applications, especially for end-users. In our future work section, we discuss methods for incorporating explicit user input in the class discovery process. Figure 6 shows each algorithm’s confusion matrix for the Apartment Kitchen environment, where all three algorithms performed relatively well (Table 3). Listen Learner achieves high Precision by ignoring samples with lower confidence, which is consistent with its lower Recall (Table 3). We believe this is acceptable for our use-case, as human activity is generally sustained across several windows.

Effect of Sound Direction
The use of our own sensing hardware for collecting a real-world dataset allowed us to integrate sound directionality information for identifying and classifying objects of interest. Figure 7 shows the distribution of the direction information for four classes in our data. In some cases, directionality information can be helpful in the recognition of stationary objects (e.g., Figure 7A). However, there are also cases where directional information does not form clean clusters, even for stationary objects (e.g., Figure 7C & D), which we hypothesize is due to multipath and poor autocorrelation with white-noise-esque signals (e.g., running faucet).

Table 3. Baseline comparison across 10 randomized cluster assignments. We show the mean macro-averaged Precision (P), Recall (R), and number of classes (C) discovered by Listen Learner. Standard deviations are shown in parentheses.
Overall, we found that the inclusion of directionality information does not increase classification accuracy (p=0.47) or accept rate (p=0.59). However, in some rooms (bathroom), the inclusion of direction can lead to increases in F1 score of up to +0.16. We hypothesize that many of our tested audio classes in that particular environment (e.g. faucet, urinal, toilet) produced similar sounds which were more easily differentiated using directional information.

Another assumption of using directionality information is that the position and orientation of the recording device does not change overtime. Our data collection procedure made this assumption and approximated a smart speaker whose location remained unchanged throughout data collection. In the future, using IMUs integrated in some consumer smart speakers (such as the Apple HomePod [5]), movement could be detected and trigger the system to recalibrate.

**INTERACTION STRATEGIES**

In our initial pilots, we explored “digest”-style labeling strategies, in which clips were replayed to users before a label was solicited. However, this required storing raw audio, which carries a significantly privacy cost. Thus, we focused on in situ labeling instead. While the primary mode of interaction we have discussed so far requires users to respond to open-ended queries (e.g., “what was that sound?”), there are other approaches for eliciting user input to label, confirm, and disambiguate classes.

One such approach is to use a pre-trained model to assign initial labels to clusters, and users are instead asked for a confirmation (e.g., “was that a microwave?”). This interaction can also be used to confirm previous labels and perform corrections if needed. Likewise, when a sound event falls within an ambiguous cluster boundary, the system can ask a disambiguation query (e.g., “was that a microwave or a faucet?”). Such techniques can streamline the labeling process and verify the correctness of a labelled classifier.

**Interaction Study**

We sought to explore and quantify several interaction strategies enabled by our system (i.e., open-ended, confirmatory, and refinement; Figure 8) through a follow-up study. We were particularly interested in 1) how each interaction modality was perceived by users, and 2) the corresponding accuracies of the resulting labels.

For this, we ran a user study with 12 participants (6M/6F, ages 21-35, average age 26.8, seven native English speakers). We selected five portable items (handheld sander, pitcher of water, hairdryer, fan, and hammer) from rooms used to create our real-world dataset (Apt. Bathroom, Apt. Kitchen, Wood Workshop, Electronics Workshop). For each item, we extracted the corresponding un-labelled classifier generated from our evaluation and used it to preload the system. We distributed the items around the study room and labelled them with numbers. We informed participants that their job was to “teach the smart speaker about what was going on around it.”

In each round of data collection, we programmed our system to ask participants one of the three query types: open-ended, confirmatory, or refinement. For refinement queries, our system made a best guess and then randomly selected a second class. Upon the prompt, participants were told to stop their current activity, respond to the query, and then continue. To investigate the “annoyance” of each interaction type, we also varied query frequency as a second condition. When the system recognized an audio event the probability (10%, 50%, or 90%) that a query would be presented. With 3 query types and 3 query frequencies, there were 9 rounds in total (randomized order). Within each round, participants used all five items for 20 seconds each, in a random order. Following each round, participants filled out a raw NASA TLX assessment.
At the end of all nine rounds, we conducted an exit interview asking participants about their experience with our system.

Results

The results of our study are shown in Tables 4 and 5. We were primarily interested in the effect different interactions had on our system accuracy and how users perceived our system. Thus, our metrics included Assignment Accuracy (i.e., portion of classes correctly labeled at the end of each session) and user feedback. While we asked users to fill out all six scales of a Raw NASA TLX assessment, we found that most scales had little variation, and so we report only the Frustration scale which had the highest variance. We believe this measure to be a reasonable proxy for annoyance and intrusiveness. We conducted statistical analysis of our results using Bonferroni-corrected unpaired t-tests.

At the end of each session (i.e., one round of using each item), around 90% of classes were correctly labeled by Listen Learner. This confirms that relatively low recall is acceptable for our use-case, as human activity is generally sustained. Altering the query frequency and type did not affect this in a statistically significant way. This is encouraging because it suggests that users can be queried relatively infrequently while still offering high accuracies.

As expected, participants became more frustrated with the system when it asked more questions, though this was also not statistically significant. In terms of utterance duration, Confirmatory queries had the shortest responses ($M_c = 1.51, \sigma_c = 1.21$) compared to refinement (p<0.01, T=5.7) and open-ended queries (p<0.01, T=11.2). Refinement queries produced the second shortest responses ($M_r = 2.27, \sigma_r = 1.33$), and Open-ended queries resulted in the longest responses ($M_o = 2.81, \sigma_o = 1.57$). Query type did not have a statistically significant impact on Assignment Accuracy or TLX Frustration, though we note on average refinement queries resulted in the least frustration.

Interestingly, when interviewed, participants gave different preferences. When asked to rank their preference of the different query types, 9 out of 12 participants said they preferred confirmation-style questions, noting it was “easier to answer” (P2, P4, P9). Compared to refinement questions, participants noted that confirmatory questions were preferred due to shorter questions, which were seen as requiring less mental effort to hear correctly and remember (P2, P5). Another factor was that since the second label for refinement questions was randomly chosen for this study, participants felt “discouraged when both the guesses were wrong” (P1, P4). Only one participant (P10) ranked refinement questions above confirmatory questions, commenting “I liked the comparisons because I knew exactly which inputs [the agent] expects”, which echoed his complaint that “the [agent] doesn’t always understand me.” Most users (10/12) became annoyed when the system asked too many open-ended questions, noting that the interaction became “repetitive, like a 2 year old” (P12). Nevertheless, one participant (P8) maintained that answering open-ended queries was easier because “I don’t have to think about the question, I just say what I’m doing”. Overall, participants preferred to have at most one or two open-ended questions per class followed by another type of query for any follow-ups.

Participants were also asked about their preference of question frequency, and almost all (11/12) participants preferred that a virtual assistant ask as infrequently as possible. Still, many offered situations where they found it acceptable for more frequent interactions. P4 noted that if he purchased a device with a contextually-aware agent, he expects the agent “to work out of the box for core stuff… I’m not here to teach the [agent] how to do its job…” but added “for things specific to my life, I would be ok with answering questions.”

Table 4. Metrics by trigger probability, query frequency, assignment accuracy (portion of correctly labeled classes), and TLX Frustration.

<table>
<thead>
<tr>
<th>Trigg. Prob.</th>
<th>Follow-ups/min</th>
<th>Assign. Acc.</th>
<th>TLX (Frus.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>3.63</td>
<td>0.88 (σ=0.16)</td>
<td>5.31 (σ=4.30)</td>
</tr>
<tr>
<td>50%</td>
<td>6.20</td>
<td>0.93 (σ=0.12)</td>
<td>5.39 (σ=4.16)</td>
</tr>
<tr>
<td>90%</td>
<td>7.20</td>
<td>0.88 (σ=0.13)</td>
<td>6.67 (σ=5.23)</td>
</tr>
</tbody>
</table>

Table 5. Metrics by query type, response length in # of words, assignment accuracy, and TLX Frustration.

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Resp. Length</th>
<th>Assign. Acc.</th>
<th>TLX (Frus.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-ended</td>
<td>2.81</td>
<td>0.88 (σ=0.14)</td>
<td>6.81 (σ=5.31)</td>
</tr>
<tr>
<td>Confirmatory</td>
<td>1.51</td>
<td>0.94 (σ=0.09)</td>
<td>5.43 (σ=4.47)</td>
</tr>
<tr>
<td>Refinement</td>
<td>2.27</td>
<td>0.87 (σ=0.16)</td>
<td>5.17 (σ=4.03)</td>
</tr>
</tbody>
</table>
participants also saw certain types of queries (e.g., confirmation questions) as a type of feedback, which would be beneficial for certain agent-controlled tasks, such as locking the door (e.g., “did you just go outside? I’ll lock the door”) when leaving the house (P8). Finally, other participants noted that their willingness to respond to queries depended on the interruption cost of their current activity (P2, P3, P5) – “If I was hammering, then I wouldn’t mind stopping, but having to stop the hairdryer and turn it back on was annoying” (P3). When asked to give a rough threshold of how many times they would be willing to be interrupted by an agent, responses ranged from 1 to 2 times per minute of performing the activity (P2, P7) to once an hour (P5, P12).

LIMITATIONS AND FUTURE WORK

The biggest limitation of Listen Learner is its inability to explicitly include classes of interest, and the high computational cost of model training (relative to traditional approaches). We speculate that accuracy could be improved with better embeddings (e.g., using ResNet instead of VGG) or improved settings (i.e., higher sampling rates). It is also possible to incorporate algorithmic changes that support learning through directed examples. Likewise, while our computational cost is relatively high, our algorithm does not need to be run in real-time and can be scheduled to run periodically (e.g., run overnight to process the day’s data).

Currently, our system is able to detect simultaneous events during inference, assuming instead that events are segmented and non-overlapping during training. However, in a real world setting, it is common for multiple events to occur simultaneously, making it difficult to segment audio based on our current adaptive amplitude threshold. We intend to investigate classical [11] and deep-learning based [27][31] audio separation (blind signal separation) techniques to further increase the efficacy of our algorithm. While these techniques have traditionally been applied to speech, they could be adapted to great benefit in this domain. In addition, other methods of clustering and classification can be used to better support the consideration of user-provided examples.

Finally, sensor fusion approaches are possible. Our evaluations show that our current hardware’s sound directional information does not significantly improve event clustering; however, this information may be useful for sound isolation using beamforming techniques [18]. In addition, other sensing hardware (e.g., motion, vibration, temperature) could also complement audio input to expand the set of activities accessible to Listen Learner.

EXAMPLE APPLICATIONS

We believe that the capabilities enabled by Listen Learner can serve as a foundation to enable many context-driven interactive experiences. In this section, we briefly describe several illustrative applications we implemented.

Wearables and Health. We built a smartwatch application that performs cross-modal learning of both acoustic and motion (e.g., IMU) information. Acoustic data streams are clustered using Listen Learner. Semantic labels from acoustic data are used to segment and train motion models. This multi-modal training scheme can then be used to extract e.g., a user’s health habits.

Home and Accessibility. We built a smart speaker application that leverages Listen Learner to label acoustic events to aid accessibility in the home. For example, the system can ask a confirmatory query: “was that a doorbell?”, in which the user responds with a “yes.” Once a label is established, the system can offer push notifications and other actions whenever the event happens again. This interaction links both physical and digital domains, enabling experiences that could be valuable for users who are e.g., hard of hearing.

Workflow Optimization. We built a desktop application plugin that clusters offline audio streams (e.g., podcasts, or field recordings) and prompts the user for labels. Once a label is provided, it is propagated across the entire audio stream. This workflow is a useful first-pass for tasks that involve audio annotation of extremely long recordings (e.g., WildDolphinProject.org, GreatElephantCensus.com, which requires finding animal sounds from 100+ hours of field recordings).

CONCLUSION

We have presented Listen Learner, a system that seeks to enable high-accuracy, low-effort acoustic activity recognition using one-shot user labeling. We built a hardware and software implementation that gradually discovers new event classes from the environment with no user demonstration or training involved. We designed our system to support a tunable parameter that prioritizes either the number of discovered classes or the model’s classification accuracy, and we evaluated each setting on two downloaded datasets and one real-world dataset collected using our own hardware. We also conducted a user interaction study that explored several approaches to user labeling. Our results show that Listen Learner provides accuracy levels suitable for common activity recognition use-cases and can augment or complement existing methods, bringing the vision of context-aware interactions closer to reality.

ACKNOWLEDGEMENTS

We thank our participants and reviewers. We also thank our collaborators at Apple: Miquel Espi Marques, Hyung-Suk Kim, and Carlos Avendano for their feedback and assistance. This research was funded in part by a NSF GRFP Fellowship.

REFERENCES


[16] Francois Chollet et al. 2015. Keras. https://keras.io


