AMIR: Active Multimodal Interaction Recognition from Video and Network Traffic in Connected Environments

SHINAN LIU, University of Chicago, USA
TARUN MANGLA, University of Chicago, USA
TED SHAOWANG, University of Chicago, USA
JINJIN ZHAO, University of Chicago, USA
JOHN PAPARRIZOS, The Ohio State University, USA
SANJAY KRISHNAN, University of Chicago, USA
NICK FEAMSTER, University of Chicago, USA

Activity recognition using video data is widely adopted for elder care, monitoring for safety and security, and home automation. Unfortunately, using video data as the basis for activity recognition can be brittle, since models trained on video are often not robust to certain environmental changes, such as camera angle and lighting changes. There has been a proliferation of network-connected devices in home environments. Interactions with these smart devices are associated with network activity, making network data a potential source for recognizing these device interactions. This paper advocates for the synthesis of video and network data for robust interaction recognition in connected environments. We consider machine learning-based approaches for activity recognition, where each labeled activity is associated with both a video capture and an accompanying network traffic trace. We develop a simple but effective framework AMIR (Active Multimodal Interaction Recognition)[1] that trains independent models for video and network activity recognition respectively, and subsequently combines the predictions from these models using a meta-learning framework. Whether in lab or at home, this approach reduces the amount of "paired" demonstrations needed to perform accurate activity recognition, where both network and video data are collected simultaneously. Specifically, the method we have developed requires up to 70.83% fewer samples to achieve 85% F1 score than random data collection, and improves accuracy by 17.76% given the same number of samples.

CCS Concepts: • **Computing methodologies** → **Activity recognition and understanding**; • **Human-centered computing** → *Ubiquitous computing*.

Additional Key Words and Phrases: multimodal learning, activity recognition, datasets

ACM Reference Format:
Shinan Liu, Tarun Mangla, Ted Shaowang, Jinjin Zhao, John Paparrizos, Sanjay Krishnan, and Nick Feamster. 2023. AMIR: Active Multimodal Interaction Recognition from Video and Network Traffic in Connected Environments. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 1, Article 21 (March 2023), 26 pages. https://doi.org/10.1145/3580818

[1]We open source our datasets, processed features, models, and analysis pipeline on https://amir-vidnet.github.io.

Authors' addresses: Shinan Liu, University of Chicago, Chicago, IL, USA, shinanliu@uchicago.edu; Tarun Mangla, University of Chicago, Chicago, IL, USA, tmangla@uchicago.edu; Ted Shaowang, University of Chicago, Chicago, IL, USA, swjz@uchicago.edu; Jinjin Zhao, University of Chicago, Chicago, IL, USA, j2zhao@uchicago.edu; John Paparrizos, The Ohio State University, Columbus, OH, USA, john@paparrizos.org; Sanjay Krishnan, University of Chicago, Chicago, IL, USA, skr@uchicago.edu; Nick Feamster, University of Chicago, Chicago, IL, USA, feamster@uchicago.edu.

2474-9567/2023/3-ART21 $15.00
https://doi.org/10.1145/3580818

1 INTRODUCTION

Healthcare researchers have long understood the value of in-home activity recognition [56]. Understanding the patterns in a person's daily living is essential to encouraging healthy diets, detecting cognitive or physical decline, and evaluating medical compliance. While prior work has considered a number of purpose-built sensors such as vibration and motion sensors/IMUs [21, 36, 40, 46, 53, 58, 64], there has been a recent convergence towards using cameras with general-purpose computer vision algorithms [13, 25, 27, 28, 41]. Cameras have several deployment advantages, as they can be used in any environment without specialized hardware and can detect both simple and complex activities with a single data collection interface. A well-placed camera directly captures the information needed to classify an in-home activity—as opposed to leveraging a potentially tenuous side-channel [2]. Furthermore, the availability of pre-trained computer vision models has democratized automatic visual perception making camera-based activity recognition more of a reality.

Unfortunately, it is well-known that the deep learning models commonly used in computer vision are susceptible to statistical failures where a training dataset may not have fully covered the "long tail" of activity variations that can be seen during deployment [32]. Correcting for such issues may require a significant amount of labeled demonstrations that are cumbersome to obtain. There are further failure modes that are simply fundamental to computer vision—occlusions, obstructions, and bad lighting—if one can't see the activity, one can't classify it. Thus, many of the advantages of camera-based activity recognition are only realized in controlled settings, where the camera's field of view can be kept clear and unobstructed, and the human subjects are instructed to perform activities in a visually consistent way.

As a consequence, we find that, unfortunately, purely relying on computer vision for activity recognition results in brittle software in realistic environments. This paper explores *how to augment visual activity recognition with alternate data without sacrificing its generality*. To this end, many in-home activities involve interactions with electronic devices, which are increasingly connected to a network. These connected devices shed off network traffic as subjects interact with them. A natural starting point is additionally using network traffic analysis for detecting device interactions [1, 6], which has primarily been studied in the privacy literature. Network as a data source to detect device interactions complements computer vision and potentially adds key redundancy that covers for visual failure modes (i.e., there might be network traffic evidence of an interaction even if it is not visually clear). *This paper advocates for the marriage of video and network data for a robust detection of device interactions.* Augmenting with network data has a few specific advantages since: (1) capturing network traffic does not require specialized sensing hardware and can be deployed in any home environment, and (2) it is agnostic to the physical arrangement of activity recognition objects and subjects [1, 6], which allows for a high level of generality in deployment.

We note that the device interactions are only a subset of all in-home user activities. However, the information obtained from these device interactions is useful, especially in the context of healthcare. For instance, excessive interactions with a smart fridge in a home are indicative of health issues. For simplicity, we use the terms *device interaction recognition* and *activity recognition* interchangeably with an understanding that the scope of the considered activities is limited to device interactions with associated network traffic. We consider machine learning-based approaches for activity recognition where labeled activities are associated with both a video capture and a network trace in a connected environment (*i.e.*, smart home, and more broadly speaking smart building, smart factory, and etc.). From an ML perspective, these data sources complement each other well, without sacrificing generality, where one captures the physical consequences of interaction and one captures the digital effects of those interactions.

This combination is not without technical challenges, since training models requires fusing information from two very different data sources. We propose a simple but effective framework called AMIR (Active Multimodal

[2]There are certainly privacy implications of using sources such as audio and video; we include a discussion of privacy in Section 6.

Interaction Recognition) that trains independent models for video and network activity recognition respectively, and then combines the predictions using a meta-learning framework. This approach crucially allows one to leverage specialized models for each data source separately, where even the training data can be collected independently. These models are combined with a small number of "paired" demonstrations where both network traffic and video are collected synchronously. A technical contribution of our work is understanding how to efficiently collect such data to train such models—a problem we call active multimodal interaction recognition. To solve this problem, we propose a heuristic that crucially reduces the amount of paired demonstrations needed. Using this method, it requires up to 70.83% fewer samples to achieve 85% F1 score than random data collection, and improves accuracy by 17.76% given the same number of samples.

Our exploration of using both network and video data for activity recognition yields the following findings that we believe can help the community build next generation of in-home activity recognition systems:

- Network traffic is a surprisingly robust data source of activity recognition even with general-purpose flow features, i.e., they don't have to be changed across tasks or environments, across a variety of internet-capable home appliances [63].
- Individually, the video-based model and the network-based model achieve comparable accuracy, but together they are significantly more accurate than either individual source.
- The best way to combine these sources together is a meta-learning algorithm that efficiently combines independently trained video and network models. This strategy minimizes the number of "paired" examples needed, which are often more expensive to collect.
- An uncertainty-minimization heuristic can actively choose what training data to collect to further reduce labeling costs.

2 MOTIVATION AND RELATED WORK

In this section, we motivate this study with key prior work.

2.1 Video/Network Activity Recognition

Visual solutions are appealing because of their generality. First, let's consider the alternative, e.g., using an accelerometer to detect whether a refrigerator door is opened in a smart kitchen. This sensor will have to be specifically placed by an expert to recognize this activity, any activity recognition task must be calibrated to the specific sensor values, and any classification of activities beyond opening the refrigerator door will be indirect at best. On the other hand, a single camera capturing video is far more flexible. Cameras are widely available, do not require expert installation, and can detect a wide variety of device interactions. However, the use of cameras shifts the sensing burden from hardware design to software since classifying such real-world activities requires advanced computer vision algorithms. We briefly survey the state-of-the-art in this space.

Activity recognition in video has been studied from both a geometric perspective [17, 18, 45, 51, 62] and an end-to-end neural network perspective [13, 25, 27, 28, 41]. In this work, we use a pre-trained SlowFast [28] neural network architecture to featurize our data, and train a separate output layer to make predictions for our classes. Due to the nature of video data, video-based models are infamously sensitive [31, 61]. Researchers have developed architectures that are robust to such issues, but such work is not yet fully mature [38].

Adding additional modalities can improve the performance of video-based models, but existing video activity recognition datasets are mostly single-modality. For example, Kinetics [19, 20, 35, 52], AVA [37] and ActivityNet [26] were collected from various environments annotated with hundreds of classes. Some existing work fuses audio-visual data for human action recognition [9, 50]. Others have considered data modalities including inertial measurement units (IMUs) [46, 53]. There are datasets with such sensor data including the CMU Multi-Modal Activity Database (CMU-MMAC) [23] that records coarse-grained human activities in a kitchen with

multi-modal data sources, including video, audio, motion capture, and multiple sensors. Similar datasets include TUM Kitchen [57] and 50 Salads [54]. These multi-modal scenarios require complex setups of motion capture devices and sensors, such as inertial measurement units (IMUs). To the best of our knowledge, no existing work combines video and network traffic together as two modalities for activity recognition. We argue that although these multimodal approaches tried to include data sources from different modalities, they are only accessible from a lab setting. It is almost impossible to democratize these sensors in real-world kitchens. However, our approach is applicable to every kitchen with internet-connected devices.

Activity recognition using network traffic has been explored in number of prior works [7, 8, 49, 60]. These works show promising, albeit limited, results for activity recognition using encrypted network data. The primary focus of these works, however, have been on the privacy implications of being able to detect in-home activities purely from network traffic. Due to the ubiquity of networked appliances, we argue that network traffic can be considered a first-class citizen along with the video for in-home activity recognition. Increasingly, there are general-purpose models and techniques for featurizing network traffic [11]. We leverage general-purpose network features for activity recognition [63].

2.2 Other Activity Recognition Modalities

There are numerous sensing modalities that have been applied in this context. However, we argue that there is a generality to video and network data not seen in these purpose-designed systems. Table 1 compares and contrasts the different approaches. Video and network traffic are two accurate sources, which can be collected from universal interfaces, such as cameras and routers. While the video is susceptible to environment changes, it generalizes across appliance types for activity recognition. Network traffic, on the other hand, is tied to devices, but it is environment-independent. These complementary characteristics are unseen in the activity recognition modalities below, thus making them a good couple.

2.2.1 Sensor-based Activity Recognition. Recently, there has been a rise in sensing technology along with the popularity of "smart home" experiences [36]. Researchers have developed different methods to recognize human activities from sensor data [21, 58]. Deep learning has also been used for human activity recognition with wearable sensors [40, 64]. While being accurate, these sensing components often require special hardware [36, 64] and can be costly to deploy correctly.

2.2.2 Audio-based Activity Recognition. Since most human activities of daily living (ADL) produce characteristic sounds, it is possible to do activity recognition purely based on audio data [39, 55]. However, audio-based approaches are easy to be affected by ambient noises [39] and are highly dependent on the material of devices.

2.2.3 Powerline-based Activity Recognition. Researchers have developed tools to measure the power consumption of each device [12, 15, 24]. With these measurements, it is possible to identify appliances based on distributed load metering data [47]. Although universal, this side-channel information is hard to acquire [12, 15, 24, 47].

2.2.4 Wireless-based Activity Recognition. Researchers have used RF signals for various tasks, including motion tracking [3], multi-person localization [2], breathing and heart rate monitoring [4], and even emotion recognition [65]. Human activity recognition is also a popular task for wireless signals [42, 59]. An advantage of RF-based methods over sensor-based counterparts is that users no longer need to carry any physical device. However, these methods are largely impacted by the indoor environment. Due to multipath effects, many works in this line need prior knowledge like floor plans, home layouts [2, 42] to achieve such goals, which makes the accuracy in real environment far from ideal.

Table 1. Activity recognition sources.

	Universal interface	Environment dependence	Device dependence	Accuracy	Literature
Video	✓	✓		✓	[13, 25, 27, 28, 41]
Network traffic	✓		✓	✓	[7, 8, 49, 60]
Vibration sensors			✓	✓	[21, 40, 58, 64]
Audio	✓	✓	✓		[39, 55]
Powerline	✓		✓		[12, 15, 24, 47]
Wireless	✓	✓			[4, 42, 59, 65]

2.3 Multimodal Learning

The world is naturally multimodal, and we have data generated from diverse sources. Multimodal learning builds machine learning models that integrate data sources from multiple modalities, so we can take advantage of complementary information from different sources [10]. There are two main categories of multimodal learning: early fusion and late fusion. Early fusion combines multiple modalities at the feature level and conducts most computations after the combination. Late fusion, on the other hand, encourages the use of separate models for different modalities and applies a light model after the combination.

The traditional machine learning literature focuses on the accuracy and generalization considerations of different types of combinations. Our work studies the labeling cost of different combining different modalities. In particular, we consider a setting where there is a mix of individually (where only one modality is captured) and paired labeled data (where both modalities are captured simultaneously). We also consider algorithms for actively choosing which classes to demonstrate in the paired setting.

Multimodal learning is a novel problem setting related to two sub-areas of machine learning called Active Learning [48] and Semi-Supervised Learning [66]. In the former, a learning algorithm can interactively query a user to label new data points with the desired outputs. In contrast, we consider a scenario where the data points themselves have to be collected. The key axis for optimizing is choosing which classes of data points to collect in the paired setting. This problem is also related to semi-supervised learning which combines a small amount of labeled data with a large amount of unlabeled data. In contrast, we explore combining a small amount of paired labeling with a large amount of individually labeled data.

3 ACTIVE MULTIMODAL INTERACTION RECOGNITION (AMIR)

In this section, we describe the core technical contribution of this work, namely, an algorithm for multi-modal interaction recognition problem using network and video data.

3.1 Problem Statement

We consider an environment with a set of networked appliances. A human subject in this environment can interact with each of the devices in one or more "activities". Let $\mathcal{Y}$ be a set of such activities that we wish to classify, including an "idle" (no activity) class. From this environment, we collect two time-aligned streams of data x_{video} and $x_{network}$. The multi-modal activity recognition problem is to produce a model that determines which activity, if any, is currently occurring:

$$y \approx f(x_{video}, x_{network}),\ y \in \mathcal{Y} \tag{1}$$

Table 2. Table of notation.

$\mathcal{Y}$	a set of activities we wish to classify
y	an activity label ($y \in \mathcal{Y}$)
x_{source}	a time-aligned data stream of *source*
$\mathbf{Y}$	ground truths of a labeled dataset
$\mathbf{X}_{source}$	the entire training data from a labeled dataset of *source*
f, g	a model that predicts activity labels based on data streams
θ_v, θ_n	model parameters for video and network models
h_m	a meta-learning model with parameters m learned from data
W	weights for each class
U	uncertainty, defined by the difference between 100% confidence and the most confident prediction

There are many ways to realize equation 1, and we describe some of the key metrics that we will consider in this work. First, we will primarily focus on supervised machine learning approaches to constructing f, namely, where labeled training data is used to fit a model. That said, the notion of labeled training data is a little subtle in multi-modal classification.

Demonstrations. A demonstration is a labeled instance of a human performing one of the activities in $\mathcal{Y}$.

Paired Labeling. A paired labeling is a demonstration where both time-aligned streams of data (x_{video} and $x_{network}$) are collected and the demonstration is associated with a label $y \in \mathcal{Y}$.

Individual Labeling. A individual labeling is a demonstration where one only one of the data streams, either x_{video} or $x_{network}$, is associated with a label y.

These two types of labelings are interesting to consider because they have different costs. Individual labeling for network traffic can be collected offline in a lab environment for each device. Similarly, for many devices, we notice that there is a significant lag relationship between an interaction and its induced network traffic, whereas video data collection induces signal in real time. Thus, even for video data, individual labeling is significantly faster to collect. In contrast, paired labelings are more costly because they require both streams of data to be time-aligned and synchronized. However, during inference, the model will have to consider time-aligned data streams. So, we want to use individually labeled data as much as possible while not sacrificing too much accuracy over fully paired data collection.

Active Multi-Modal Activity Recognition Problem Statement. Let $(\mathbf{X}_{video}, \mathbf{Y})$ be an individually labeled training set of video demonstrations, and $(\mathbf{X}_{network}, \mathbf{Y})$ an individually labeled training set of network demonstrations. Given these individually labeled demonstrations, we collect a small set of demonstrations with paired labelings $(\mathbf{X}_{both}, \mathbf{Y})$ to improve the accuracy of the model. Our algorithmic objective is to determine which classes to demonstrate to improve the accuracy of the model as quickly as possible ,i.e., with as few paired demonstrations.

3.2 Meta-Learning For Multi-Modal Models

The core challenge is designing a model that can train on a mix of individually labeled data and paired labeled data. To do this, we leverage meta-learning. Meta-learning refers to machine learning algorithms that learn from the output of other machine learning algorithms.

Let's suppose that we have trained two models on the individually labeled data. We can think of a model as a parametrized conditional probability distribution, i.e., a probability over the activity classes given a particular set

of features:

$$f_{\theta_v}(x_{video}) = Pr[\mathbf{Y}|\mathbf{X}_{video}] \qquad g_{\theta_n}(x_{network}) = Pr[\mathbf{Y}|\mathbf{X}_{network}]$$

f_{θ_v} and g_{θ_n} can represent any type of a parametric model, e.g., SVM, Logistic Regression, Random Forest or a Neural Network, where θ_v and θ_n describe the model parameters.

A meta-learning model is a parametrized function h that combines f_{θ_v} and g_{θ_n}. This is a function of the predicted class probabilities. We can think of the meta-learning model as a function of two vectors of class probabilities, it defines a consensus scheme between these two predictions:

$$y = h_m(f_{\theta_v}(x_{video}), g_{\theta_n}(x_{network}))$$

The parameters of the meta-learning model m are learned from data, i.e., it learns how to integrate those probabilities.

To train such a meta-learning model, one needs to first train f_{θ_v} and g_{θ_n} on their respective individually labeled datasets. Then given finalized models for f_{θ_v} and g_{θ_n}, the meta-learning model h_m is trained with a smaller amount of paired examples. One constructs a meta-learning training dataset $(f_{\theta_v}(\mathbf{X}_{video}), g_{\theta_n}(\mathbf{X}_{network}), \mathbf{Y})$ where the features are the prediction vectors from f and g, and the labels are the original labels. The model h is then trained to learn how to combine those prediction vectors. This scheme allows us to train models on a mix of individually and paired labeled data.

To summarize, meta-learning models have a number of crucial advantages over simply combining the features $\mathbf{X}_{video}$ and $\mathbf{X}_{network}$ together (which we call a monolithic architecture):

- f and g can be designed and implemented in very different ways and potentially using different software libraries. A monolithic architecture would require that both data sources are converted to a common feature format.
- f and g can be trained on a mix of individually labeled and paired labeled data. A monolithic architecture would require all training data to be paired.
- The approach can be easily scaled to multiple different data sources.

3.3 Active Collection of Paired Examples

Next, we consider techniques for selecting which paired examples to collect. We assume that we already have trained individual models f and g for video and network data respectively. The goal is to derive a distribution of weights $W(f, g)$ for each class to guide the active paired example collections, a step before the actual training of the meta-learner. The paired data for each class is then collected in proportion to the derived weights. For instance, if the weights for a 3-class classifier are [0.5, 0.3, 0.2] and training data is collected in batches of 10, then during the first batch, we collect [5, 3, 2] samples from the corresponding classes. This process then can be repeated for the next 10-sample batch, and so forth. The question then is how to derive the weights. A naive method is to use equal weights, i.e., to sample equally from all classes. However, it can lead to redundant data collection. We design our approach based on one insight: paired examples are most valuable in classes where neither the f or g model is very confident in its prediction, *i.e.*, the prediction probability has the highest entropy.

We assume that during individual model training, there's a cross-validation process. Each sample from a cross-fold validation set serves as a probe. We derive a set of probability vectors, $\{P_f(x^i_{video})\}$ and $\{P_g(x^j_{network})\}$ for each video sample x^i_{video} and network sample $x^i_{network}$ in the cross-validation set, respectively. Here, $P_f(x^i_{video})$ is the probability vector output by f while predicting on x^i_{video}, indicating the prediction probability of each class. It should be noted that every prediction probability in $P_f(x^i_{video})$ or $P_g(x^j_{network})$ is between 0 and 1 and the probability across classes add to 1.

Based on these probability vectors, we compute an uncertainty score for each sample. We compute the difference between 100% confidence and the most confident prediction, referred to as least confidence, as the uncertainty:

$$U_f(x^i_{video}) = 1 - \max P_f(x^i_{video})),\ U_g(x^j_{network}) = 1 - \max P_g(x^j_{network})$$

The lower the maximum probability for the sample, the higher will be the uncertainty score. For example, if we have three classes $[0, 1, 2]$ and classification probabilities $[0.1, 0.1, 0.8]$, then the corresponding uncertainty score will be 0.2. Each of these uncertainty scores are then grouped into classes based on the actual label of the corresponding sample. We then obtain the average uncertainty score for each class. This essentially gives us weights $W(f), W(g)$ for each class on classifier f and g. We also compute the F1 scores of each class (*i.e.*, $F(f)$ and $F(g)$) as a baseline comparison approach. A final decision is made by seeking the average of $W(f)$ and $W(g)$. Thus, we are able to find the distribution of weights to follow during collection that $W_{uncer}(f, g) = Normalize[(W(f) + W(g))/2]$.

We also test metrics like margin of confidence, ratio of confidence, and entropy [44] to measure the prediction uncertainty. They present similar performance and for brevity we omit them in the description and evaluations.

4 DATASET AND MODELS

We evaluate our multimodal interaction recognition framework using paired network and video data collected in the lab. We also collect data from a home environment to evaluate the transferability of our approach. In this section, we first describe our experimental setup followed by the list of activities for which we collect video and network data. Then, we describe the video and network-based activity classifiers including the featurization of video and network data and the associated machine learning model. The purpose of this section is to demonstrate that our activity recognition method (i.e., data collection, featurization, and modeling) is general and can be applied in many different ubiquitous computing settings.

4.1 Testbed

4.1.1 IoT Lab. The IoT lab, as depicted in Figure 1a, consists of a variety of devices that are typically found in a smart home. All IoT devices in the lab are connected using a dedicated WiFi with an upstream connection to the university network. The traffic from the WiFi router is also mirrored to an NVIDIA Jetson Nano, a single-board computer. The Jetson Nano is then used to capture and process the network traffic. In addition to network traffic, we also collect video data for activity using one stationary camera connected to the Jetson Nano. [3] Connecting the camera to the Jetson Nano allows us to automate data collection during the experiments.

Table 3. Data collection for testbed in IoT Lab.

Device	Location	Data collected	Data format
Router	1 in the lab	Network traffic	pcap
Stationary Cameras	1 in the lab	Video	mp4

4.1.2 IoT Devices. To cover a rich variety of devices , the IoT Lab is equipped with 32 IoT devices which are selected based on their commercial shares. We divide these into six different categories based on their types of interactions (see Table 4): computer, mobile device, smart appliance, home automation, smart agent, and surveillance. The first three categories often require physical (e.g., mechanical, touch) engagement to operate

[3]Videos from an extra camera at a different angle are also collected (but not necessary to our pipeline). They are released along with the dataset.

(a) The IoT Lab

(b) Sample activity: open close fridge.

Fig. 1. Sample images for the IoT Lab and the data collection.

Table 4. Devices in the IoT Lab.

Category	**Device**	**Interaction**
Computer	Personal laptop	Physical
Mobile device	Android / iOS smartphone	Physical
Smart appliance	*Samsung smart fridge*, *General Electric washer* and dryer, Samsung smart TV, Samsung smart stove, Samsung dishwasher, Bose wireless speaker	Physical, remote
Smart agent	Google Home, *Amazon Echo*	Proximate, remote
Home automation	Philips light bulbs, TP-link smart plug, *iRobot smart vacuum*	Proximate, remote
Surveillance Camera	*Nest camera*, Wyze camera, Ring camera	Proximate, remote

on those devices, which can be potentially captured by video footage and network traffic simultaneously. The latter 3 categories, on the contrary, often require non-physical but *proximate* (i.e., in the same house or room) interactions. Because of this characteristic, this type of interaction sometimes can not be completely captured by a fixed-lens camera. A few device categories (e.g., smart appliance) also sometimes involve *remote* interactions (e.g. control, schedule outside the room) through smartphone applications, which leave minimal or no traces on video but relatively greater signal on network. As we will show in the later section, using network data can yield higher accuracy for such *proximate* and *remote* interactions as compared to using video data.

4.1.3 Data Collection and Labeling Pipeline. We develop an easy-to-deploy web-based interface to facilitate the data collection and labeling process. The interface consists of a home page containing instructions as well as controls to start or stop the data collection for participants. On the back end, the server implements the data collection pipeline. The network data is collected using `tcpdump`, while the video data from the two cameras is collected using `ffmpeg`. This pipeline generates labels in real-time during the course of paired video and network traffic collection, it also automatically logs metadata like participant ID, repetition number, activity, device, and

timestamp. A deliberate design choice we make is that between two collections, we collect 20 seconds of data as paddings. This is to accommodate the lead-and-lag phenomenon that we observed in some cases, i.e., the time asynchronousness between video and network traffic.

A participant who is collecting a new set of data needs to enter a participant ID to retrieve a tailored data collection script with a set of randomly shuffled activities. The script for each activity contains a list of sub-tasks that the participant needs to perform to finish the activity. For instance, to collect "open close fridge" (see Figure 1b), the script pops up like "`Walk to the fridge. Open the door. Wait for 2–5 seconds. Close the door. Walk away.`" Note, however, the script is vague about some specifics including the walking direction, which fridge door to open, with which hand, and the exact waiting time. This leads to a natural variation in the collected data across multiple runs for the same participant and across participants. After the participant is done with the demonstration, the participant presses the end button to record the execution time. If a mistake happens, the participant can record the mistake at any time and the demonstration restarts itself automatically. We discard the data for such failed transactions. Detailed scripts and encoded naturalness for all the collected activities can be found in Table 10 (Appendix B.)

4.1.4 Smart Home. To evaluate the transferability of our approach across environments, we also collect videos and network traces within a residential apartment. The smart home (shown in Figure 8, Appendix B) uses the same data collection mechanisms for paired data captures. We specifically collect data for an iRobot smart vacuum from this environment.

4.2 Dataset Descriptions

To the best of our knowledge, we collect the first dataset that explores the paired observations on video and network traffic (to facilitate future explorations in this direction, we open source our datasets, processed features, models, and analysis pipeline on https://amir-vidnet.github.io). As shown in Table 5, we first choose four most representative devices to collect both physical and proximate interactions. From May 2020 to March 2022, 1960 demonstrations are collected in 3 rounds (about 22.14 hours long data, equivalent to 30 hours of human labor with overhead), with an equal amount of demonstrations across activities for fridge, washer, Alexa and Nestcam. Each demonstration is 40.67 seconds on average of paired observations, with a minimum of 21 seconds and maximum of 138 seconds. Note that to accommodate the diversity of user behaviors, the activities are collected by 7 different users. Camera angles and IoT Lab environment may slightly change from round to round.

We select activities that are important to record the daily usage of each device. For example, on a smart Samsung fridge, we collect screen interactions, open close fridge, take out and put back items, which help us to monitor the usage of such device, shedding light on the diet health (food intake) of a user. The "check on device" activity means that a participant approaches closer to a device to see what's going on, inspecting whether it is still working or under errors with no physical touch or mechanical movements on the device. In addition, each device has an idle state, where there is no human in the IoT Lab. We collect 648 idle traces in total. While performing per-device classification, we subsample the idle cases to make the amount of data similar to other classes. This is done to maintain class balance.

In October 2022, an additional round of data (3.14 hours of data, based on six hours of human labor with overhead) is collected, both in the IoT Lab and the smart home, to test the transferability of approach across physical environments. The data is collected on a robot vacuum, a unique device that differs from others in its capacity to move. The robot vacuum can be controlled both by on-surface physical buttons and smartphone apps. We also consider a new type of activity, i.e., remote interaction through the app, which is not covered by our previous collections. Each demonstration is on average 37.33 seconds (ranging from 21 to 69 seconds) of paired observations.

Table 5. Activity list and dataset description. Some devices have more number of demonstrations than others because the classification is harder. Some demonstrations are collected in both lab and home environments. The colored IDs are also referenced in later sections for a clearer presentation.

Device	Activity	# of Demos	Environment	ID
All	idle	699 / 50	Lab / Home	I
Fridge	check on fridge	144	Lab	F_1
	screen interaction	145	Lab	F_2
	open close fridge	155	Lab	F_3
	take out item	151	Lab	F_4
	put back item	149	Lab	F_5
Washer	check on washer	96	Lab	W_1
	start washing	113	Lab	W_2
	end washing	113	Lab	W_3
Alexa	check on Alexa	49	Lab	A_1
	ask weather	49	Lab	A_2
	play music	49	Lab	A_3
Nestcam	ask time	50	Lab	N_1
	trigger motion detection	50	Lab	N_2
Vacuum	check on vacuum	50 / 50	Lab / Home	V_1
	remote control	50 / 50	Lab / Home	V_2
	click button	50 / 50	Lab / Home	V_3

4.3 Featurization for Video and Network Data

To investigate further on the combinations of video and network modalities, we describe our featurization approaches that consider the State-Of-The-Art (SOTA) models from both sides. Note that these models are plug-ins that could be replaced by other advanced models.

4.3.1 Video. According to Section 2.1, video-based activity recognition models often have two approaches: pose estimations and human action categorization. We mainly focus on human action categorization because of a more extensive literature base [13, 25, 27, 28, 41] and actual deployments [22].

SlowFast pretrained model. We look into SlowFast networks for activity recognition [28]. The model processes the videos at a low frame rate to capture spatial semantics, and at a high frame rate to capture fine-grained motions temporally. We leverage the SlowFast model pre-trained on Kinetics-400 [34] using 8x4 GPUs, which achieves top-1 accuracy of 76.94% and top-5 accuracy of 92.69% [1]. This SlowFast model has 34.57 Million parameters, and we take the second to last layer output to create our feature group. The intuition behind this was that the last few layers of the pre-trained SlowFast model can represent the data in a lower-dimensional latent space. We tested the impact of different layers of output and found that using the 2nd-5th last layers perform quite similarly (67 to 73% F1 scores for the 15-class video classifiers) to each other (See Appendix A). We then select the penultimate layer as it minimizes the number of input features. Based on these samples, we train a shallower One vs. Rest Logistic Regression classifier to fine-tune the pre-trained features on our datasets.

4.3.2 Network. One major difference between video and network activity recognition is that the latter does not have pretrained models. Therefore, we search to find the most relevant feature representations to build our model. The representation of network traffic is still an open problem. Recent years have witnessed some major efforts in

standardizing machine learning representations for network traffic [33, 63]. nPrint [33] presents a per-packet representation while NetML [63] creates features based on flow-level summary statistics. We find NetML to be more suitable and fairly lightweight for this task. Using nPrint, on the other hand, would lead to a dimensionality explosion, especially given the network data as time series. Thus, we select and modify NetML for this task.

NetML features. We make modifications on the NetML library to extract pcap-level features. We essentially take all inbound and outbound flows into two groups, each containing statistics features of these aggregated flows. The features include packet rates, byte rates, distribution of packet sizes, distributions of inter-arrival time, and the number of flows from both directions. We test multiple classifiers and Random Forest is the best of them.

5 EVALUATION

In this section, we evaluate the effectiveness of combined modalities first by baseline comparisons, given different evaluation targets. We show that all modal combination methods outperform single modality. The stacking (the main meta-learning approach) classifier that combines video and network traffic shows more efficiency than feature fusion (a monolithic architecture) and more adaptiveness than soft voting (another meta-learning approach). Then, we evaluate the proposed active collection algorithm of AMIR. Given different assumptions on the classifiers, we seek to understand how well the active collection performs and under what conditions it fails. Finally, we evaluate the transferability of our approach to a different physical environment.

5.1 Evaluation Settings

Practically, two approaches can be used for real-world activity recognition deployment: a tree of hierarchical per-device classifiers, or a uniform classifier that involves all devices. We consider both cases.

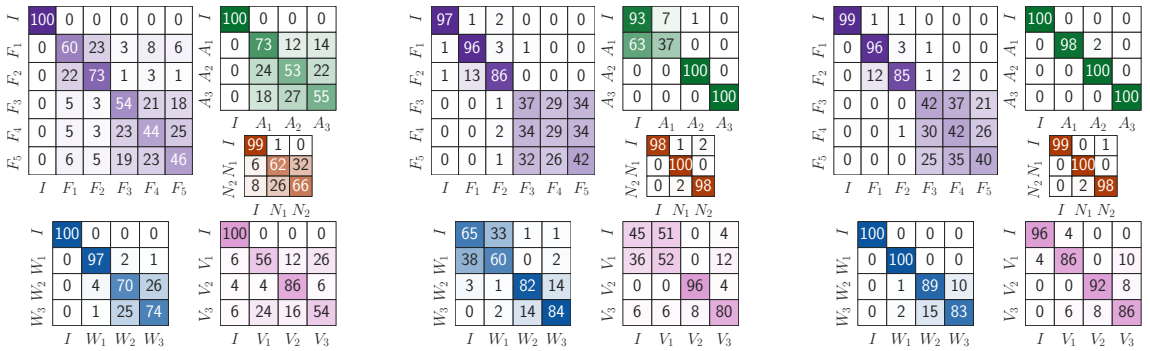
5.1.1 Tree-based Structure for Per-device Classifiers. A tree-based hierarchy contains a device identifier as well as several activity classifiers for each device. It provides the flexibility to add, remove, or update each per-device classifier, but it requires knowledge of device identity. This knowledge, however, is not hard to obtain from modern routers' address resolution protocol (ARP) data; it can also be inferred through widely explored device identification mechanisms using network traffic [5, 43]. We can either use the device-IP (or MAC address) mapping relationship provided by a homeowner, or we can first isolate the flows by IPs and use a device classifier to help us to get the device-IP relationship. A per-device network classifier inquires IP-filtered pcap files to have clean source and destination flows. And a per-device video classifier focuses on per-device activities. We evaluate both the device classifier and per-device activity classifier to show the effectiveness.

5.1.2 Uniform Classifier for All Devices. Another approach does not require IP or MAC addresses and is easier to manage, but it loses the flexibility to scale (*i.e.*, each time it scales, it needs an entire retrain on the data). It is to train on the entire dataset for the task of activity classification. In this case, the biggest difference with the per-device classifier is that the uniform network classifier does not require IP filters, the pcaps contain all the traffic from the 5 devices we tested.

5.2 Effectiveness of Combination Methods

To understand the prediction power of meta-learning combination methods compared to single video and network classifiers, we conduct an accuracy evaluation and cover a variety of classification settings.

5.2.1 Confusion Matrix. We compute the confusion matrix based on 5-fold cross-validations (*i.e.*, training testing split is 4:1). We test the video classifier using SlowFast [28] intermediate outputs, network classifier using pcap-level NetML [63] features, and a stacking classifier (a meta-learning approach that uses Random Forest as the meta-learner) that leverages the prediction probability vectors from both learners. Device classification results can be found in Appendix A, and we focus on per-device classifiers as well as uniform classifiers in the main text.



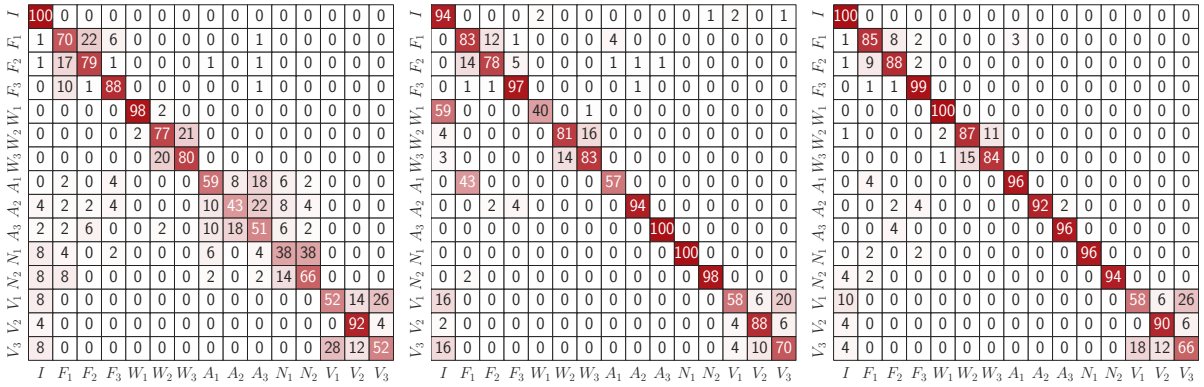
(a) Video classifier. (b) Network classifier. (c) Stacking classifier.

Fig. 2. Confusion matrices of individual classifiers and stacking classifier for each device. The results shown here mean accuracy (%). Use Table 5 as the key for activity names. • Fridge, • Washer, • Alexa, • Nestcam, • Vacuum.

As illustrated in Figure 2, we present activity classifications of the fridge (6 classes), washer, Nestcam , Alexa, and vacuum before and after combinations. Overall, the stacking classifiers improve upon accuracy, precision, and recalls, compared to the base models. For the washer classifier, "start washing" (W_2) and "end washing" (W_3) are harder to tell from the video classifier, because of similar movements. "Idle" (I) and "Check on washer" (W_1) are harder to get from network traffic due to no signal on the network. The stacking classifier largely improves on these two confusion cases. Similarly, the F1 score increases from 69.72% ± 6.31% (95% confidence interval, video model) and 82.91% ± 3.41% (network model) to 99.54% ± 0.91% (stacking combination) for Alexa activity classification. For the vacuum, both video and network models have different difficult classes: "check on vacuum" (V_1) and "click button" (V_2) are hard to distinguish on the video classifier leading to an overall accuracy of 72.22% ± 4.48%; for the network classifier (66.95% ± 4.98%), "idle" (I) and "check on vacuum" (V_1) are challenging to distinguish due to lack of network data generated during the interactions. The stacking classifier leads to significant performance improvement (88.19% ± 3.55%) by merging signals from both sources. For the Nestcam, although the stacking model does not improve much upon an already well-trained network model (97.41% ± 3.20%), it keeps the same level of performance even when video classifier fails to distinguish N_1 and N_2.

For the 6-class fridge predictors, despite the continued confusions from F_3 to F_5, the meta-learning approach is able to pick up the best prediction from two individual classifiers. For example, "check on fridge" (F_1) and "screen interaction" (F_2) share similar characteristics on video footage, but in F_1 a motion sensor triggers the fridge to wake up and send network traffic, which differs from the F_2 traffic patterns. "Open close fridge" (F_3), "take out item"(F_4), and "put back item"(F_5) are especially hard to distinguish and the network model nearly makes random guesses among them. Thus, the failure in the combination roots in the inability to classify from base learners. So we also test the 4-class fridge classifier and 15-class all-device classifier without F_4 and F_5 in later comparisons.

In Figure 3, we test the uniform classifier with 15 classes. The average F1 score climbs up from 69.58% ±2.56% using only video (and 81.48% ±2.67% using only network) to 88.90% ±1.36% using stacking. Also, in the stacking confusion matrix, fewer mistakes are made across devices. The network classifier, on the contrary, has 43% of A_1 mistaken as F_1. We also find that V_1 (check on vacuum) and V_3 (click button) get confused with each other across all models, although there is a slight improvement in the stacking classifier. A potential reason is that



(a) Video classifier. (b) Network classifier. (c) Stacking classifier.

Fig. 3. Confusion matrices of individual classifiers and stacking classifier for 15-class uniform classification. Results shown here mean accuracy (%). Use Table 5 as key for names.

both activities have reduced video and network signals in this all-device uniform classifier. The video for both activities is quite similar, consisting of a participant walking in front of the vacuum and reaching down to it. As per network data for these activities, we find that clicking buttons only generates 4 to 6 packets, which is likely harder to distinguish when traffic from other devices is also considered in the uniform classifiers.

5.2.2 Baseline Comparisons. We further test the effectiveness of stacking by comparing it with two other baselines, namely feature combination and soft voting. Feature combination [30] has a monolithic architecture that essentially requires every input sample to be paired observation. It concatenates feature vectors for both video and network to train a model. Soft voting [16] is another meta-learning approach, which can be regarded as a special case of stacking. It does not need any paired observation as it simply uses the average of the probability scores from the base models.

We not only test the device classification problem but also the per-device activity classifier and the uniform classifier. Due to the similarity among F_3, F_4, and F_5, we consider 4 classes for fridge and 15 classes for uniform activity classifiers. Because the vacuum activities are collected 7 months away from the other devices, we also report the results for a 12-class uniform classifier without vacuum. From Figure 4, video and network classifiers have different prediction power for different tasks. Video model has a higher F1 score for washer and vacuum, similar performance for 6-class fridge, and lower F1-score for the device, 4-class fridge, Alexa, Nestcam, and uniform classifiers, compared to network models. As expected, combining two data sources outperforms either of the single-modality classifiers. Within combination approaches, we observe that feature combination has an accuracy similar to the meta-learning approaches, but it also has the highest training overhead.

We next look into the differences between soft voting and stacking in greater detail.

5.2.3 Soft Voting vs. Stacking. Figure 4 shows that both stacking and soft-voting have statistical similarities in performance in most cases. Only for the Alexa activity classifier, stacking is significantly better than soft voting. The biggest advantage of soft voting is that it does not require paired observations, however, it suffers from some major drawbacks. (1) Soft voting highly depends on the capacity of base learners (single-modality models). As shown in Figure 5, the original model (Random Forest) for 15-class uniform classification is on the left. After we change the network model to a simple One vs. Rest Gaussian Process (OvRGP), the soft voting algorithm is

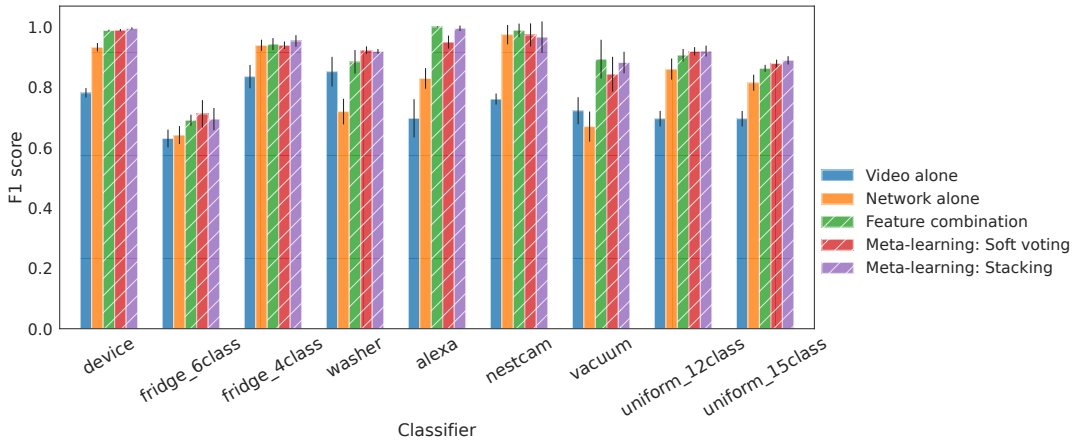


Fig. 4. F1 Score for different classifiers (vacuum included). For fridge classifiers, separate classifiers are trained with and without F_4 and F_5. For uniform classifiers, separate classifiers are trained with and without vacuum activities. The error bars are computed using 95% confidence interval.

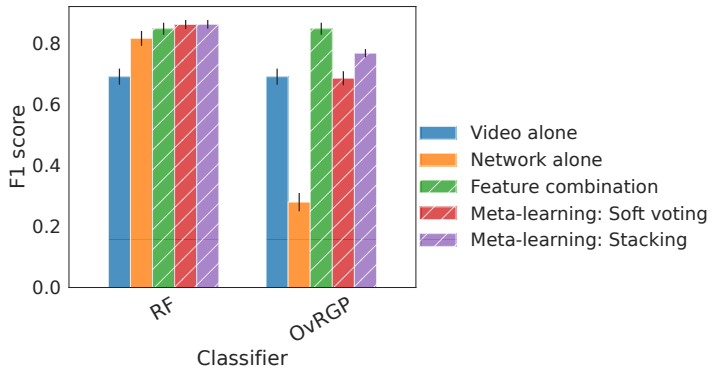
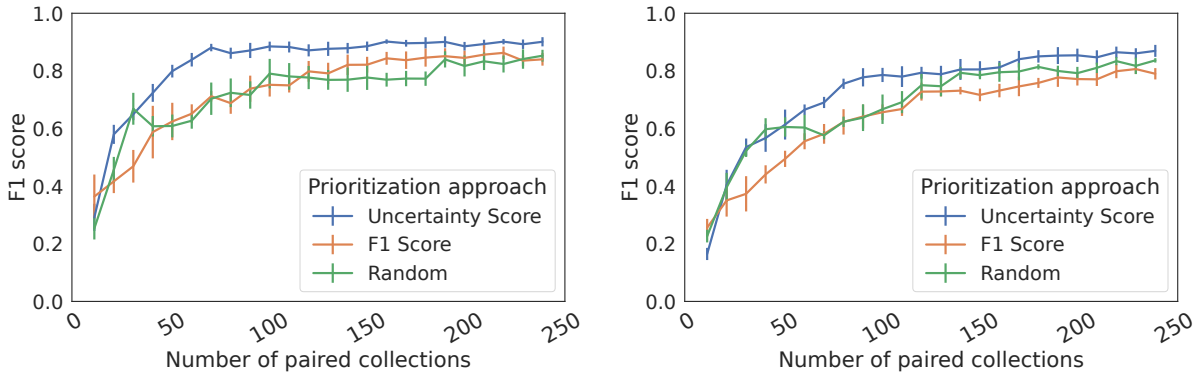


Fig. 5. Stacking vs Soft voting under two different network models. The video model is fixed to be One Vs Rest Logistic regression. Feature combination is using Random Forest.

instantly influenced, showing a great drop -17.62% in the accuracy. But the stacking classifier is not as affected (-9.46%). (2) Soft voting has lower adaptability to change in physical environments. Here, we assume that we train the base learners and test the algorithms in different environments. Active collection of paired examples from the new environment in stacking ensures the existing models are fine-tuned in a different environment. This is because the statistical characteristics of the newly collected sample are likely closer to the testing set. With active collections, this can be captured in the stacking meta-learner training process. However, a soft voter will make decisions based on the voting heuristics and predict biased results. As we discuss later in Section 5.4, when models trained in the lab are tested at home, the soft voting mechanism fails to remain effective, with F1-score dropping from 85% to 41%. The stacking classifier, however, shows a more consistent performance with only a 6% drop in accuracy. (3) Moreover, theoretically, stacking is a direct generalization of soft-voting. In order to achieve higher flexibility and effectiveness, we believe stacking is a better meta-learning approach for such tasks.



(a) 12-class classification.

(b) 15-class classification

Fig. 6. The F1 score (with 95% confidence interval as error bars) changes when the number of paired observations varies for 12-class and 15-class uniform classifiers.

5.3 Effectiveness of Active Collection Algorithm in AMIR

Now that we have demonstrated that "combination is an effective approach" and "stacking meta-learner does not require all paired observations as the feature combination, and is more robust than soft voting meta-learner" we seek to optimize the stacking effectiveness, *i.e.*, to evaluate the active paired observation collection algorithm we proposed in Section 3.3. More specifically, we analyze the amount of reductions and final accuracy on paired collections using uncertainty distribution prioritization compared to F1 and random prioritization (standard stacking).

The idea behind paired observation prioritization is to "cherry-pick" the most informative group of data for collection and training the stacking meta-learner. By this intentionally "biased" collection method, the saturation of F1 score is achieved earlier and may potentially end at a higher level. We use the same train-test split approach as a regular stacking method, and we use 5-fold cross-validation on the training set to derive the uncertainty and F1 class distributions. Then additional data is collected. The stacking classifier (Random Forest) is unchanged and is same as in random prioritization.

As shown in Figure. 6, we present two uniform classification tasks with different complexity levels. Although all three methods eventually achieve saturation of F1, the overall trend is that the prioritization by uncertainty-based class distribution is the fastest to saturate (70 and 170 samples, respectively for 12-class and 15-class problem) and has the highest F1 score (90.15% ± 0.93% and 86.95% ± 2.08%). In comparison, random prioritization with equal class distributions requires 120 (~2 hours of human labor) and 50 (~50 minutes of human labor) more samples for saturation. Meanwhile, uncertainty prioritization achieves higher final F1 scores than the other two methods by 3.30% to 4.88%. Also, from the comparison between 12-class and 15-class problems, the more complex the task, the more samples are needed for saturation.

In Table 6, the F1 score changes are illustrated for each per-device classifier. Uncertainty prioritization is the best approach for all devices except for the fridge. The uncertainty approach needs only half of the samples to achieve the optimal F1 score for the washer, Alexa, and Nestcam. Prioritization by F1 score is an intuitive baseline, but the effectiveness is not as good. The reason behind it is that over-sampling the accurate classes does not help with faster decision making on inaccurate classes. For example, for the Alexa classifier in Figure 2, the video model is accurate in distinguishing "Idle" (I) and the rest, while the network model is good at telling A_2

Table 6. The F1 score changes when the number of paired observations varies for the per-device classifier. Uncer. denotes prioritization by uncertainty scores for each class. The sample numbers of the earliest saturation and corresponding methods are shaded for each classifier. We also bold the best F1 scores across 3 prioritization methods for each number of samples.

#Sample	Fridge (4 classes)			Washer			Alexa			Nestcam		
	Uncer.	F1	Random	Uncer.	F1	Random	Uncer.	F1	Random	Uncer.	F1	Random
10	0.536	0.867	**0.899**	0.556	**0.602**	0.600	0.642	**0.791**	0.491	**0.956**	0.816	0.547
20	0.885	**0.939**	0.884	**0.921**	0.879	0.764	**0.903**	0.821	0.777	**0.979**	0.881	0.970
30	0.919	**0.941**	0.876	**0.921**	0.880	0.893	**0.971**	0.903	0.928	**0.979**	0.965	0.972
40	0.926	0.940	**0.942**	**0.921**	0.909	0.900	**0.995**	0.944	0.931	**0.979**	0.974	**0.979**
50	0.939	0.940	**0.944**	**0.917**	0.904	0.908	**1.000**	0.994	0.931	**0.979**	0.978	0.969

from A_3. This near-complimentary case makes the collection of A_1 tricky. The uncertainty approach prioritizes A_1's collection while using F1-score lowers the priority. Thus, it is faster to find the decision boundary using the former method.

Note that for each class, the uncertainty approach does require a relatively accurate prediction from at least one base learner. We do not observe the same effectiveness of uncertainty prioritization on the 6-class fridge classifier, partly because both individual classifiers are uncertain and inaccurate on F_3, F_4, and F_5. Prioritization based on uncertainty distribution biases the classifier towards classes that do not have reliable information, which breaks the decision boundary searching process. We will further explore the best approach that considers both uncertainty and accuracy in future work.

5.4 Transferability across Environments

We now test the transferability of AMIR across physical environments. We specifically analyze the transferability of different components of the AMIR framework including the base models, combination algorithms, and the active collection strategy. We use the data collected on the iRobot vacuum in the IoT lab and from a smart home for this analysis. Our results show that (1) network models are more robust to environmental changes compared to video; (2) stacking with some paired examples shows adaptive capacities and performs the best across environments; (3) the uncertainty-based prioritization of AMIR helps to derive a meta-learner when environments are different.

5.4.1 Model Transferability. We begin by calculating the F1 scores of the base and combination models under the two environments. As shown in Table 7, when the training and testing datasets are from the same environment, both the video and network classifiers achieve good performance above 70%. The stacking classifiers perform better because it combines two complementary modalities together with some paired examples to train meta-learners, achieving an F1-score of 89% in the lab and 98% at home.

We now consider classifier accuracy when the training data is collected from one environment and the test data is collected from the other. We notice a drastic performance drop (50% and 65% of F1 score decreases) in the video classifier. The reason is that video classifiers are sensitive to the physical environment including factors such as the home layout, lighting, different objects, and camera angle. Variations in these factors lead to a drop in model accuracy. On the contrary, network traffic traces are largely independent of the physical environment, and we only observe slight drops (6% and 5%) in a completely different environment.

Within the combination models, soft voting also reports a drop in accuracy. This is because it only averages the output probability vectors of two base learners, with no paired examples. So it is not able to adapt to the new environment, getting losses of 44% (trained in the lab, tested at home) and 38% (trained at home, tested in the lab) in the F1 score. In contrast, stacking is able to adapt to the new environment as it uses a few paired examples to

Table 7. The F1 score changes when the training and testing sets are different. Soft voting does not use paired samples, but Stacking (randomly prioritized) uses some paired samples in the testing environment.

Dataset		**F1 of Classifiers**			
Base Training	Testing	Video	Network	Voting	**Stacking**
In-lab	In-lab	0.74	0.74	0.85	**0.89**
	At-home	0.24	0.68	0.41	**0.83**
At-home	At-home	0.83	0.70	0.95	**0.98**
	In-lab	0.18	0.65	0.57	**0.90**

Table 8. The F1 score changes when the number of paired observations varies for the vacuum classifier across environments. In the parenthesis, the environment before the arrow provides a training set for base learners and weights for prioritization. And the environment after the arrow forms the training set of the meta-learner, as well as the test set.

#Sample	**Vacuum (Lab->Lab)**			**Vacuum (Lab->Home)**		
	Uncertainty	F1	Random	**Uncertainty**	F1	Random
10	0.687	**0.698**	0.555	**0.501**	0.494	0.444
20	0.847	**0.887**	0.795	**0.673**	0.662	0.524
30	0.863	**0.896**	0.851	0.739	0.755	**0.770**
40	0.876	**0.891**	0.861	**0.812**	0.804	0.786
50	**0.899**	0.891	0.890	**0.822**	0.798	0.813
120	**0.901**	0.874	0.896	**0.877**	0.834	0.831

train the meta-learners. It outperforms both the base learners as well as soft-voting reporting an F1-score of 83% and 90% in the lab and at home, respectively; a decrease of 5% and 8% compared to testing in the same setting.

5.4.2 Transferability of Active Data Collection. In Section 5.3, AMIR is able to effectively reduce the amount of samples being used to train a meta-learner, while improving the performance. Here we evaluate the transferability of the approach, *i.e.*, whether the weights for prioritization obtained from one environment are transferable to a new setting.

In Table 8, after training base learners from one setting, we obtain the weights of prioritization in the same environment and use them to train a meta-learner. In the *Lab->Lab* case, we derive the weights from the base learners trained in lab, and the meta-learner is also trained and tested in the lab. Compared to the random prioritization approach, the uncertainty-guided method is able to gain about 1% of more average F1 scores and saturates much faster, *i.e.*, with 40% fewer (from 70 to 50) paired samples.

We now consider the more practical and operationally valuable use case of our framework, where we use the prioritization weights derived from the base models trained in one environment to prioritize paired data collection in a new environment. Thus, we test whether these weights can still be transferred to improve meta-learning. As shown in the *Lab->Home* case in Table 8, we find that using the weights derived from the lab environment, the uncertainty-based prioritization is able to decrease the number of paired samples required for saturation of F1-score from 150 (random selection) to 120 pairs. It also reaches an F1 score of 87.7%, which is 4.6% higher than conducting random selections. Note that the number of paired samples needed to achieve saturation in this case (120) is significantly higher than that within the same environment (50). This indicates that training a meta-learner in a new environment is relatively more challenging than in the same setting.

6 DISCUSSION AND FUTURE WORK

6.1 Limitations

6.1.1 Applicable Activities. Our approach is limited to interactions with an Internet-connected entity (e.g., device, infrastructure, etc.), generating both physical signatures (video) and network footprints. For example, our augmentation framework applies to "opening and closing a smart fridge", but not "walking on the open ground". However, "walking under a motion-sensing camera" is within our scope because it captures changes from the video and triggers an upload on network traffic for motion analytics. We argue that the scope of these interactions is only going to increase in the future, as more and more devices are becoming connected to the Internet.

6.1.2 Guided vs. Natural Activities. One may argue that our data is collected through guided activities and does not cover the same degree of variation as natural activities in the real world. However, collecting data corresponding to natural activities is challenging as they are more spontaneous and sparse, and have significant privacy concerns and labeling efforts. Moreover, our data, although collected through guided activities, exhibits some degree of variation as it spans 7 users that collect data over multiple rounds and over different times of the day. The (deliberate) vagueness in the instructions also leads to some variations. For instance, the script to "take out an item from the fridge" does not specify the walking direction (left or right), the hand, the fridge door or rack (upper or bottom), and the specific item. Similar variations exist for other activities as described in Appendix B. Finally, we argue that the main focus of the paper (*i.e.*, AMIR) is independent of the labeling process and whether the activities are natural or instructed.

6.1.3 Limitations of Network Traffic. Our evaluation shows that for a network-assisted approach to be effective, the network signal should be fairly unique. For instance, it was challenging to distinguish between open-close, take-out, and put-back activities in the case of a smart refrigerator in Section 5.2. There can be other limitations that arise due to practical considerations including the vantage point location, firmware updates, network conditions, and device brands. For instance, device or firmware changes can lead to changes in the network traffic patterns, and thus may require re-training of the base model. Similarly, different network conditions can affect model accuracy. For example, in an extremely throughput-constrained environment, packet losses and re-transmissions may be high or a connection may terminate, which can in turn impact the model accuracy. In future work, we will analyze the robustness of network models to such changes.

6.1.4 Transferability. The design of AMIR assumes the same set of labels/activities across all environments and modalities. However, this assumption may not hold true when dealing with different devices and activities in different settings. For instance, the lab classifiers may include activities from a smart fridge, Nestcam, and Alexa, but a home environment may have a subset of devices and associated activities (e.g., a smart fridge and Alexa). While a meta-learner can be trained on new labels to transfer the knowledge of old models, the uncertainty weights from old labels cannot be trivially transferred. A possible solution is to use tree-based per-device classifiers that can be easily replaced in a plug-and-play fashion. These classifiers can then be retrained for specific devices and associated activities to derive the new uncertainty weights. In the example provided, only the Alexa and fridge models would need to be retrained at the lab for alignment. Besides, our evaluation of transferability is limited using data from two physical environments for one device only. We find that although the uncertainty weights can transfer between these two environments, the amount of paired demonstrations needed increases. Further, not until real collection is done do we know the optimal number of new demonstrations to saturate accuracy. For future work, we will investigate the transferability of AMIR in more environments on more devices, under the assumption that labels across environments might change. The latter calls for new algorithm designs (e.g., estimate label similarities for weight alignment) for uncertainty weights to be transferred.

6.2 Privacy Concerns

Activity recognition has associated privacy concerns, especially when using video data. Related work has proposed a privacy-preserving video activity recognition system including downgrading video quality [14] and edge computing for processing data locally [29]. All of our models have been tested using the Jetson Nano, indicating the framework is lightweight enough and amenable to edge computing using an inexpensive Single Board Computer like Jetson Nano or Raspberry Pi.

The network data also possess additional privacy concerns with the possibility of an in-network adversary or consumer Internet Service Provider using the network models to infer activity. A possible solution is to use techniques like data padding [6]. However, the location of data obfuscation is critical. For instance, if data padding happens on the device itself, the data no longer remains usable for our framework. A safer approach is to perform padding at an aggregation point upstream (e.g., router) for improved user privacy. This requires more in-depth discussion and is out of the scope of this paper.

6.3 Alternate Multimodal Learning Techniques

In this paper, we explore two classic ensemble learning methods, namely soft voting and stacking. However, there are other methods useful in this context including Bayesian model averaging and variants of stacking. For instance, we could train the stacking-based meta-learner with base features in addition to model output. Similarly, there could be other learning frameworks that require no or even fewer paired observations. A simple combination framework could be to use either a video model or a network model based on prior knowledge about the accuracy of the model under different activities. Similarly, there can be other active learning techniques that can be used to train a meta-learner with as few samples as possible. We tried a few other metrics including a combination of uncertainty and F1 score but that did not improve accuracy. We plan to explore these other techniques in future work, looking at meta-learning approaches in addition to stacking (e.g., hierarchical multi-modal classification)

6.4 Other Modalities

In addition to network and video data there are other data sources present in an in-home setting such as smartphones, wearable sensors, and audio. Our results, so far, indicate that combining data at the model level can yield similar accuracy to the feature-level combination. This is likely due to the complementary nature of video and network data sources with one containing no information about the other. This is an important insight and can be potentially used for other data sources as well. For instance, audio, motion sensors, and wireless signals are likely to be complementary to both video and network. If true (needs validating), it is easy to extend our framework. One can simply train a model using sensor data alone and then combine the model-level outputs of more data sources using a meta-learner.

7 CONCLUSION

This paper advocates for the synthesis of video and network data for improved accurate in-home activity recognition. Our evaluation shows that using both network and video data results in higher accuracy compared to using video and network data alone. Furthermore, our active learning approach significantly reduces the paired network and video observations needed for training the meta-learner. Our results indicate that there is a huge potential in synthesizing complementary data sources to improve elder care, safety, and automation for connected homes. In our future work, we plan to augment additional data modalities (e.g., audio, motion sensors, wireless signals.) with network and video data. AMIR can also be extended to other connected environments like smart buildings and smart factories, exploring meta-learning approaches in addition to stacking (e.g., hierarchical multi-modal classification). Finally, we plan to explore privacy and security aspects of our techniques, identifying and mitigating adversarial threats and privacy violations.

ACKNOWLEDGMENTS

We thank the anonymous associate editors and reviewers for their constructive feedback and suggestions. This research is partly supported by grants from NSF CNS-2124393 and CNS-2126327, as well as a generous grant from Intel and SRI.

REFERENCES

[1] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. 2020. Peek-a-boo: I see your smart home activities, even encrypted!. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 207–218.

[2] Fadel Adib, Zachary Kabelac, and Dina Katabi. 2015. Multi-Person Localization via RF Body Reflections. In *12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA, May 4-6, 2015*. USENIX Association, 279–292. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/adib

[3] Fadel Adib, Zachary Kabelac, Dina Katabi, and Robert C. Miller. 2014. 3D Tracking via Body Radio Reflections. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014*, Ratul Mahajan and Ion Stoica (Eds.). USENIX Association, 317–329. https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/adib

[4] Fadel Adib, Hongzi Mao, Zachary Kabelac, Dina Katabi, and Robert C. Miller. 2015. Smart Homes that Monitor Breathing and Heart Rate. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, Bo Begole, Jinwoo Kim, Kori Inkpen, and Woontack Woo (Eds.). ACM, 837–846. https://doi.org/10.1145/2702123.2702200

[5] Ahmet Aksoy and Mehmet Hadi Gunes. 2019. Automated iot device identification using network traffic. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[6] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. *arXiv preprint arXiv:1708.05044* (2017).

[7] Noah J. Apthorpe, Dillon Reisman, and Nick Feamster. 2017. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. *CoRR* abs/1705.06805 (2017). arXiv:1705.06805 http://arxiv.org/abs/1705.06805

[8] Noah J. Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. *CoRR* abs/1708.05044 (2017). arXiv:1708.05044 http://arxiv.org/abs/1708.05044

[9] Pradeep K Atrey, M Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S Kankanhalli. 2010. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems* 16, 6 (2010), 345–379.

[10] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence* 41, 2 (2018), 423–443.

[11] Onur Barut, Yan Luo, Tong Zhang, Weigang Li, and Peilong Li. 2020. NetML: a challenge for network traffic analytics. *arXiv preprint arXiv:2004.13006* (2020).

[12] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. 2014. NILMTK: An open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th international conference on Future energy systems*. 265–276.

[13] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. 2021. Is space-time attention all you need for video understanding. *arXiv preprint arXiv:2102.05095* 2, 3 (2021), 4.

[14] Daniel J Butler, Justin Huang, Franziska Roesner, and Maya Cakmak. 2015. The privacy-utility tradeoff for remotely teleoperated robots. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*. 27–34.

[15] Bradford Campbell and Prabal Dutta. 2014. Gemini: A non-invasive, energy-harvesting true power meter. In *2014 IEEE Real-Time Systems Symposium*. IEEE, 324–333.

[16] Jingjing Cao, Sam Kwong, Ran Wang, Xiaodong Li, Ke Li, and Xiangfei Kong. 2015. Class-specific soft voting based multiple extreme learning machines ensemble. *Neurocomputing* 149 (2015), 275–284.

[17] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[18] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *CVPR*.

[19] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. 2018. A Short Note about Kinetics-600. *CoRR* abs/1808.01340 (2018). arXiv:1808.01340 http://arxiv.org/abs/1808.01340

[20] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. 2019. A Short Note on the Kinetics-700 Human Action Dataset. *CoRR* abs/1907.06987 (2019). arXiv:1907.06987 http://arxiv.org/abs/1907.06987

[21] Liming Chen, Jesse Hoey, Chris D. Nugent, Diane J. Cook, and Zhiwen Yu. 2012. Sensor-Based Activity Recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 6 (2012), 790–808. https://doi.org/10.1109/TSMCC.2012.2198883

[22] MMAction2 Contributors. 2020. OpenMMLab's Next Generation Video Understanding Toolbox and Benchmark. https://github.com/open-mmlab/mmaction2.

[23] Fernando De la Torre, Jessica Hodgins, Adam Bargteil, Xavier Martin, Justin Macey, Alex Collado, and Pep Beltran. 2009. Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC) Database. (2009).

[24] Samuel DeBruin, Branden Ghena, Ye-Sheng Kuo, and Prabal Dutta. 2015. Powerblade: A low-profile, true-power, plug-through energy meter. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 17–29.

[25] Haodong Duan, Yue Zhao, Yuanjun Xiong, Wentao Liu, and Dahua Lin. 2020. Omni-sourced webly-supervised learning for video recognition. In *European Conference on Computer Vision*. Springer, 670–688.

[26] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. 2015. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 961–970.

[27] Christoph Feichtenhofer. 2020. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 203–213.

[28] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. SlowFast Networks for Video Recognition. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 6201–6210. https://doi.org/10.1109/ICCV.2019.00630

[29] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. 2015. Edge-centric computing: Vision and challenges. , 37–42 pages.

[30] Peter Gehler and Sebastian Nowozin. 2009. On feature combination for multiclass object classification. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 221–228.

[31] Alexander Grushin, Derek Monner, James A. Reggia, and Ajay Mishra. 2013. Robust human action recognition via long short-term memory. In *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*. IEEE, 1–8. https://doi.org/10.1109/IJCNN.2013.6706797

[32] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. 2021. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15262–15271.

[33] Jordan Holland, Paul Schmitt, Nick Feamster, and Prateek Mittal. 2021. New directions in automated traffic analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3366–3383.

[34] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* (2017).

[35] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. 2017. The Kinetics Human Action Video Dataset. *CoRR* abs/1705.06950 (2017). arXiv:1705.06950 http://arxiv.org/abs/1705.06950

[36] Gierad Laput, Yang Zhang, and Chris Harrison. 2017. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3986–3999.

[37] Ang Li, Meghana Thotakuri, David A. Ross, João Carreira, Alexander Vostrikov, and Andrew Zisserman. 2020. The AVA-Kinetics Localized Human Actions Video Dataset. *CoRR* abs/2005.00214 (2020). arXiv:2005.00214 https://arxiv.org/abs/2005.00214

[38] Yunpeng Li, Dominik Roblek, and Marco Tagliasacchi. 2019. From Here to There: Video Inbetweening Using Direct 3D Convolutions. *CoRR* abs/1905.10240 (2019). arXiv:1905.10240 http://arxiv.org/abs/1905.10240

[39] Dawei Liang and Edison Thomaz. 2019. Audio-based activities of daily living (adl) recognition with large-scale acoustic embeddings from online videos. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 1–18.

[40] Ye Liu, Liqiang Nie, Li Liu, and David S. Rosenblum. 2016. From action to activity: Sensor-based activity recognition. *Neurocomputing* 181 (2016), 108–115. https://doi.org/10.1016/j.neucom.2015.08.096 Big Data Driven Intelligent Transportation Systems.

[41] Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. 2021. Tam: Temporal adaptive module for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 13708–13718.

[42] Ching-Hu Lu and Li-Chen Fu. 2009. Robust Location-Aware Activity Recognition Using Wireless Sensor Network in an Attentive Home. *IEEE Trans Autom. Sci. Eng.* 6, 4 (2009), 598–609. https://doi.org/10.1109/TASE.2009.2021981

[43] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. 2017. ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*. 506–509.

[44] Robert Munro Monarch. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster.

[45] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 2019. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

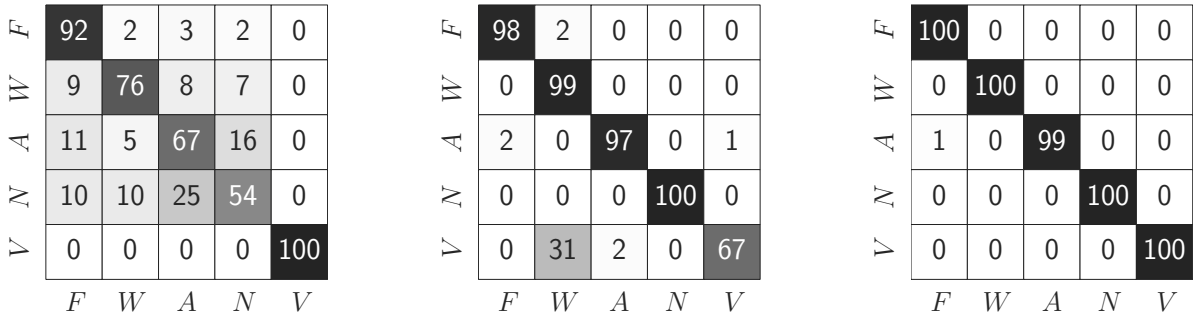
[46] Caetano Mazzoni Ranieri, Scott MacLeod, Mauro Dragone, Patrícia Amâncio Vargas, and Roseli Aparecida Francelin Romero. 2021. Activity Recognition for Ambient Assisted Living with Videos, Inertial Units and Ambient Sensors. *Sensors* 21, 3 (2021), 768. https://doi.org/10.3390/s21030768

[47] Andreas Reinhardt, Paul Baumann, Daniel Burgstahler, Matthias Hollick, Hristo Chonov, Marc Werner, and Ralf Steinmetz. 2012. On the accuracy of appliance identification based on distributed load metering data. In *2012 Sustainable Internet and ICT for Sustainability (SustainIT)*. IEEE, 1–9.
[48] Burr Settles. 2009. Active learning literature survey. (2009).
[49] Mustafizur R. Shahid, Gregory Blanc, Zonghua Zhang, and Hervé Debar. 2018. IoT Devices Recognition Through Network Traffic Analysis. In *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*, Naoki Abe, Huan Liu, Calton Pu, Xiaohua Hu, Nesreen K. Ahmed, Mu Qiao, Yang Song, Donald Kossmann, Bing Liu, Kisung Lee, Jiliang Tang, Jingrui He, and Jeffrey S. Saltz (Eds.). IEEE, 5187–5192. https://doi.org/10.1109/BigData.2018.8622243
[50] Shankar T Shivappa, Mohan Manubhai Trivedi, and Bhaskar D Rao. 2010. Audiovisual information fusion in human–computer interfaces and intelligent environments: A survey. *Proc. IEEE* 98, 10 (2010), 1692–1715.
[51] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In *CVPR*.
[52] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. 2020. A Short Note on the Kinetics-700-2020 Human Action Dataset. *CoRR* abs/2010.10864 (2020). arXiv:2010.10864 https://arxiv.org/abs/2010.10864
[53] Ekaterina H. Spriggs, Fernando De la Torre, and Martial Hebert. 2009. Temporal segmentation and activity classification from first-person sensing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2009, Miami, FL, USA, 20-25 June, 2009*. IEEE Computer Society, 17–24. https://doi.org/10.1109/CVPRW.2009.5204354
[54] Sebastian Stein and Stephen J. McKenna. 2013. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *The 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '13, Zurich, Switzerland, September 8-12, 2013*, Friedemann Mattern, Silvia Santini, John F. Canny, Marc Langheinrich, and Jun Rekimoto (Eds.). ACM, 729–738. https://doi.org/10.1145/2493432.2493482
[55] Johannes Andreas Stork, Luciano Spinello, Jens Silva, and Kai Oliver Arras. 2012. Audio-based human activity recognition using Non-Markovian Ensemble Voting. In *The 21st IEEE International Symposium on Robot and Human Interactive Communication, IEEE RO-MAN 2012, Paris, France, September 9-13, 2012*. IEEE, 509–514. https://doi.org/10.1109/ROMAN.2012.6343802
[56] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson. 2004. Activity recognition in the home using simple and ubiquitous sensors. In *International conference on pervasive computing*. Springer, 158–175.
[57] Moritz Tenorth, Jan Bandouch, and Michael Beetz. 2009. The TUM Kitchen Data Set of everyday manipulation activities for motion tracking and action recognition. In *12th IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2009, Kyoto, Japan, September 27 - October 4, 2009*. IEEE Computer Society, 1089–1096. https://doi.org/10.1109/ICCVW.2009.5457583
[58] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2019. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 119 (2019), 3–11. https://doi.org/10.1016/j.patrec.2018.02.010 Deep Learning for Pattern Recognition.
[59] Shuangquan Wang and Gang Zhou. 2015. A review on radio based activity recognition. *Digital Communications and Networks* 1, 1 (2015), 20–29. https://doi.org/10.1016/j.dcan.2015.02.006
[60] Wei Wang, Alex X. Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. 2017. Device-Free Human Activity Recognition Using Commercial WiFi Devices. *IEEE J. Sel. Areas Commun.* 35, 5 (2017), 1118–1131. https://doi.org/10.1109/JSAC.2017.2679658
[61] Ying Wang, Kaiqi Huang, and Tieniu Tan. 2007. Human activity recognition based on r transform. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.
[62] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *CVPR*.
[63] Kun Yang, Samory Kpotufe, and Nick Feamster. 2020. Feature Extraction for Novelty Detection in Network Traffic. *arXiv preprint arXiv:2006.16993* (2020).
[64] Shibo Zhang, Yaxuan Li, Shen Zhang, Farzad Shahabi, Stephen Xia, Yu Deng, and Nabil Alshurafa. 2022. Deep Learning in Human Activity Recognition with Wearable Sensors: A Review on Advances. *Sensors* 22, 4 (2022), 1476.
[65] Mingmin Zhao, Fadel Adib, and Dina Katabi. 2016. Emotion recognition using wireless signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, MobiCom 2016, New York City, NY, USA, October 3-7, 2016*, Yingying Chen, Marco Gruteser, Y. Charlie Hu, and Karthik Sundaresan (Eds.). ACM, 95–108. https://doi.org/10.1145/2973750.2973762
[66] Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning* 3, 1 (2009), 1–130.

A ADDITIONAL EXPERIMENTS

Confusion Matrices for device classifier. The device-IP mapping relationship can either be obtained from user inputs or automatic collection of an in-home router, or we can use a particular device classifier based on our collection. We use a 5-fold cross validation on the dataset to obtain the confusion matrices. In Figure 7, confusion matrix of the video classifier shows worse prediction power and is bad at classifying devices with proximate

interactions (*i.e.*, interactions that happened at surrounding with no physical touch or mechanical movements). The network classifier, however, demonstrates strong inference capacity on all devices. By combining them together using a meta-learning stacking classifier (Random Forest), 2% of accuracy increase is observed on *F* (Samsung smart fridge).



(a) Video classifier. (b) Network classifier. (c) Stacking classifier.

Fig. 7. Confusion matrices of different classifiers for 5-device classification. Results shown here means accuracy (%). Results for each device might not add up to 100% because of rounding on the number. Use Table 5 as key for names and devices.

Choice of features. We chose the latter (2nd to last) layer outputs of the SlowFast model because they in general convey the learned derivation of the inputs. In order to test out the impact of different layers of outputs, we also show the results in Sec 6.2 using the 3rd, 4th, and 5th to last layer and they are similarly good 67 to 73% F1 scores for the 15-class video classifiers.

Table 9. The performance of 15-class video classifiers (OneVsRest Logistic regression) when we select different layer outputs from SlowFast models as features.

Layer name	Pooling	Linear	Dropout	AvgPool3d
Layer to last	2nd	3rd	4th	5th
Macro F1	0.67	0.68	0.73	0.73
Weighted F1	0.82	0.82	0.85	0.85

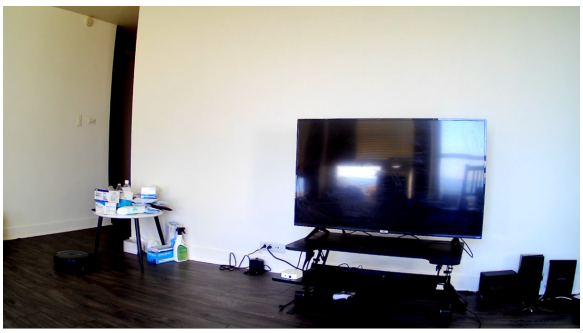
B DATA COLLECTION

Activity scripts. In Table 10, we list the comprehensive scripts of activities and all devices. In this table, we verbosely note down the degrees of freedom for these tasks, which shows the variations of the dataset and the spontaneous choices made by the users.

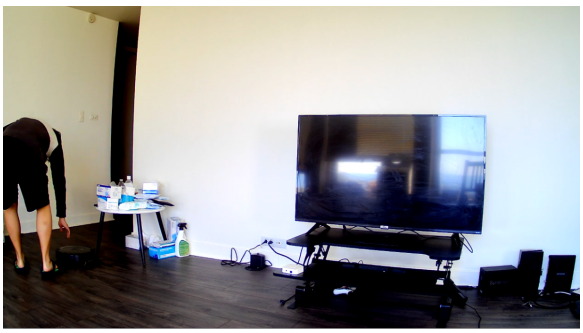
Real-world smart home. Figure 8a shows our home environment for data collection. Apart from our target device (robot vacuum), there are a number of irrelevant objects such as the TV and video game consoles in a real-world environment. Figure 8b shows a sample activity where the user clicks a button on the robot vacuum.

Table 10. Activity list and corresponding scripts and degrees of freedom.

Device	Activity	Script	Degree of freedom
All	idle	Do nothing. Make sure no user is in the camera.	light condition, hour of day
Fridge	check on fridge	Walk to the fridge. Do nothing for 2-5 seconds. Walk away.	Enter and leave direction, walking speed, waiting time
	screen interaction	Walk to the fridge. Tap the screen on the right. Interact with it for 5 seconds (e.g., browse recipe, watch video, say hey bixby <some command>). Walk away.	Enter and leave direction, walking speed, interaction time, interaction type, preferred hand
	open close fridge	Walk to the fridge. Open the door. Wait for 2-5 seconds. Close the door. Walk away.	Enter and leave direction, walking speed, waiting time, preferred hand(s), fridge door(s)
	take out item	Walk to the fridge. Open the door. Take an item out. Close the door. Walk away with the item.	Enter and leave direction, walking speed, waiting time, preferred hand(s), fridge door(s), item
	put back item	Pick an item. Walk to the fridge. Open the door. Put the item in the refrigerator. Close the door. Walk away.	Enter and leave direction, walking speed, waiting time, preferred hand(s), fridge door(s), item
Washer	check on washer	Walk to the fridge. Do nothing for 2-5 seconds. Walk away.	Enter and leave direction, walking speed, waiting time
	start washing	Walk to the washer. Turn on washer, put in clothes, close lid, start washing, walk away from washer.	Enter and leave direction, walking speed, waiting time, preferred hand(s), cloth(es)
	end washing	Walk to the washer. End washing, open lid, get clothes out, turn washer off. Walk away.	Enter and leave direction, walking speed, waiting time, preferred hand(s), cloth(es)
Alexa	check on Alexa	Walk to the Echo. Do nothing for 2-5 seconds. Walk away.	Enter and leave direction, walking speed, waiting time
	ask weather	Walk to the Echo. Say 'Alexa, what is the weather'. Walk away.	Enter and leave direction, walking speed, waiting time, voice tone, weather response
	play music	Walk to the Echo. Say 'Alexa, play some music'. Let it play for a while and say 'Alexa, stop.' Walk away.	Enter and leave direction, walking speed, waiting time, voice tone, music
Nestcam	ask time	Walk to the Nest Camera. Say 'OK Google, what's the time?'. Walk away.	Enter and leave direction, walking speed, waiting time, voice tone, time response
	trigger motion detection	Walk towards the Nest Camera. Do nothing for 2-5 seconds. Walk out of the view.	Enter and leave direction, walking speed, waiting time
Vacuum	check on vacuum	Walk to the Vacuum. Check its status. Wait for 2-5 seconds. Walk away.	Enter and leave direction, walking speed, waiting time
	remote control	Begin collection. Start vacuuming on App. Wait for 2-5 seconds. Pause vacuuming on App. Stop collection.	Enter and leave direction, walking speed, waiting time, inside/outside LAN
	click button	Walk to the Vacuum. Click a button on robot vacuum. Walk away.	Enter and leave direction, walking speed, waiting time, preferred hand(s), button



(a) The home environment



(b) Sample activity: click a button on the robot vacuum.

Fig. 8. Sample images for the home environment and the data collection.