Enabling System Support for General-Purpose Sensing in Smart Environments

Sudershan Boovaraghavan

CMU-S3D-24-106

September 2024

Software and Societal Systems Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Yuvraj Agarwal, Chair
Chris Harrison
Mayank Goel
Anind K. Dey (University of Washington)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Societal Computing.

The research reported here was supported, in whole or in part, by a National Science Foundation grant under SaTC-1801472, the CMU's CyLab Security and Privacy Institute, and a gift by JP Morgan Chase. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government, or any other entity.

Keywords: Internet of Things, Sensing and Sensor Technologies, Distributed Sensor Network, Privacy, Machine Learning, Context Recognition, Real–World Deployment

To my dearest family and friends

Abstract

The Internet-of-Things (IoT) sensing systems have the potential to revolutionize our living environments, yet their transformative potential remains largely unrealized. Despite the rapid proliferation of IoT devices and their immense potential for a range of applications like building maintenance and healthcare monitoring, their integration into real-world environments faces significant hurdles due to practical deployment challenges and escalating privacy concerns.

Current IoT sensing systems are typically built with monolithic, purpose-specific architectures that focus on a limited range of sensing capabilities designed for specific applications. This results in isolated, vendor-controlled solutions with limited features to support diverse application requirements for machine learning (ML), scale, and reliability. As a result, IoT ecosystems become fragmented, which hinders both widespread adoption and long-term viability.

To address these limitations of current IoT systems, this thesis proposes a shift towards general-purpose sensing systems that support current and future applications, adapt to evolving stakeholder needs, and provide robust privacy safeguards. This thesis introduces several novel system design approaches to achieve this vision. Starting with **Mites**, a scalable, general-purpose sensing platform that delivers fine-grained environmental data and establishes the foundational architecture for extensible and adaptable IoT deployments across various application scenarios. Building on this, **MLIoT** is presented as an end-to-end general-purpose machine learning system designed to transform raw sensor data into high-level inferences, supporting the entire ML lifecycle for IoT applications. To further enhance the interpretability of these inferences, **TAO**, a context recognition framework, is developed to detect semantically meaningful contexts from the inference, improving understanding and usability agnostic to the underlying ML inference pipelines. Complementing these advancements, **Kirigami** showcases a general-purpose edge audio speech filter that removes human speech segments while preserving other sounds, thereby maintaining high accuracy for non-speech inferences and balancing privacy with utility. The thesis demonstrates how comprehensive system support for general-purpose sensing facilitates various applications and meets the needs of diverse stakeholders through the real-world deployment of more than 300 multimodal sensor devices in a fully occupied, five-story university building at Carnegie Mellon University (CMU). Through these innovative system design approaches, this thesis advocates a transformative shift towards scalable, privacy-preserving, and general-purpose IoT sensing systems, unlocking the full potential of smart environments.

Acknowledgments

This dissertation was only possible because of my advisors, mentors, collaborators, friends, and family. They have enabled my work, made my Ph.D. path extremely rewarding, and helped develop me into a better person.

First, I would like to thank my advisor, Yuvraj Agarwal, for his invaluable mentorship and support over the years. Your mentorship has been invaluable, from my beginnings as a student intern in your lab to completing my Ph.D. He has been a crucial part of my growth and development. Thank you for being so patient in discussing all the crazy ideas with me and guiding me to discover the final direction outlined in this dissertation. Your guidance, support, and invaluable insights have been instrumental in the success of this work. Your dedication and expertise have not only helped shape this research but have also inspired me to strive for excellence. Thank you for your unwavering support and mentorship throughout this journey. This work would not have been possible without you.

I am also deeply grateful to my thesis committee members, Chris Harrison, Mayank Goel, and Anind Dey, for their insightful feedback and continuous support. Chris Harrison has been a great source of inspiration; his constructive feedback and encouragement have been crucial in refining this research. I have learned a great deal from him about presenting my work, thinking creatively, and paying attention to detail. Dr. Goel's critical insights and thoughtful suggestions have significantly improved the quality of the Kirigami project, pushing me to frame it more cohesively and impactfully. Dr. Dey's extensive knowledge and guidance have provided a strong foundation for this research. His influence on my approach to research and study design has been incredibly helpful. Thank you for your time, effort, and unwavering support throughout this process. Your collective wisdom and encouragement have been a source of great inspiration.

I have been fortunate enough to have been surrounded by the most amazing mentors. I want to extend my special thanks to my mentors, Ram Konduru and Dr. Raj Reddy, who gave me the incredible opportunity to come to CMU as an intern. Their belief in my potential and the chance they provided me have been pivotal in my academic and professional journey. I am deeply grateful for their mentorship, which opened doors that have shaped my career. I sincerely thank Jim Herbsleb, Jason Hong, Lorrie Cranor, and Dave Eckhardt, who have been incredibly supportive throughout my academic career. Their valuable feedback and suggestions have significantly contributed to making the system better and more usable.

I also want to extend a special thank you to the administrative staff, particularly Tom Pope, Cole Jester, Stefan Hadricky, Erin Murray, Jennifer Cooper, Lisa Dougherty, Helen Higgins, Connie Herold, Alisha Roudebush, Aaron Caldwell, and Emanuel Bowes. Their behind-the-scenes efforts and assistance with the many details and logistics have been indispensable. Their efficiency and support have made this process so much smoother, and I sincerely appreciate all that you've done.

I also want to extend my heartfelt thanks to my incredible collaborators and labmates: Tarun Karuturi, Gokul Krishna, Chen Chen, Dohyun Kim, Han Zhang,

Anurag Maravi, Shreyas Nagare, Ben Weinshel, Abhijit Hota, Yang Zhang, Gierad Laput, Karan Ahuja, Prasoon Patidar, Haozhe Zhou, Robert Xiao, Jason Koh, Rushil Khurana, Zach Meng, Dezhi Hong, Bharathan Balaji, Eiji Hayashi, Matus Tomlein, Xiaolin Charlene Zang, Matilda Ferguson, Prahaladha Mallela, Mike Czapik, Haojian Jin, Abhijith Ragav, Saksham Chitkara, Yuchen Liang, Lucas Blanchard, Xingchen Wang, Priyanshi Jain, Suryaa Kasthurisamy Selvaraj, Bingchen Li, Riku Arawaka, Vimal Mollyn, Daewha Kim, Catherine Tianhong Yu. Your support, camaraderie, and shared enthusiasm have made this journey not only productive but also enjoyable. I am deeply grateful for the collaborations and the many inspiring conversations we've had. Having you all as part of this journey has truly enriched my experience, and I am thankful for each of you.

I also want to express my deepest thanks to my friends, who have been an incredible support system throughout this journey. I want to thank my incredible Pittsburgh family: Prof. Adithya Pediredla, Akhila Kolapalli, Arya Pediredla, Dr. Alankar Kotwal, Aparajita Sahoo, Dr. Jayanth Krishna Mogali, Mukesh Thangadurai, Geetha Barani, Ashok Kishore, Bhavya Jha, Dr. Ezgi Bakirci, Dr. Mallesh Dasari, Janani Venkatraman, Kalyani Tembhe. Your encouragement, company especially during COVID, has been invaluable. I will always cherish our board game nights and the insightful conversations we shared. Your presence and support have made a significant difference, and I'm truly grateful for having you all by my side.

I want to express my deepest gratitude to my family. Your love, patience, and unwavering support have been my bedrock throughout this journey. Your belief in me, your sacrifices, and your constant encouragement have been a source of strength and motivation. I am truly thankful for everything you have done and for always being there for me.

I want to give a special thanks to Neeha Dev Arun. Your love, understanding, and support have been a pillar of strength throughout this entire journey. Your patience, encouragement, and belief in me have been a constant source of motivation. Thank you for being there for me and for sharing this journey with me.

Contents

1 Introduction **1**
- 1.1 Overview of this Thesis . . . 2
- 1.2 Thesis Outline . . . 5

2 Design and Deployment of a General-Purpose Sensing Infrastructure **6**
- 2.1 Challenges . . . 7
  - 2.1.1 Diverse Application Requirements . . . 7
  - 2.1.2 Diverse Stakeholder Requirements . . . 8
  - 2.1.3 Lack of System Primitives to Support Long-Term Deployments . . . 8
- 2.2 Background . . . 9
  - 2.2.1 IoT Sensors . . . 9
  - 2.2.2 IoT Cloud Platforms . . . 9
  - 2.2.3 Combined Systems (IoT Sensors + IoT Cloud Platforms) . . . 9
  - 2.2.4 Design Goals . . . 10
- 2.3 Mites System Architecture . . . 11
  - 2.3.1 Mites Sensor Board . . . 11
  - 2.3.2 Mites Firmware . . . 13
  - 2.3.3 Mites Software Backend: Architecture and Design . . . 14
- 2.4 Evaluation . . . 18
  - 2.4.1 Experimental Setup . . . 19
  - 2.4.2 System Microbenchmarks . . . 19
  - 2.4.3 Overall System Evaluation . . . 21
- 2.5 Conclusion . . . 22

3 End-to-End General Purpose Machine Learning System for Smart Buildings **23**
- 3.1 Challenges . . . 24
  - 3.1.1 IoT Application Workloads . . . 24
  - 3.1.2 Challenges for ML systems in IoT settings . . . 25
- 3.2 Background . . . 25
  - 3.2.1 Large-scale ML Serving Systems . . . 25
  - 3.2.2 Large-scale ML Training Systems . . . 26
  - 3.2.3 ML Training/Serving Hybrid Systems . . . 26
- 3.3 System Design and Architecture . . . 27
  - 3.3.1 Device Selection Layer (DSL) . . . 28

3.3.2 Training-Serving Layer (TSL) . . . 30
3.4 Evaluation . . . 32
3.4.1 Experimental Setup . . . 32
3.4.2 System Adaptation and Scaling . . . 33
3.5 Conclusion . . . 34

4 Raising the Abstraction for Activity Inferences using Context Sensing Framework 35
4.1 Challenges . . . 36
4.1.1 Healthcare-based Applications . . . 36
4.1.2 Smart Building Applications . . . 37
4.2 Background . . . 37
4.3 TAO: System Architecture . . . 39
4.3.1 Overview . . . 39
4.3.2 Ontological Pipeline . . . 40
4.3.3 Temporal Clustering Pipeline . . . 42
4.3.4 Putting it All Together: The TAO System . . . 47
4.4 Evaluation . . . 47
4.4.1 Evaluation Setup . . . 48
4.4.2 Evaluation of TAO's Ontological, Temporal and Hybrid Pipelines . . . 49
4.4.3 Evaluation of TAO's Performance . . . 51
4.5 Conclusion . . . 52

5 General-Purpose Edge Audio Filter for Privacy-Preserving Activity Recognition 53
5.1 Challenges . . . 54
5.1.1 Microphone for Activity Recognition . . . 54
5.1.2 Phonemes in Audio . . . 55
5.1.3 Deep-learning-based Automatic Speech Recognition Models . . . 56
5.1.4 Threat Model . . . 57
5.2 Background . . . 58
5.2.1 Audio Privacy Filters for Activity Recognition . . . 58
5.3 Feasibility of Inferring Speech From Filtered Audio . . . 59
5.3.1 Fine-Tuning Phoneme-based Speech Inference Models . . . 60
5.3.2 Fine-Tuning Word-based Speech Inference Models . . . 61
5.3.3 Need for PER and WER contextualization . . . 62
5.4 Kirigami: Lightweight Speech Filter . . . 63
5.4.1 Machine Learning-based Kirigami Speech Filter . . . 63
5.4.2 Configuring Privacy vs. Utility Tradeoffs . . . 64
5.4.3 Kirigami Speech Filter on the Edge . . . 66
5.4.4 Adapting the Kirigami Speech Filter for Real-World Environments . . . 66
5.5 Evaluation . . . 68
5.5.1 Evaluation Setup . . . 68
5.5.2 RQ1: Feasibility of Speech Inference from Prior Filtering Approaches . . . 70
5.5.3 RQ2 : Performance of Kirigami filters . . . 72
5.5.4 PER v.s WER Contextualization . . . 72

5.5.5 Comparison of Kirigami and Prior Speech Filtering Approaches 74
5.5.6 RQ3: Evaluation of Kirigami filter in the Real World 76
5.6 Discussion and Limitations 80
5.7 Conclusion 81

6 Real-World Applications Enabled by our General-Purpose Sensing System 82
6.1 Management and Maintenance 83
6.1.1 Spatial Environment Monitoring Application 83
6.1.2 Device Health Monitoring Application 83
6.2 Occupancy Modeling 85
6.2.1 Availability of Conference Rooms 85
6.2.2 Find a Quiet Space 86
6.3 Human Activity Modeling Applications 87
6.4 Extensible Design for Rapid Prototyping Applications 87

7 Conclusions and Future Directions 88
7.1 Lessons Learnt 89
7.1.1 Iterative Hardware Design: Don't Reinvent the Wheel, but You May Have to Build Your Own 90
7.1.2 Known Unknowns and Unknown Unknowns for Sensor Deployments . . 91
7.1.3 Real-world Conditions are Chaotic and Unpredictable 92
7.1.4 Deployability is the Key to Reducing Costs 92
7.1.5 Community Sense of Privacy is the Key to Building Trust 93
7.2 Future Directions 93
7.2.1 Improving Support for Privacy 93
7.2.2 Real-world Activity Recogntion 94
7.2.3 Synthetic Data Generation 95
7.2.4 Robust Edge Computing 96
7.2.5 App Store for General-Purpose Sensing Systems 96

Bibliography 97

List of Figures

1.1 Overview of various components of an IoT system and how my work enables systems support for general-purpose sensing at different levels of the stack. This thesis aims to enable system support for general-purpose sensing and privacy at various components of an IoT system that can enable diverse applications and stakeholder requirements, ultimately fostering scalable long-term real-world deployments. 3

2.1 Overview of the Mites system and deployment. Each room has a Mites device on the wall and in the ceiling, with larger rooms and shared areas having multiple devices in the ceiling. Each device sends an encrypted stream of featurized data for 12 sensor dimensions to our Mites software backend, which provides several key features supporting large-scale data collection and APIs for application development. 12

2.2 Design of Mites sensor board (a–e) shows the multiple design iterations of our Mites device. Our final design (e) consists of nine discrete sensors with twelve unique sensor dimensions (vibration, thermal infrared, air pressure, magnetic field, light color, temperature, motion, Bluetooth devices, sound, WiFi signal strength, humidity, and light intensity). 12

2.3 Overview of our privacy-preserving data collection architecture. We show the three distinct DataStores with their metadata information. “Internal Data Store” keeps the entire mapping of actual locations and users and is kept separate. The actual location and the real owners are only added to the database with sensor data, “External Data Store”, after the user’s consent and are manually verified by a trusted Admin. The “Telemetry Data Store” stores the telemetry data for Mites devices to monitor the status of devices. 16

2.4 Screenshots of the Mites mobile application. Our app provides occupants with controls for Mites devices in their office and the ability to view sensor data from the devices accessible to them. (a) The onboarding process for the Mites device, which includes a login to our system and claiming a Mites device to their account, and gathering user consent. (b) Various privacy controls. (c) Live data viewer allows a user to view sensor data and annotate different events for training ML models. (d) Different “Applets” that they can install, which use the sensor data from their Mites devices, some of which can have their own consent screens. 18

2.5 Benchmarking Load balancing and Fault tolerance of Mites system with three Device Management Layer (DML) nodes (DML-1, DML-2, DML-3) that handle a total of 314 Mites devices. The area plot shows that all devices stream data to DML-1 initially. As DML-2 and DML-3 are instantiated at time instances t1 and t2, we see the Mites devices reboot and are *load-balanced* across the available DML nodes 2 and 3. Similarly, when the DML nodes go offline (At t4, DML3 is offline, and at t5, DML2 is offline), the remaining devices reboot and connect to the available DML nodes showing fault tolerance and recovery. We also see that the time taken for all devices to recover is very short, ranging from 18 – 31 seconds. 19

2.6 Efficacy of our adaptive packet rate scheme. On the left graph, we see that the number of reboots is much lower for lower send rates (1 Hz or 5 Hz) compared to sending data at the full 10 Hz. On the right graph, we show a histogram of the number of sensors binned into different packet rates. As seen in the histogram, our adaptive scheme has a much higher fraction of sensors in the 6 - 8 Hz and 8 - 10 Hz bins as compared to the 10 Hz static configuration while also observing significantly fewer reboots overall (not shown in the figure). The overall send rate for our adaptive scheme is 8.77 Hz (on average) compared to 7.16 Hz and 4.21 Hz for static 10 Hz and 5 Hz settings. 20

2.7 Comparison of Mites system performance with various system optimizations enabled as shown by the average % of packets delivered/minute and average reboots per day across all 314 Mites devices in our deployment, gathered over 12 days. We observe that the adaptive packet rate optimization (Days 4 - 6) results in more packets delivered compared (and lower reboots $\sim$25 across the deployment) to a statically configured rate of 10 Hz for all devices (baseline, Days 1 - 3). When we enable the opportunistic data send optimization (Days 7 - 9), we see that both the number of packets delivered and average reboots are lower since data is only sent from the Mites devices during periods of significant change from the 'Ambient Background.' Finally, when we enable both optimizations (Days 10 - 12), the average % packets delivered (reboots comparable) is generally higher and depends on the activities happening. 21

3.1 Overall System Architecture of MLIoT. The system consists of two components: the Device Selection Layer (DSL), managing devices and scheduling ML tasks, and the Training-Serving Layer (TSL), which instantiates workers for each ML task. 27

3.2 Overall architecture of the Training Worker (TW) and Serving Worker (SW). The TW is responsible for training models in parallel and optimizing hyperparameters using the input training data. Based on different policies, an ensemble of models is selected and sent to the SW. The SW uses the ensemble to make predictions using certainty estimation while aggregating serving side data for feedback from the user. 30

3.3 Timeline of the six application traces (Ts) where all the traces have diverse policy requirements. Traces have policy constraints on latency, resource and best effort. 34

4.1 Overview of the TAO's system architecture. TAO leverages OWL-based ontologies and temporal clustering approaches to identify context from the stream of activities obtained from Human Activity Recognition(HAR) systems. The contexts detected by TAO are then sent to our example Wellness application which then infers productivity and stress. 39
4.2 Overview of our ontological pipeline within TAO. 40
4.3 Query to infer *A: typing* 42
4.4 Query to infer *C: On a phone call* 42
4.5 An example of context representation (five activities only) for 2 min stacking window(t) and 10 min lag window(T). 43
4.6 Architecture of temporal autoencoder (TAE). 45
4.7 Overview of TAO's Temporal Pipeline. Representation Trainer(a) pretrains TAE Encoder-Decoder to learn dense context embedding from sparse input representation, then Cluster Trainer (b) uses TAE-Encoder and learned embeddings to discover optimum context cluster count and their sparse representation, and finally, Context Labeler(c) uses ontology to provide labels to generated context clusters. 45
4.8 Comparing the accuracy (JC), precision, and recall of context detection for TAO's temporal pipeline for different duration of *Lag Window*, across the *Casas* and the *Extrasensory* dataset. Our approach has high accuracy(65%-75%) for short (5 and 10 minutes) *Lag Window* in comparison to long (30 and 60 minutes) *Lag Window*. We also observed for both datasets, recall of context detection is consistently higher than the precision of context detection. 50
4.9 Comparing the accuracy (JC), precision, and recall of context detection for TAO's temporal pipeline as the data (in days) available for training for a single user increases. For the *Casas* dataset, our approach shows a consistent increase in accuracy, precision, and recall values as the training data increases. For the *Extrasensory* dataset, our approach shows high accuracy within two days of training data. 50
4.10 Comparing the distribution of predicted contexts learned over 45 days (1/7/14/30/45 days) for Casas and six days (1/2/4/6 days) for Extrasensory. In Casas, we see that with 30 days of training data, our pipeline can detect a wide range of contexts more accurately (72% JC). For the *Extrasensory* dataset, our approach shows that it can detect several contexts within two days of training data 51
4.11 Comparing the Accuracy (JC), Precision, and Recall of context detection on the two datasets. 52

5.1 An architecture of an audio-based activity recognition approach that takes featurized audio as inputs for applications such as cough recognition or event detection 54
5.2 The 39-phoneme table of CMUdict [66] 56
5.3 The phoneme and grapheme of an example word "pizzerias".Phonemes are the individual speech sounds composing words, while graphemes are the corresponding letters or letter groups representing those sounds. 56
5.4 FFT Spectrogram of several privacy filters proposed by prior work. 59

5.5 Illustration of Kirigami's Logistic Regression model for phoneme prediction. The LR Pred line graph shows the predictions from 0 to 1, while LR0.5 to LR0.1 shows predictions at threshold values indicating speech (1) or non-speech (0). The original and filtered spectrogram demonstrate a balance between privacy (speech filtering) and utility (activity recognition). 65

5.6 Flowchart depicts the adaptive background masking process in conjunction with Kirigami's speech filter. The process involves background detection, buffer comparison, heuristic calculation, and the generation of a background mask to filter out background frequencies. The resulting background-filtered FFT enhances Kirigami's speech filter for improved accuracy by eliminating background noise. 67

5.7 Results of (a) phoneme-based speech inference, (b) word-based speech inference, and (c) activity classification accuracy on prior filtering approaches. 70

5.8 Results of (a) phoneme-based speech inference, (b) word-based speech inference, and (c) activity classification accuracy on Kirigami filtering approaches. . . 71

5.9 Results of user transcribing sentence (a), recognizing words (b), and inferring speech topic (c) at various PER from ASR models. 73

5.10 Results of user transcribing sentence (a), recognizing words (b), and inferring speech topic (c) at various WER from ASR models. 74

5.11 Scatter Plot of Privacy and Utility Trade-Off of Different Audio Featurization Techniques. The vertical dashed lines represent the drop in accuracy for the utility measure in the presence of simultaneous speech and ambient sounds (overlaid sounds). Informed by our user study, we consider regions more than 60% PER and 80%WER as safe zones as little information from speech can be inferred. The ideal region is the top-right corner as it maximizes error in reproducing the spoken content and accuracy for the end goal task. For both plots, several Kirigami configurations are near that corner. In (a), SAMoSA [155] is close to the corner too, but the drop off in utility due to the presence of overlaid sounds is substantial. In comparison, Kirigami filters are more immune to noisy environments. 75

5.12 Comparing the performance of Kirigami LR filter with and without background mask from Microcontroller and Laptop. The figure shows the percentage of (a) speech-filtered and (b) activity data retained and overall activity recognition accuracy. 77

5.13 Comparing the performance of Kirigami LR filter on Laptop with and without background mask at various locations: L1-Lab, L2-Makerspace, L3-Conference Room. 78

5.14 Results of (a) phoneme-based speech inference, (b) word-based speech inference, and (c) classification accuracy on prior filtering approaches on the clean and noisy dataset. 79

6.1 Management and Maintenance applications built on the Mites system. Figures (a) and (b) showcase the spatio-temporal view of the temperature data of each office on a specific floor, captured from the Mites. The color-scaled temperature values that indicate the variation of warmth in rooms are less accurate for location-obfuscated data. (b) The location of individual Mites devices anchored on the floor plan. The color for each circle is filled with a corresponding scaled RGB value based on the number of reboots of that device – an important metric to assess the devices' health spatially. 84
6.2 Occupancy modeling applications built on top of Mites platform. Figure (a) showcases an occupancy application that detects the availability of conference rooms using PIR sensor data. Figure (b) shows an application that identifies quiet spaces in the building using acoustic features from multiple Mites devices (wall and ceiling) in the same location. 85
6.3 Activity modeling applications built on top of Mites and MLIoT platforms. Figure (b) shows the end-to-end ML toolchain illustrating data collection, annotation, model selection, training and inference. Figure (b) shows an activity recognition application that can detect activities in a kitchenette such as *making coffee* and *heating food* utilizing several sensor channels. 86

7.1 Overall timeline of the different projects outlined in this thesis. I highlight key events at the top and the features we worked on for different parts of our stack. The timeline shows the iterative design process, including multiple revisions to the Mites device, backend, and integration with services such as ML. 89
7.2 Overall packet rate performance (0 - 100%) of the each Mites device over a four-month period. From Oct 2021 - Jan 2022, we see the trend that the packet rate of devices changes from worse (dark blue) to better (light blue), while some of them continue to remain worse due to WiFi receptivity issues after several upgrades to the campus WiFi network maintenance (e.g., software updates to APs) over time. 90
7.3 The set of images showcases the development journey of the Mites hardware prototype, from initial design to full-scale deployment. The first five images highlight early-stage prototype designs. The next two images transition to the factory setting, where batches of Mites devices are assembled on production lines, showing the systematic process of building the hardware at scale. Next, a hands-on moment is captured where we manually flash firmware onto over 350 devices, ensuring each unit is correctly programmed. The final four images depict the deployment phase, showing the Mites devices installed on the walls and ceilings of a building at Carnegie Mellon University, ready to collect data and power real-world applications. 91

List of Tables

3.1 Hardware platforms used in our experiments 33

4.1 Comparing the accuracy (JC) and the total contexts detected by TAO's ontological pipeline only with prior work on two HAR datasets. Our approach detects different contexts (sequential and parallel) at a higher accuracy (72.8% – *Casas* and 53.9% – *ExtraSensory*). 49

5.1 Summary of evaluated speech filtering approaches 58
5.2 The list of evaluated ASR models against audio privacy filters. 60
5.3 Example inference results and Phoneme Error Rate (PER) from the CRDNN model. 60
5.4 Sample speech inference results and Word Error Rate (WER) from the Whisper model 62

Chapter 1

Introduction

In recent years, we have seen significant growth in Internet of Things (IoT) sensing systems in our daily lives, particularly in enabling the creation of "smart" buildings. The possibilities are endless, with these systems demonstrating unprecedented opportunities to revolutionize the way we live and work. From improving energy efficiency and sustainability [6, 9, 24, 49] to enhancing personal health, accessibility and overall quality of life, the benefits are undeniable [119, 147, 227, 241].

Despite the overwhelming potential of ambient IoT sensing systems, the real-world deployment and adoption of such systems have been limited and challenging. This challenge stems from a lack of system and architectural support capable of meeting the diverse requirements of IoT applications. These requirements span a wide range, including varied sensor data types, privacy safeguards, security measures, machine learning (ML) integration, scalability, and reliability. Furthermore, the deployment of such systems has raised significant privacy concerns [29, 52, 210], as they collect sensitive information through various sensors like audio, video, and vibration, potentially compromising individual privacy. Consequently, supporting these diverse applications and stakeholders requires a robust and flexible operating system in the complex and dynamic physical context of a "living" building, which requires a multifaceted technical solution that has so far been elusive.

The challenges and concerns associated with ambient IoT sensing systems are primarily due to their inflexible and monolithic system design. These systems have a limited capability to sense parameters and tend to focus on specific applications without incorporating adequate features for generalizability. Essential privacy-related features such as sensor access control, data minimization, and transparency are generally an afterthought. For example, several research efforts have proposed heterogeneous sensing systems in different application domains, such as energy analysis [198, 235], occupancy modeling [8, 30, 83, 112], thermal control [9, 26], building modeling [157], safety [182] and maintenance applications [33]. These systems depend on *purpose-built* hardware instead of *general-purpose* designs that only sense a limited set of parameters, such as presence (using PIR or motion sensors) or energy usage, often specific to a certain application use case. Such systems often lack accompanying tools for privacy, like device access control and management features supporting impactful long-term deployment. Moreover, these systems are typically vertically integrated stacks, offering minimal extensibility to support diverse application requirements. Such systems only support a subset of building stakeholders,

such as building managers or administrators, and cannot be easily extended to provide support for end-users in real-world IoT environments. Consequently, these systems are not widely adopted and have been deployed at a small scale, only for shorter durations, have limited functionality for end-users data access, and have limited primitives for privacy and security [16, 95].

To overcome these challenges, in this thesis, I aim to address the question: "*How can we build a general-purpose IoT sensing platform with comprehensive support for privacy and security from the ground up?*". *General-purpose* implies the system support required to enable the diversity of application and user requirements. This includes support for extensibility to accommodate the breadth of sensing, scalability to handle large-scale deployments and increasing data volumes, and reliability to ensure consistent performance and data integrity over time. Additionally, integrating machine learning (ML) capabilities is essential for processing and interpreting vast amounts of sensor data, providing actionable insights, and enhancing system intelligence. The *support for privacy* implies the tools required to achieve access control, data minimization, and transparency such that the IoT system doesn't access or leak private user data either directly or indirectly without a clearly defined and verifiable purpose being presented to and accepted by users. The *support for security* implies the need for system features for IoT systems only to allow authorized entities, whether computer programs or humans, to access their services. The vision articulated in this thesis is to create IoT sensing systems that are accessible and beneficial to end users, scalable to accommodate growing data and user demands, extensible to integrate new functionalities, reliable to provide consistent performance and secure to protect user privacy. Without these foundational elements, IoT systems risk being discarded along with many other technologies that showed promise but were ultimately ignored after deployment.

Thesis Statement: *By co-designing **general-purpose** features alongside robust **privacy and security measures** we can build **trustworthy** IoT sensing systems.*

1.1 Overview of this Thesis

This thesis aims to break down the existing monolithic and vertically integrated ambient IoT sensing platforms and design a novel general-purpose sensing system with a series of architectural optimizations for privacy, security, and scalable data processing that enable diverse applications and long-term real-world deployments. Specifically, as illustrated in figure 1.1, this thesis identifies opportunities for integrating architectural support for general-purpose sensing with comprehensive privacy features, proposing design insights and solutions in the following areas:

1. Designing a high-fidelity, general-purpose sensing platform.
2. Development of end-to-end machine learning system for IoT.
3. Developing a framework to generate rich contextual insights from ML-based inferences.
4. Implementing a general-purpose audio filter for privacy-preserving ML inferences.

My approach enables privacy support throughout the entire system stack, ensuring that potential privacy risks are minimized at every stage of data flow.

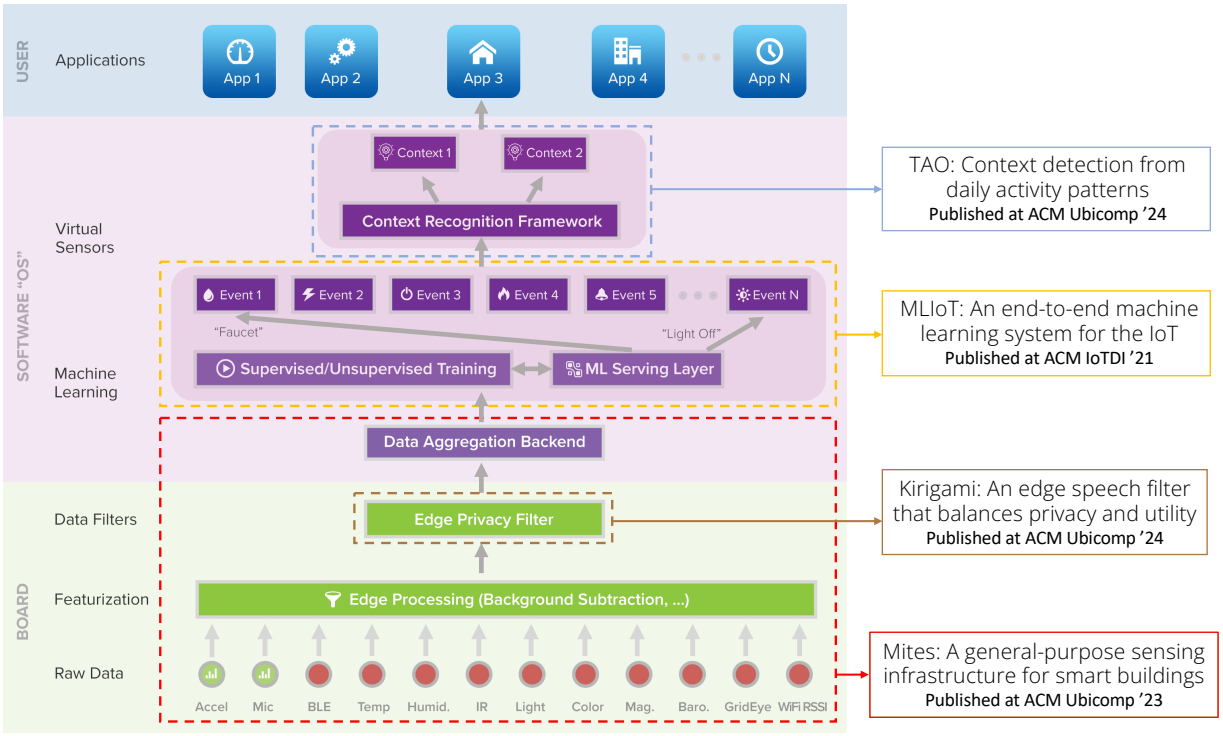


Figure 1.1: Overview of various components of an IoT system and how my work enables systems support for general-purpose sensing at different levels of the stack. This thesis aims to enable system support for general-purpose sensing and privacy at various components of an IoT system that can enable diverse applications and stakeholder requirements, ultimately fostering scalable long-term real-world deployments.

This thesis makes the following contributions:

1. **Designing a high-fidelity, general-purpose sensing platform (§2):** I first investigate the limitations and challenges of existing IoT sensing systems, particularly why these systems have been deployed only on a small scale and for shorter durations despite their potential to support diverse applications. I identify the pervasive challenge of accommodating a variety of application needs and stakeholder demands within an IoT sensing platform. In particular, the absence of essential system primitives for privacy, security, and scalability presents a significant hurdle to real-world deployment. Based on this, I identify the key design goals and requirements to guide the development of a comprehensive general-purpose sensing system.

   I present the design and development of Mites, a general-purpose sensing system that consists of a new physical sensor board and a backend architecture that supports fine-grained sensing in large-scale building deployments with a series of architectural optimization for scalable data processing, privacy, and security. Importantly, the Mites device firmware performs on-board featurization of the sensor data to obscure Personally Identifiable Information (PII). This 'edge' featurization balances data privacy while providing useful data supporting a diverse range of applications. I then present the design of our custom Mites software backend to handle data streams from hundreds of Mites devices efficiently and securely while supporting various primitives for data privacy and granular data access control by occupants. I evaluate

our iterative design decisions through a series of microbenchmarks and show that our system adapts to real-world network conditions and achieves high data delivery rates (94% of packets). I deployed and evaluated the Mites system in TCS Hall, a five-floor mixed-use office building on the Carnegie Mellon University campus.

2. **Development of an end-to-end machine learning system for IoT (§3):** In addition to the hardware and system stack for general-purpose sensing, another important challenge in existing IoT systems is the lack of an ML platform that supports the diverse application requirements. Most existing ML systems today are optimized for specific sensing modalities, require a dedicated hardware compute environment, and do not adapt to changing resources and different IoT application requirements. In a general-purpose sensing environment, specifically the Mites system, the ML systems need to have the ability to transform multimodal data into actionable insights and, importantly, account for model drift over time with real-world environmental changes, which requires models that need to be retrained and optimized. Additionally, as data flows up the ML stack, ensuring users' awareness and control over model types, the data being accessed, and local execution capabilities become paramount. Current ML systems often lack built-in privacy controls for executing computations on edge compute hubs or for isolating ML tasks associated with individual sensor streams. To address these multifaceted challenges, innovative solutions are required to enhance the adaptability and efficiency of ML platforms within general-purpose sensing environments.

   I propose the design and implementation MLIoT, an end-to-end machine learning system contextualized for IoT scenarios. MLIoT provides a flexible policy-driven selection of hardware platforms, ML models, and various optimizations for closely coupled training and serving tasks. More importantly, I designed MLIoT to provide user-expressed preferences (for edge computing, accuracy) and benchmarking device capabilities, providing fine-grained control in the ML model lifecycle processes. In addition, we incorporate several system optimizations such that MLIoT adapts to changes to the IoT environment or compute resources by re-training and updating the served models on the fly while maintaining accuracy and performance. Our evaluation across a set of benchmarks shows that MLIoT can handle multiple IoT tasks, each with individual requirements, while maintaining high accuracy and performance.

3. **Framework to generate contextual insights from ML-based inferences. (§4):** The initial chapters of this thesis will center on enhancing the general-purpose sensing of IoT platforms by enabling the transformation of low-level featurized data into high-level inferences, thereby bolstering its capability to support a wide array of applications. A key challenge, however, is that high-level inferences are complex in nature and often require contextual information about the inference to be useful for downstream applications. For example, a wellness application that assesses an individual's productivity requires contextual information about an activity. An activity inference such as talking may or may not indicate if the individual is being productive, depending on whether it is happening in a context denoting office work or a different context of having a meal. Thus, understanding higher-level and semantically meaningful contexts of daily activities is crucial to supporting applications such as tracking productivity or wellness.

   To address this, I propose TAO, a context detection system that combines ontological and deep unsupervised clustering approaches for inferring a rich set of contexts from a wide variety of

daily activities. The TAO system models the different activity patterns sequential, parallel, or interleaved activities as context information using the OWL-based ontologies. The temporal pipeline uses an unsupervised clustering algorithm to detect context from new activity patterns and automatically extends our ontology based on these patterns. I evaluate the TAO system on two large-scale real-world activity recognition datasets, *ExtraSensory*[215], and *Casas*[57] and show that TAO system achieves $70-75\%$ accuracy (measured using the *Jaccard Similarity Coefficient*(JC)) for context detection of new activity patterns across multiple users. With this approach, I hope that users have more control over when their activity inferences are accessed by applications based on the context information of the activity.

4. **General-purpose edge audio filter for privacy-preserving ML inferences (§5):** So far, this thesis focuses on designing and developing novel solutions to enable system general-purpose sensing for IoT systems, particularly addressing the requirements of different applications and stakeholders. To ensure privacy, one of the key features I developed is on-device edge-featurization to ensure that raw data never leaves the device. This edge featurization removes sensitive information before data is sent off the device, thereby helping protect privacy. However, several approaches for data featurization exist for sensor data, each with varying privacy risks and utility tradeoffs. In this final part of the thesis, I will systematically characterize various featurization techniques on audio data, particularly those that extract statistical and spectral features using Fast Fourier Transforms (FFTs), and evaluate the privacy risks and utility tradeoffs. I first explore different FFT-based featurization approaches proposed in prior works that aim to remove sensitive information from raw audio while providing utility to activity recognition tasks. I will then study the recent advancements in deep learning-based automatic speech recognition (ASR) and their potential impact on these edge audio featurization techniques. I will also investigate the utility of different featurization approaches in generating discernible features for machine learning prediction. I then propose Kirigami, a general-purpose edge audio speech filter that is resilient to various speech recognition or audio reconstruction techniques while being feasible to implement on edge devices with limited computational power. Our overarching goal is to create a framework that practitioners can use to systematically evaluate different featurization and filtering approaches to understand the tradeoff between the level of privacy protections and the utility of being able to make activity inferences on the featurized data.

1.2 Thesis Outline

This thesis is organized as follows: Chapter 2 details the design and development of a general-purpose sensing infrastructure for IoT environments, emphasizing scalability and real-world deployment challenges. Chapter 3 delves into implementing an end-to-end machine learning platform for IoT, addressing the lifecycle of model training, serving, retraining, and deployment to effectively adapt to dynamic IoT environments. Chapter 4 introduces a context-sensing framework designed to elevate the abstraction of activity inferences, enhancing the granularity and utility of sensing data in diverse applications. Chapter 5 systematically evaluates privacy and utility trade-offs in feature engineering techniques for IoT sensors, focusing on developing a general-purpose edge audio filter for privacy-preserving machine learning inferences.

Chapter 2

Design and Deployment of a General-Purpose Sensing Infrastructure

The vision of building scale, rich-sensing infrastructure that powers a wide variety of sensing applications has been a long-standing research goal. Application domains span from infrastructure utilization (e.g., conference room availability, coffee machine utilization, parking garage capacity, lounge occupancy, trash receptacle overflows) [14, 49, 166] to occupant wellness and productivity (e.g., room temperature, noise level, artificial lighting intensity, color temperature, air circulation, and HVAC performance) [119, 147, 227, 241]. Such data is also vital in making buildings more sustainable and maintainable.

Supporting these diverse applications and stakeholder needs, as well as robustly and flexibly operating in the complex and dynamic physical context of a "living" building, requires a multi-faceted technical solution that has so far been elusive [29, 52, 210]. While existing research has proposed many building-scale distributed sensing approaches [5, 8, 25, 26, 44, 111, 136, 198, 199, 212, 235], comprehensive and real-world deployments of general-purpose IoT sensors continue to be relatively rare and challenging [16, 81, 95, 213]. A major limitation is that today's building-wide sensing approaches are not built with the necessary primitives for long-term and stable deployment at scale or with the richness in sensing necessary to support a diverse range of applications. For example, several research efforts have proposed heterogeneous sensing systems in different application domains, such as energy analysis [198, 235], occupancy modeling [8, 30, 83, 112], thermal control [9, 26], building modeling [157], safety [182] and maintenance applications [33]. These systems depend on *purpose-built* hardware instead of *general-purpose* designs that only sense a limited set of parameters, such as presence (using PIR or motion sensors) or energy usage. Fidelity is typically low, and systems lack accompanying tools, interfaces, and features to support impactful long-term deployment. Even state-of-the-art commercial building management systems such as those offered by Johnson Controls [113, 204] sense a limited set of parameters (e.g., presence, temperature, humidity) consisting of vertically integrated stacks that are most often single-point, vendor-controlled solutions with little to no extensibility to enable other applications. While numerous cloud-based sensor kits offer rapid prototyping capabilities (e.g., Adafruit FeatherBoards [4], Arduino/RaspberryPi + miscellaneous sensor shields [145, 184]), none of these approaches are integrated with necessary components such as storage, security, and machine learning components that are critical for practical deployment.

Several systems [5, 111, 128, 199, 212] have attempted to overcome at least some of these limitations with cloud-connected, general-purpose sensing infrastructure. Most often, these efforts deploy sensor suites that consist of multiple sensor modules, either as a single device or a constellation of devices. Although these approaches have shown the promise of high-fidelity sensing and machine learning, these systems have only been deployed at a small scale, only for shorter durations, have limited functionality for end-user's data access, and have limited primitives for privacy and security [16, 95]. In addition, these systems only support a subset of stakeholders, such as building managers or administrators, are not easily scaled to operate in real-world IoT environments, and often do not support closely integrated ML tools.

This chapter of the thesis illustrates the key insights in developing a unified, high-fidelity, and privacy focussed general-purpose sensing system for smart buildings. We first present the key challenges in developing a general-purpose sensing system and then showcase the widespread issue of missing system primitives for privacy, security, and scale required for real-world deployment. We then identify the key design goals and requirements to guide the development of a privacy focussed general-purpose system.

To achieve our vision, we present the design and development of Mites, a general-purpose sensing system that consists of a new physical sensor board and a backend architecture that supports fine-grained sensing in large-scale building deployments with a series of architectural optimization for scalable data processing, privacy, security, and machine learning. We first present the design of a custom Mites hardware device to provide the most comprehensive suite of sensors to enable general-purpose, high-fidelity sensing. We then present the design of our custom Mites software backend to handle data streams from hundreds of Mites devices efficiently and securely while supporting various system primitives for data privacy and granular data access control by occupants. To showcase the design features of Mites, we developed five proof-of-concept applications across three application domains (building maintenance, occupancy modeling applications, and activity modeling applications), targeting different stakeholders.

The rest of this chapter describes Mites at a high level. Additional details on our completed work in this space are presented in a full paper on this topic [37].

2.1 Challenges

This chapter identifies challenges in enabling a general-purpose sensing system for buildings.

2.1.1 Diverse Application Requirements

Smart building applications have diverse requirements in terms of sensor data, privacy, security, support for machine learning, scale, and reliability. For example, environmental modeling [14, 49], management and maintenance [69, 166], energy apportionment [6] only require data at a coarse granularity or at a low temporal fidelity (e.g.,, average hourly temperature across floors, etc.). In addition, these applications may require less privacy-sensitive data (such as temperature or humidity) and may not need support for machine learning. On the other hand, applications such as occupancy-based HVAC control [9, 24] and sophisticated fault detection algorithms [49, 158] need highly fidelity data from multiple sensors, including those denoting

occupancy. More importantly, emerging use cases such as detecting human activities using ML [119, 127], or detecting semantically meaningful contexts for occupant productivity or wellness [119, 147, 241], require rich sensor data from multiple modalities at finer granularities and fidelity to be not only effective but also often need data from more privacy-invasive sensors such as audio or presence/movement. Ultimately, supporting these current and emerging applications needs support for rich multi-modal sensing, data annotation tools, built-in ML support, and extensive support for privacy controls.

2.1.2 Diverse Stakeholder Requirements

Buildings consist of diverse stakeholders, including the occupants, building/facility managers, and administrators, each with their own requirements for privacy, security, and potential uses of IoT sensor data. These requirements have come to the forefront given high-profile compromises of consumer IoT devices [41, 76, 165, 248], general safety and security issues with them [237], and consumer concerns about their privacy implications [72]. These requirements are, however, different across stakeholders. Building managers require the sensing hardware and infrastructure to be secure and can access the sensor data from Building Management Systems (BMSs) to monitor and manage the overall building. In addition, they require the sensing infrastructure to support several use cases (e.g., understanding general building occupancy trends to optimize lighting schedules and maintaining the IoT network) with significantly less granular data (e.g., averages across the entire floor). On the other hand, building occupants have traditionally not had access to any of the sensor data from the BMS, including even temperature/humidity trends from their own offices. Furthermore, given the increasing privacy concerns with IoT [72], occupants will naturally require expressive mechanisms for notice (e.g., what sensor data is being captured, what apps are using this data) and choice (e.g., turn on/off sensors in their own spaces, data sharing controls). Most importantly, we must ensure that any collected sensor data cannot personally identify an individual (no-PII) and that any indirect association risk of an occupant and the data from their office is also mitigated.

2.1.3 Lack of System Primitives to Support Long-Term Deployments

Supporting existing and emerging smart building applications requires high-fidelity sensing, as mentioned above. However, the data generated from such IoT devices cannot be transmitted over low-power, low-bandwidth networks, and cost/practicality necessitates leveraging existing WiFi infrastructure in buildings. Furthermore, with a scale of hundreds to thousands of these sensors in a building, the backend design to gather data must dynamically adapt to efficiently manage all these data streams based on the available compute resources [29, 52]. Finally, to support long-term deployments lasting many years, the system design must be resilient and fault-tolerant to handle common failures that are either intermittent (network drops) or more catastrophic (e.g., machines crashing or power outages) and be able to recover. Last but not least, these features need to be supported in a way that makes it easy to manage a network of this scale.

2.2 Background

This chapter provides background on IoT device software and hardware stacks related to IoT sensors and IoT cloud platforms.

2.2.1 IoT Sensors

In recent years, both researchers and industry have made strides towards providing hardware sensing solutions that capture different environmental facets [30, 78, 145, 161, 209, 221]. These fall broadly into three categories: (1) *Prototyping platforms*, (2) *Special-Purpose*, and (3) *General-Purpose*. *Prototyping platforms* often include developer kits that have sensors and actuators for communities wanting to build quick prototypes and proofs-of-concept (e.g., Phidgets [176], Gadgeteer [221], Arduino or Adafruit Featherboards [4]). Other *Special-Purpose* sensing approaches target a single environmental facet using single or multiple sensors, towards specific special-purpose applications. For example, Risojevic et al. [197] designed a low-power sensor node with a microphone to detect ambient sound levels, while ThermoSense [30] used a low-power thermal sensor array and a PIR sensor to estimate room occupancy. In contrast, *general-purpose* sensing advocates the use of several sensing modalities to detect multiple events and environmental facets at the same time. For example, TI's Sensor Tag [209] integrates numerous sensors on a small battery-powered device with an accompanying smartphone app to gather data over the BLE radio. While promising, such approaches generally have small-scale deployments (e.g., homes) for short durations, focused on sensing technology and specific applications (e.g., activity recognition). For long-term, large-scale building deployments, these approaches fall short in providing the right primitives, such as security/privacy, scalability, reliability, and integrated ML.

2.2.2 IoT Cloud Platforms

Over the years, several IoT cloud-based platforms (such as Microsoft IoT Core [150], AWS IoT [20], Arduino Cloud [17], and ThingWorx [180]) have emerged to support the developers of IoT devices. Although these platforms provide several functionalities relevant to IoT sensor deployments (e.g., data storage, scalability, reliability), they lack the features and customizability to enable useful sensing in buildings. These platforms are hardware agnostic and general-purpose in nature but lack the end-to-end design to support several design requirements of a high-fidelity sensing infrastructure. Furthermore, the security and privacy support are limited given these are closed vendor platforms, with the lack of expressive and granular privacy primitives needed for sensor data (e.g., controlling individual sensor streams) or even where the data is stored and processed (often no on-premise support). Although these platforms support the use of the other cloud-based ML systems (e.g., [1, 154]), they are generic in nature and are not meant to be tailored towards the smart building use cases.

2.2.3 Combined Systems (IoT Sensors + IoT Cloud Platforms)

The most relevant related works are research efforts and commercial systems that consist of IoT sensing hardware and an accompanying software backend. Within a *Living Lab* context, Philips

HomeLab [62], PlaceLab [103] and Georgia Tech's Aware Home [122] instrumented spaces with cameras and microphones and other simpler sensors for activity recognition, behavioral monitoring, etc. Recent research approaches towards creating a *Smart Home* has focused on instrumenting real-world homes with sensors for applications such as activity recognition or elder care. For example, Klakegg et al. [123] proposed a non-intrusive sensing platform by leveraging proximity and contact switches to recognize activities of the elderly living alone. However, these approaches are geared towards smaller-scale home setups, and their focus is either on the sensing technology itself or building an application toward activity recognition. In addition, they provide limited, if any, support for the building requirements for high-fidelity sensing, scale, security, privacy, reliability, and management.

Researchers have also proposed smart building deployments for different applications such as occupancy detection [9, 78], energy management [136], etc. Most of these sensor deployments were prototypes that lasted for a short duration of the study. Projects such as SmartCampus [159] and Sensor Andrew [199] deployed IoT devices within buildings with sensors to measure temperature, humidity, etc., in real-world office environments. Similarly, MakeSense [111] deployed several custom-built sensing devices called IoTEgg to capture occupant activities using environmental data obtained from the device. These deployments, again are not geared towards supporting high-fidelity, general-purpose sensing, nor do they address many of the requirements for long-running deployments to support diverse applications for different stakeholders.

2.2.4 Design Goals

Motivated by the above challenges and background, we define key challenges to enable a practical general-purpose sensing infrastructure for smart buildings.

- **D1: Rich Sensing to Enable Diverse IoT Applications:** To support existing and emerging smart building applications with diverse sensing needs, the underlying sensing infrastructure should support capturing a wide variety of physical phenomenon (*D1.1: Breadth*). Furthermore, sensor data must be synchronized and sampled at sufficient temporal fidelity to enable robust capture of subtle (e.g., writing on the whiteboard) and transient signals (e.g., door closed) in order to be truly general purpose. Additionally, the machine learning output must be accurate and stable (*D1.2: Accuracy*).
- **D2: Built-in Support for Privacy and Security:** Most IoT platforms and deployments do not provide sufficient security primitives or privacy notice and choice mechanisms [233]. Since these sensors are present in private offices and shared spaces, captured sensor data should prevent the identification of an individual (*D2.1: No PII*), including mitigating indirect association using metadata about occupants and the sensor location (*D2.2: Mitigate Association Risk*). In addition, security primitives are needed to prevent interception of and tampering with sensor data (*D2.3: Data Security*) and mitigate potential adversarial attacks (*D2.4: Tamper Resistant*). Finally, the occupants should have expressive mechanisms for notice and choice, i.e., to know and control what data is being collected in their space, how it is used, and who can access it (*D2.5: Privacy Controls*).
- **D3: Scalable, Reliable and Resilient System for Long-Term Deployments:** Building-scale IoT systems need to handle the computational requirements of several thousands of

sensors efficiently (*D3.1: Efficient Compute*) by leveraging existing network WiFi infrastructure (*D3.2: Efficient Network Use*), as well as dynamically adapt to available compute and network resources (*D3.3: Adaptable*). Finally, the system design should provide reliable data collection from a dense sensor deployment and recover from common types of failures seamlessly (*D3.4: Reliable and Resilient*).

- **D4: Comprehensive Support for System Management and Maintenance:** IoT deployments in buildings, particularly research efforts, are often short-lived and expensive to maintain beyond a few weeks [79, 108] since they lack features to monitor and manage devices at scale. Hence, for multi-year operation, the system needs mechanisms for monitoring the health of all components (*D4.1: Comprehensive Monitoring*), as well as the ability to remotely manage individual devices (*D4.2: Device Management*).
- **D5: Integrated Machine Learning (ML) Capabilities:** The system design should support the entire lifecycle of ML inferencing on multi-modal IoT sensor data. This includes user interfaces to allow different building stakeholders to view live sensor data and to provide training labels for inferences and events they are interested in (*D5.1: Data Annotation and Management*). Furthermore, the system must efficiently handle training and inference requests from hundreds of sensor streams and applications based on available compute resources (*D5.2: Efficient ML*).
- **D6: Designed for Extensibility:** Current IoT-based sensing solutions for buildings tend to be vertically integrated systems designed for limited purposes, often just one (e.g., occupancy, motion-triggered lighting), and are not readily extended to support other use cases [113]. Furthermore, any updates to include new features over time require extensive ground-up redesign. Thus, the system design should use extensible APIs that enable adding new components and applications (*D6.1: Extensible Architecture*) supporting different stakeholders (*D6.2: Diverse End-Users*).

2.3 Mites System Architecture

We describe the architecture of our Mites system, highlighting how we achieve our key design goals from Section 2.2.4. We describe each of the underlying components (as shown in Figure 2.1) in further detail, namely, the Mites hardware sensor package (Section 2.3.1) and the firmware (Section 2.3.2) and our Mites software backend (Section 2.3.3).

2.3.1 Mites Sensor Board

The lowest foundation layer of our stack comprises our Mites hardware package. Our goal is to provide high-fidelity sensing of various ambient environmental facets in physical spaces in a building, ultimately enabling a diverse set of IoT applications. These applications include existing ones such as energy apportionment [7], managing HVAC systems [9, 23, 24], improving occupant comfort [26, 30, 54], occupancy analytics [54, 218], automatic fault diagnosis [158]. In addition, numerous new applications can be built based on the rich set of ML-powered inferences, such as detecting the context of office occupants, improving occupant productivity, detecting

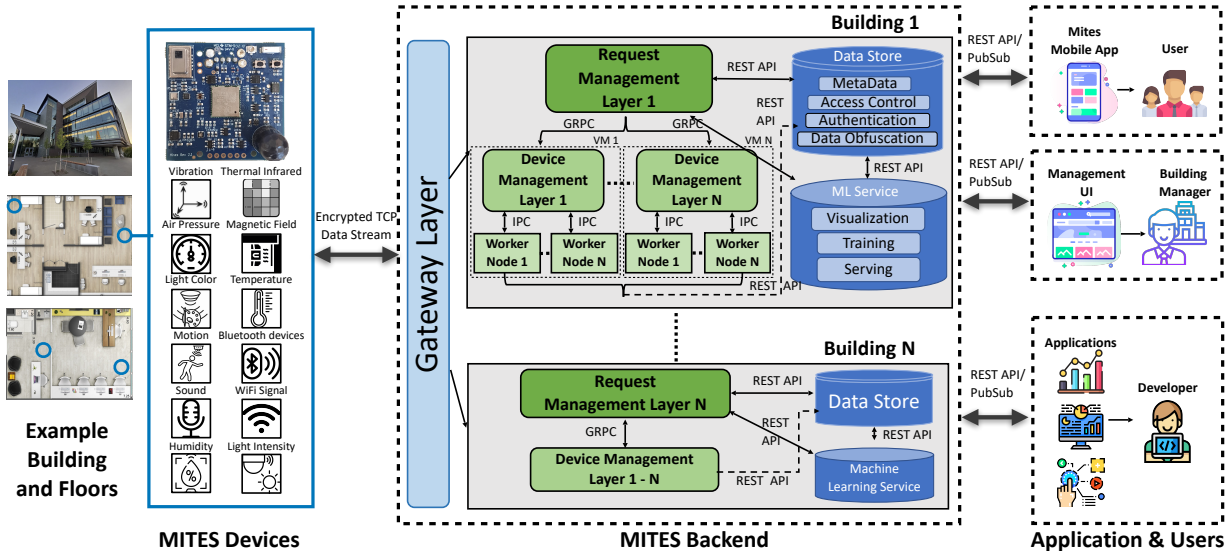


Figure 2.1: Overview of the Mites system and deployment. Each room has a Mites device on the wall and in the ceiling, with larger rooms and shared areas having multiple devices in the ceiling. Each device sends an encrypted stream of featurized data for 12 sensor dimensions to our Mites software backend, which provides several key features supporting large-scale data collection and APIs for application development.

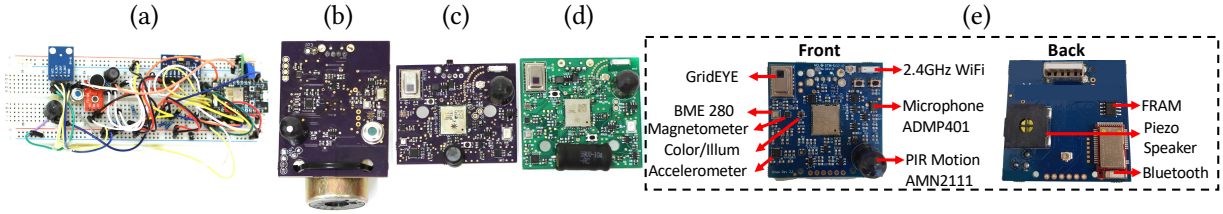


Figure 2.2: Design of Mites sensor board (a–e) shows the multiple design iterations of our Mites device. Our final design (e) consists of nine discrete sensors with twelve unique sensor dimensions (vibration, thermal infrared, air pressure, magnetic field, light color, temperature, motion, Bluetooth devices, sound, WiFi signal strength, humidity, and light intensity).

resource waste, and improving facility management. Since our Mites deployment was in a brand new university building, we had the opportunity to work closely with the architects in the building design phase itself. We had contractors install PoE switches in network closets and run Ethernet cables to all Mites locations in walls and ceilings. However, in existing buildings, Mites devices will have to be incrementally deployed, and hence we also have a second version of our device which uses a standard wall-powered 5V USB adapter. In addition, we decided to use WiFi (2.4 GHz) for data transfer in both versions, as WiFi is ubiquitous in almost all buildings and does not require additional PoE switches and infrastructure.

Designing the Mites Device to Capture a Diverse Set of Physical Attributes: While there are several multi-modal sensors [111, 128, 209], our goal was to create a device with the union of numerous sensor modalities to capture a wide variety of environmental facets (*D1.1: Breadth*). In addition, the multi-modal sensors used in prior approaches were custom prototypes for small-scale studies, and their hardware or software was not available for us to modify. More impor-

tantly, building our own hardware device allows us to choose specific sensors and control all aspects of the firmware, including signal processing and computation performed on edge. Figure 2.2 illustrates the multi-year iterative hardware design process [160] which was crucial in identifying hardware problems early, improving sensing fidelity, helping with sensor selection, and managing hardware revision costs. The first set of revisions, shown in Figure 2.2(a-d), was important for us to explore different sensors, accuracy, range, fidelity, space/volume, and cost tradeoffs. We explored alternatives, for example, using a geophone [151] vs. various 6-axis IMUs for measuring ambient vibrations. In our latest design, shown in Figure 2.2(e), we settled on the TDK InvenSense MPU-6500 [208] as it provided a good trade-off with respect to its footprint and sensitivity. We also incorporated a Bluetooth Low Energy (BLE) module, the MDBT40 [188], which enables each Mites device to advertise its presence and services using our custom implementation of the beaconing mode (iBeacon or Eddystone protocols). Altogether, we believe our design offers the most comprehensive set of sensors in a single package to date (*D1.1 - Breadth*).

2.3.2 Mites Firmware

Edge Featurization to Denature Sensor Data: We featurize and denature the raw sensor data on-board to ensure that raw privacy-sensitive data does not leave our Mites device. For example, the microphone data is converted into a low-fidelity spectral representation alongside basic statistical features (min, max, median, variance, etc.) to prevent reconstruction of the source audio or to decipher speech or detect the speaker's identity. We have two high-frequency sensors: A microphone (16 kHz) and an accelerometer (X, Y, and Z at 4 kHz). Furthermore, we have nine low-frequency sensors (sampled at 10 Hz): temperature, humidity, air pressure, light color, light intensity, WiFi RSSI, motion, magnetometer (X, Y, and Z), and the GridEYE sensor [169].

For our high-frequency sensors, such as the microphone, we maintain a rolling 256-point buffer for each sensor channel's time-series data. We then calculate a Fast Fourier Transform (FFT) for time-series data on the device. Specifically, every 100 ms, we take the first 256 values out of 1600 values (as data is sampled at 16 kHz), discarding the rest of the data (13440 of 16000 samples per second). We then calculate a low-resolution FFT that produces 128 frequency bins to further denature the data. We only keep the magnitude information from the FFTs and discard phase information, further preventing reconstruction of the original signal. In addition, note that the send rate of the sensor can be further reduced by building managers or end users from 10 Hz to 2 Hz (2 FFTs), 1 Hz (1 FFT), etc.

For our low-frequency sensors, we keep a rolling ten-sample buffer that stores approximately one second of data. Before data transmission, we compute the same seven statistical features used above: min, max, range, average, sum, standard deviation, and centroid. This level of featurization ensures that the raw sensor data is denatured before being sent out, unable to be reconstructed, and substantially safeguards against PII from being collected (*D2.1: No PII*).

We selected the features (FFT and statistical values) based on prior literature to ensure that the featurized data from these sensors can detect multiple environmental facets with varying temporal characteristics. For instance, activity recognition applications [32, 49, 55, 127] that need to identify subsecond- to seconds- scale events (e.g., *door knock* to *brewing coffee*) have been supported by FFT data from high-frequency sensors (e.g., Accelerometer, Microphone). Simi-

larly, applications [14, 69] that need to track hour-, day- to week-long events such as *day light patterns, lighting usages* can be characterized by the statistical data such as average, standard deviation from the low-frequency and slow varying sensors (e.g., temperature, humidity, color, and illumination). Thus, we use a combination of FFT and statistical features (min, max, standard deviation, etc.) for our high and low sample sensors to enable a good tradeoff between privacy and generalizability to different applications.

Sensor Control: Given the wide variety of physical sensors (particularly a microphone) on the Mites device, and its ubiquitous deployment across public/shared spaces and private offices, a key requirement is that authenticated users, or designated administrators, can enable/disable one or more specific sensor streams on each Mites device. This feature is essential for privacy (*D2.5: Privacy Controls*), as it gives users visibility and control of what is being sensed in their private offices/spaces.

Secure Data Communication: To set up an end-to-end secure channel between each Mites device and our backend, the system firmware on our devices combines standard asymmetric key cryptography to bootstrap a symmetric session key that is unique per connected device. Each Mites device creates an asymmetric key pair, and during initial commissioning, its public key is sent to the Mites gateway, and the gateway's hostname is added to each Mites device's flash storage. Thus, each Mites device and the Mites backend can mutually authenticate each other. After the initial handshake, the backend creates an AES 128-bit symmetric session key for each session with a particular Mites device, rotated periodically. With these safeguards, we prevent the possibility that an adversary takes over a Mites device remotely by changing the hostname-to-IP-address mapping, as the Mites devices will not be able to complete a successful handshake and will not send any sensor data. In addition, a tampered device that does not communicate with our backend will raise a flag and generate a notification to administrators (*D2.3: Data Security & D2.4: Tamper Resistant*).

2.3.3 Mites Software Backend: Architecture and Design

We designed a custom backend with capabilities to efficiently handle all encrypted data streams from hundreds of Mites devices in a typical building deployment (Sections 2.3.3 and 2.3.3). We also introduce a novel mechanism for privacy-aware data collection (Section 2.3.3). We now review these components in greater detail.

Adapting the Device Packet Rate based on Real-World Network Conditions: At first glance, the maximum data rate of 20 KB/s per Mites device does not seem high even with hundreds of devices, and we assumed that enterprise WiFi networks would easily handle it. For example, our building deployment with 314 deployed Mites devices, translate into a mere total bandwidth of 6.5 MB/s. Our real-world empirical measurements, however, showed that our devices could not maintain this 10 Hz send rate reliably due to device reboots and packet loss (as shown in Figure 2.6(a)). There are numerous reasons for this, including only three non-overlapping channels in 2.4 GHz leading to co-channel interference, channel contention caused due to relatively small packets [219] sent by hundreds of Mites devices, and other WiFi devices on the same network.

To address this challenge, we designed a simple adaptive packet rate scheme where each Mites device adapts its send rate to the underlying network conditions (*D3.1: Efficient Compute & D3.2: Efficient Network Use*). Our scheme increases the packet rate of a device whenever they send reliable data within each hour (additive increase) and approximately halves the send rate in case of observed packet drops (multiplicative decrease) to discrete levels 1 Hz, 5 Hz, or 10 Hz. Our scheme leads to individual Mites devices with different send rates based on their observed network conditions, and our evaluations show that it significantly improves the overall packet rates as compared to a statically configured send rate. A key lesson based on our experience is that while WiFi networks are pervasive and attractive for IoT deployments, using them for building-scale, high-fidelity general-purpose sensing requires comprehensive mechanisms to adapt to existing network conditions dynamically.

Opportunistic Data Sending: The total amount of data sent by each Mites device over a day translates to 1.65 GB at 10 Hz. However, only a fraction of these sensor data is actually useful and likely to reveal interesting events given long periods of inoccupancy and no human activity, including nights and weekends. Ultimately, this leads to redundant WiFi data transmission, computational costs to process the data, storage for the backend databases, and unnecessary computation from our ML service. Instead, in Mites, we propose an opportunistic data sending approach by adding edge intelligence on the Mites device. We implemented an anomaly detection algorithm using Euclidean distance as a metric in the Mites firmware with two classes: (a) the "Ambient Background"; and (b) all the other classes of interest. (*D3.3: Adaptable*) As a heuristic, we capture this ambient background profile for each Mites device late at night when we observe no movement. Then, we use this adaptive background model for each sensor channel (rolling mean and standard deviation). We store the "Ambient Background" values on each Mites device itself and calculate a normalized Euclidean distance metric with actual featurized data.

As part of a microbenchmark, we collected data for 12 different real events (e.g., activities such as talking, and knocking) for a Mites device in an example office. We experimented with a number of different non-ML-based methods such as sending data when motion is detected using the PIR sensor, or distance-based anomaly detection algorithms such as Euclidean, Dynamic Time Warping (DTW) and DTW-windowed. In addition, also explored several ML-based methods (Linear Outlier, KNN, LR) for anomaly detection and measured their accuracy, missed events (false negatives), latency and model size. The comparisons are shown in our paper [37] in more detail. We ruled out using ML-based methods due to the model sizes and/or their computational complexity. Ultimately, we found that the Euclidean distance metric worked best. In the firmware of each Mites device, we calculate the Euclidean distance between two arrays: the current featurized data and the data from its own "background" profile. Each device only sends the data if it is greater than a threshold (set conservatively). In addition, each device still sends *periodic keep-alive* packets at a configurable interval (set at $\sim$5 mins) regardless to our backend.

Architectural Design for Privacy-Aware Data Collection: Given that some of the Mites devices are located in offices with a single or a limited set of shared occupants (2 - 4 people) there is a risk of indirectly associating (with some probability) the sensor data in those spaces with the behavior of one or more of those individuals. For example, while the PIR data in an office indicates *someone's* presence, it is *more likely* to be due to the actual office occupant(s).

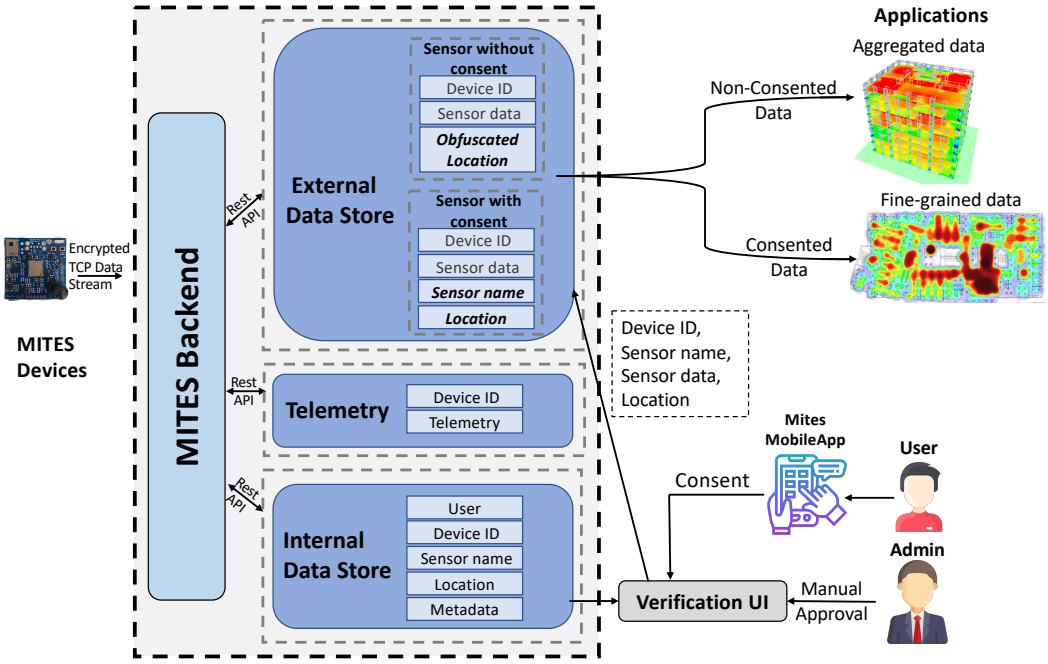


Figure 2.3: Overview of our privacy-preserving data collection architecture. We show the three distinct DataStores with their metadata information. "Internal Data Store" keeps the entire mapping of actual locations and users and is kept separate. The actual location and the real owners are only added to the database with sensor data, "External Data Store", after the user's consent and are manually verified by a trusted Admin. The "Telemetry Data Store" stores the telemetry data for Mites devices to monitor the status of devices.

In our threat model, we assume that researchers and building managers managing the system are trusted. We consider outside attackers who may try to gain access to sensor data and associate it with individuals who may not have yet consented to data collection. In addition, we assume that developers of apps (e.g., administrators and community members) may try to access more data than they need (over-privileged apps). We also assume that authenticated occupants may be honest but curious and may try and access data they don't have access to (e.g., from someone's office) or turn on/off sensors in their own spaces without the knowledge of their co-occupants.

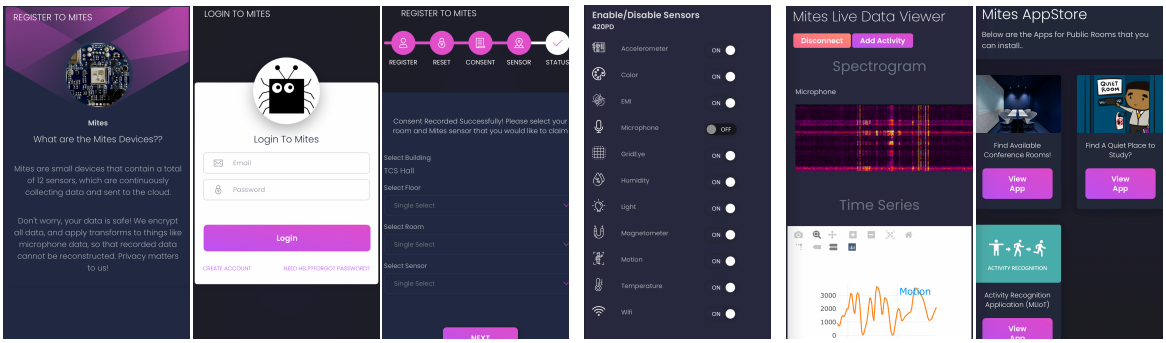
Given this potential association risk, we explored a novel design to enable privacy-aware data collection from individual/shared offices by removing the precise location of the Mites device by default (*D2.2: Mitigate Association Risk*). Our approach works as follows. A Mites device in an office is initially tagged with an obfuscated location ID that corresponds to a group of offices on a floor instead of its actual location/room. For example, a Mites device will have location metadata as `Corridor:300; Cardinality: East; Room #Random_Hash` instead of its actual location in `Room:301, Floor 3`, indicating that it could be in any one of the several offices in that corridor (e.g., there are 10 offices in corridor 300, on the East facing side of our building).

Notably, we choose these obfuscated location IDs to ensure the grouping of a minimum set of offices (e.g., 7), thus breaking the association of a Mites device and any data from it with an exact location (i.e., an office). Subsequently, if an occupant wants to interact with the Mites

device in their office, we collect their informed consent to associate the actual location of the Mites device with its data, since for most applications (e.g., HVAC control, viewing sensor data in your office) the actual location is necessary. For shared offices, we require consent from all occupants before we can associate the Mites device data with its actual location. However, there are corner cases where our approach needed further refinement. For example, consider the case when after a number of office occupants have consented (i.e., real locations associated with their Mites devices), only a few offices in a corridor remain unconsented (i.e., the Mites devices using obfuscated location IDs). In this scenario, it is feasible for an adversary to indirectly infer the real location of an obfuscated location ID by observing which locations are missing from a corridor (i.e., only Rooms 301 and 302 are missing from Corridor:300). To address this, when the number of non-consented offices in a particular corridor and cardinality (e.g., C300 East) drops below a threshold (7 offices), we automatically obfuscate the remaining anonymous location IDs to be completely random e.g., "Corridor:Unknown;Cardinality:Unknown; Room#Random_Hash". Furthermore, to ensure at least seven random IDs, we designate a random set of Mites devices location IDs to also have anonymous location IDs.

Our privacy-aware data collection architecture is shown in Figure 2.3, which includes three separate Data Store instances. The "Internal Data Store" stores all metadata, such as device name, device identifiers, and the actual locations (e.g., Floor, Room, Building) of the Mites devices along with occupant information. This information is kept separate, with limited access, and is not used during regular operation. "Telemetry Data Store" stores the telemetry data for Mites devices such as reboots, packet rates, and overall uptime, as well as logs for backend status monitoring. The "External Data Store" stores all sensor data from the Mites devices, using obfuscated locations for sensor data for spaces with unconsented occupants. The actual location and the owners are only added to the "External Data Store" after the user's opt-in consent and are manually verified by a trusted administrator. Note that sensor data with obfuscated locations are still useful for certain applications and aggregate statistics, e.g., the average temperature for rooms in a corridor or coarse occupancy patterns at the floor level, without any privacy risks. This approach prevents the indirect association of the non-consented individuals with the Mites sensor data collected in their offices.

Data Model to Create Views of Sensor Data for Privacy: Applications may need data from one or more sensors on a Mites device (e.g. occupancy detection may need PIR and thermal GridEYE data). Similarly, occupants may want to share data from a subset of their sensors with other users. Thus, having fine-grained mechanisms to enable/disable access to specific sensor(s) from a Mites device, as well as specifying the level of access (read, write, change permissions) is necessary to prevent overprivileged apps. Our data model uses a notion of a 'parent' sensor to which we post all the data from a Mites device and then implement 'data views' which can be created on demand to grant/revoke read or write access to each sensor as needed. While the solution to grant permissions to each of the 13 sensors individually may seem ideal, it leads to a management burden for the user and results in system inefficiency to post data to our backend (e.g., 130 POST REST API calls, rather than just 10 per second). This design prevents overprivileged apps or shared sensor views from violating the occupant's privacy (*D2.5: Privacy Controls*) while also being efficient regarding data ingestion, storage, and its use for ML-based inferences.



(a) App onboarding, login & device claiming process (b) Privacy controls (c) Data viewer & ML (d) Applet installation

Figure 2.4: Screenshots of the Mites mobile application. Our app provides occupants with controls for Mites devices in their office and the ability to view sensor data from the devices accessible to them. (a) The onboarding process for the Mites device, which includes a login to our system and claiming a Mites device to their account, and gathering user consent. (b) Various privacy controls. (c) Live data viewer allows a user to view sensor data and annotate different events for training ML models. (d) Different "Applets" that they can install, which use the sensor data from their Mites devices, some of which can have their own consent screens.

Mites Mobile Application for User Control: Ultimately, we designed the Mites system to support various stakeholders, including the occupants, the building managers, administrators, and researchers. This usability is important to ensure longer-term community participation. We considered a number of requirements and key functionalities when designing the Mites mobile app, which is the main interface used by these stakeholders. First, many users will be interacting with the Mites multimodal sensing device for the first time and will need to be onboarded with its purpose and capabilities and give their informed consent. Second, the app needs to authenticate users and allow them to search for Mites devices by name/location or by proximity. Once the occupant can identify their Mites device, e.g., in their office, they need to "claim" the device and go through the onboarding process as shown in Figure 2.4(a). Note that all claiming requests must be manually validated by a trusted admin who checks the building directory for office occupants with the person trying to claim a device. For visibility on the data collection and to enable/disable any sensors in their spaces and respond to any permission requests to share their sensor data, we provide users with granular controls as shown in Figure 2.4(b). Users can also view the sensor data from the sensors they own and have claimed, including annotating different events and specifying labels, which are then used to train ML models to provide inferences, as shown in Figure 2.4(c). In the future, users can also install "applets" that would use the sensor data on their Mites devices after they grant explicit permission to the granularity of the individual sensor, as illustrated in Figure 2.4(d).

2.4 Evaluation

We now briefly summarize the results of our evaluation. A more detailed evaluation is available in our full paper [37].

2.4.1 Experimental Setup

We deployed Mites devices in TCS Hall on the Carnegie Mellon campus, a medium-sized university building (90,000 square feet, five floors) comprising offices, shared spaces, research labs, and classrooms. While our deployment consists of 334 Mites devices, several occupants requested the devices in their offices to be turned off using our privacy controls. As a result, we evaluate and report data for 314 Mites that have been in operation for the duration of our experiment. During the deployment, while several devices required manually updating the firmware of the devices to enable bug fixes, *none* of the deployed final version of Mites devices experienced a hardware failure that required replacement, which is a testament to our iterative hardware design process. On the contrary, a significant portion of our earlier Mites prototypes experienced hardware failures.

2.4.2 System Microbenchmarks

We first evaluate the different components of the Mites system that support our design goals, namely, features that enable rich sensing (D1) and make our system scalable, reliable, and resilient (D3). The features to enable privacy and security (D2) and system monitoring (D4) goals have been elaborated upon in their own sections, and hence we do not evaluate them here further.

System Scalability and Reliability: Figure 2.5 illustrates a trace of our load balancing and fault tolerance design across a three-DML node deployment. In this experiment, we start by handing

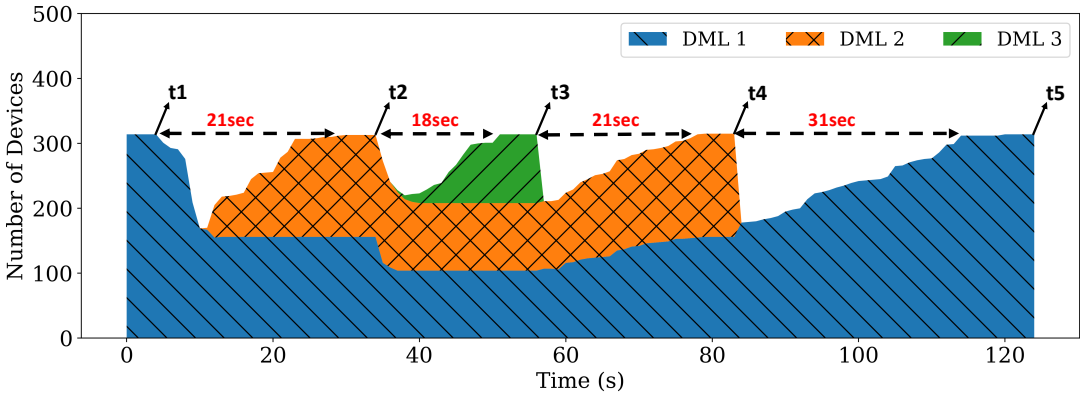
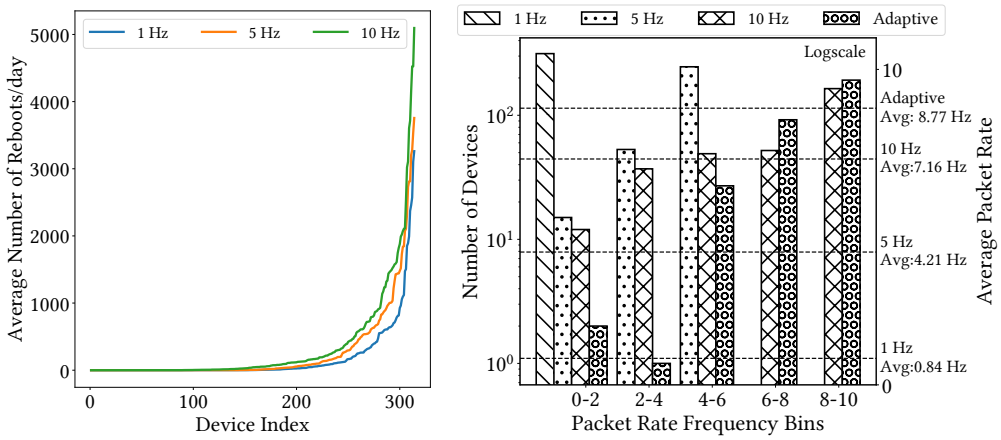


Figure 2.5: Benchmarking Load balancing and Fault tolerance of Mites system with three Device Management Layer (DML) nodes (DML-1, DML-2, DML-3) that handle a total of 314 Mites devices. The area plot shows that all devices stream data to DML-1 initially. As DML-2 and DML-3 are instantiated at time instances t1 and t2, we see the Mites devices reboot and are *load-balanced* across the available DML nodes 2 and 3. Similarly, when the DML nodes go offline (At t4, DML3 is offline, and at t5, DML2 is offline), the remaining devices reboot and connect to the available DML nodes showing fault tolerance and recovery. We also see that the time taken for all devices to recover is very short, ranging from 18 – 31 seconds.

all 314 Mites on a single DML node (DML-1). We then incrementally add a second node (DML-2) and a third node (DML-3). As the timeline shows, the sensors are load balanced across all



(a) Variability in device reboots (b) Adaptive versus static packet rate configurations

Figure 2.6: Efficacy of our adaptive packet rate scheme. On the left graph, we see that the number of reboots is much lower for lower send rates (1 Hz or 5 Hz) compared to sending data at the full 10 Hz. On the right graph, we show a histogram of the number of sensors binned into different packet rates. As seen in the histogram, our adaptive scheme has a much higher fraction of sensors in the 6 - 8 Hz and 8 - 10 Hz bins as compared to the 10 Hz static configuration while also observing significantly fewer reboots overall (not shown in the figure). The overall send rate for our adaptive scheme is 8.77 Hz (on average) compared to 7.16 Hz and 4.21 Hz for static 10 Hz and 5 Hz settings.

three nodes (at t=60 seconds / t3). To illustrate our fault tolerance design, we see that when DML3 is offline (at t=57 seconds / t3) and when DML2 is offline (at t=83 seconds / t4), all the Mites devices fail over to the only remaining node, DML1, in a short period (15 seconds).

System Resiliency – Adapting the Device Packet Rate to Network Conditions: We compare the efficacy of our adaptive send rate technique vs. statically configured packet rates (1 Hz, 5 Hz, and 10 Hz) in Figures 2.6(a) and Figure 2.6(b). Figure 2.6(a) shows that at higher send rates (10 Hz or 5 Hz), there are many more devices with a significantly higher number of reboots than at 1 Hz. Reboots are caused by the hardware watchdog resetting the device when a Mites device cannot send data for a period of time (60 seconds).

In contrast, we showcase the efficacy of our adaptive packet rate scheme in Figure 2.6(b). For this figure, we collected data for 12 days for different packet rate configurations. We collected data for each static configuration at packet send rates (1 Hz, 5 Hz, and 10 Hz) for 3 days each and another 3 days by enabling our adaptive packet rate scheme. Figure 2.6(b) shows a histogram of the number of devices in different bins of packet send rates for each configuration. We can see that our adaptive packet rate scheme performs the best with the highest fraction of sensors in the 6 - 8 Hz and 8 - 10 Hz bins and overall has the highest average packet send rate of 8.8 Hz compared to various static send rates demonstrating its effectiveness.

2.4.3 Overall System Evaluation

We evaluate the performance of our end-to-end system compared to various system optimizations mentioned in the previous chapters.

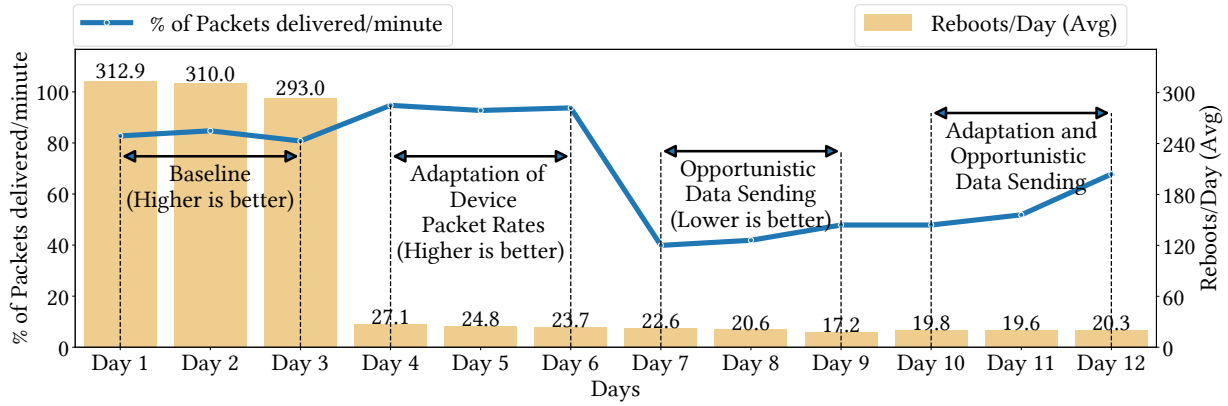


Figure 2.7: Comparison of Mites system performance with various system optimizations enabled as shown by the average % of packets delivered/minute and average reboots per day across all 314 Mites devices in our deployment, gathered over 12 days. We observe that the adaptive packet rate optimization (Days 4 - 6) results in more packets delivered compared (and lower reboots ∼25 across the deployment) to a statically configured rate of 10 Hz for all devices (baseline, Days 1 - 3). When we enable the opportunistic data send optimization (Days 7 - 9), we see that both the number of packets delivered and average reboots are lower since data is only sent from the Mites devices during periods of significant change from the 'Ambient Background.' Finally, when we enable both optimizations (Days 10 - 12), the average % packets delivered (reboots comparable) is generally higher and depends on the activities happening.

Figure 2.7 plots the percentage of packets delivered/minute and the average reboots/day from the 314 Mites devices to show the efficacy of our adaptive packet rate scheme (Section 2.3.3) and opportunistic data sending (Section 2.3.3) optimizations. We performed this evaluation for 12 days, ensuring no other external factors (network reboot, user requests, etc.) affected the data collection. From Day 1 to Day 3, we configured all devices to send packets at 10 Hz (baseline). Given this is a real-world deployment, we limited our data collection to 12 days primarily to prevent any service interruptions to real-world users. From Day 4 to Day 6, we enabled the adaptive packet rate scheme. From Day 7 to Day 9, we enable opportunistic data sending. Finally, from Day 10 to Day 12, both adaptive packet rate and opportunistic data sending are enabled. We observe that enabling adaptive packet rate increases the percentage of packets delivered per minute (mean: 94% packets/min, standard deviation (SD): 1) while reducing the average reboots/day (∼27) compared to the baseline of sending data at 10 Hz (mean: 83% packets/min, SD: 2) and 300 average reboots/day. Since the Mites system adaptively switched packet rates to maximize packet delivery rates, we can achieve higher packet rates. Similarly, only enabling opportunistic data sends results in a lower percentage of packets (mean: 43% packets/min, SD: 4.16). This is because devices send data (at 10 Hz) only when a likely event is detected by comparing the sensor data to the ambient background. Finally, when we enable both optimizations, we see a

higher packet delivery rate (mean: 56% packets/min, SD: 10.58) whenever an event occurs. This shows that devices send data without packet loss whenever an activity happens.

Overall, our results demonstrate the efficacy of different design optimizations (adaptive packet rate and opportunistic data send) to obtain featurized sensor data at a high rate reliably from our entire network of Mites devices while ensuring that our end-to-end system consumes fewer resources.

2.5 Conclusion

Real-world deployment of a large-scale sensing system is challenging due to the lack of design and architectural support for high-fidelity sensing. In addition, existing sensing systems are limited as they are geared toward specific, vertically integrated applications. Such methods are unreliable, have limited functionality for users, and have few, if any, primitives for privacy and security. We have presented Mites, a scalable end–to–end hardware-software stack for supporting and managing high-fidelity distributed general-purpose sensing in buildings. Specifically, the goals of the Mites system are to support ubiquitous large-scale management and operation of infrastructure in a way that is extensible, easy to use, and provides security while maintaining occupant privacy. We deployed and evaluated our Mites system in TCS Hall, a five-floor mixed-use office building on the Carnegie Mellon University campus. We share our key insights and sincerely believe they will impact future researchers and practitioners attempting to design and deploy a similar general-purpose sensing system for buildings.

Chapter 3

End-to-End General Purpose Machine Learning System for Smart Buildings

The previous chapter showed the design and deployment of the hardware-software stack of the general-purpose sensing system with key system design primitives to support privacy controls. In addition to the hardware and system stack, a core component of the general-purpose sensing system is to support Machine Learning (ML) tasks to enable diverse IoT application use cases such as Activities of Daily Living (ADLs) [43, 64] or Context-driven Applications [8, 24]. For example, detecting activities and events in smart environments requires machine learning on various IoT sensor data sources. However, there are ongoing concerns about the potential privacy implications of ML tasks on the breadth of the data collected by such ubiquitous IoT device deployment [45, 124, 223].

The holy grail today for ML systems is to "train" a generalized ML model once and then "serve" or deploy it to make predictions. In fact, several general-purpose ML serving-only systems exist, [1, 19, 59, 230, 232] including those for IoT-relevant audio data [131] or image sources.Other approaches propose Programming by Demonstration (*PbD*) [130, 152] to enable users to train personalized models 'in-situ' in a specific IoT environment. One common pitfall of such system approaches is that they primarily focus on system optimization strategies to reduce latency or improve accuracy during the ML lifecycle and fail to take into consideration the primitives required for IoT settings. First, these systems are optimized for specific and often dedicated hardware resources and do not provide the privacy controls to run their ML computation on edge compute hub nor isolate the ML computation for sensor streams based on different users. Second, each IoT deployment/environment is different, requiring 'in-situ' (re-)training based on the sensors and events in that setting. For example, multi-modal sensor [130, 152] or audio-based context recognition [73, 131] is affected by the physical space characteristics and changes to the ambient environment. Third, accurate pre-trained models, such as ImageNet or YoLo, need a significant amount of labeled training data and computation resources, which is atypical in IoT scenarios. Finally, these systems are optimized for specific and often dedicated hardware resources and do not adapt to changes in resource availability, for example, an edge compute hub (for privacy) as compared to a server on the cloud.

In this chapter of the thesis, we describe our work on designing and implementing a MLIoT, an end-to-end machine learning system tailored for IoT use cases, supporting the entire lifecycle

of initial training, serving, and retraining processes. MLIoT adapts to (heterogeneous) compute resources by performing device selection based on user-expressed preferences (policies for edge computing, accuracy) and benchmarking device capabilities. MLIoT adapts to different data sources and tasks by automatically training, optimizing, and serving models based on expressive preferences (policies). MLIoT adapts to changes to the IoT environment or compute resources by re-training, and updating the served models on the fly, while maintaining accuracy and performance. MLIoT performs these adaptations dynamically for multiple tasks, each with their own training-serving pipelines and requirements.

The rest of this chapter describes MLIoT at a high level. Additional details on our completed work in this space is presented in a full paper on this topic [36].

3.1 Challenges

To motivate the challenges unique to IoT settings, we present several examples of IoT applications and their data sources. We then identify the unique challenges for ML systems in IoT settings, which existing general purpose training [63, 207, 230] and serving [1, 59] systems fall short on.

3.1.1 IoT Application Workloads

Activity Recognition using Multi-Modal sensors: There is a rich history of activity and event detection in IoT-enabled environments, using non-intrusive ambient sensors [98, 120, 130, 152], or direct instrumentation [40, 224], and even in-direct sensing [56, 94, 201]. These systems use labeled data from sensors such as Accelerometers, Gyroscopes, Microphones, Temperature, and Humidity to explicitly train ML models for specific events and test their accuracy on live data. Recent systems[130, 152] incorporate as many as 13 different hardware sensors, extracting over 2,000 features on a single package. They propose a Programming-by-Demonstration (PbD) approach to train a single ML model to predict the occurrence of various events. Their interface allows end users to add more training data and retrain the ML model if desired. Given the diversity of IoT-relevant hardware sensors present on this platform, we collect and use multiple datasets from the Mites to evaluate MLIoT, as well as compare our accuracy and performance to this state-of-the-art system.

Audio Based Activity Recognition: Given that many activities have an audio signature (e.g., appliances running, faucets being turned on), Ubicoustics proposes using audio only for activity detection, particularly by building a *generalized* a pre-trained deep model [131]. Their system collects audio, processes it into VGG-16 network with 6144 features, and then trains a large DNN. We collect a set of audio benchmarks and compare MLIoT with Ubicoustics.

Object Recognition using Image Data: Several emerging IoT applications apply computer vision algorithms on images, to detect objects, adding labels and bounding boxes[190, 191]. To evaluate MLIoT for image data, we use the popular MNIST dataset which has labelled images for handwritten digits [134].

3.1.2 Challenges for ML systems in IoT settings

Motivated by the above IoT application and workloads, we identify key challenges for ML Systems geared towards them.

Adapting to Different IoT Application requirements: IoT applications requirements can differ substantially. Deep learning based computer vision applications often need hardware accelerators with significant memory[190]. Activity recognition, using classical models, often need to run on *edge* gateways such as a Rasbperry PI [185] or a SmartThings Hub, [202] with modest compute resources for faster latency and to alleviate privacy concerns. Developers of these applications themselves may have different requirements such as accuracy, latency, cost (cloud vs local inferences), models to use, and even different priorities between their different IoT tasks. MLIoT provides expressive policy-driven mechanisms to balance user requirements with the available resources.

Adapting to Device Capabilities and Resource Availability: The inherent heterogeneity in individual device capabilities (CPU complexity, number of cores, memory, bandwidth, accelerators) affect the performance (training and serving) of ML algorithms. It is essential to thus estimate the comparative performance of the devices available, which MLIoT achieves by benchmarking devices in different conditions and using that information for device selection. Furthermore, the resources available on individual devices also change as different IoT ML tasks run on them, each with their own requirements. MLIoT provides load balancing and various dynamic adaptation mechanisms such as changing the models served, to meet these requirements.

Adapting to Changes in Environmental Context: Existing ML systems [1, 19, 59, 230, 232] often rely on generalized ML models, which are infeasible in IoT settings since each environment is unique and models need to be contextualized to that environment. IoT systems are also affected by changes in the ambient environment, both temporary and permanent (e.g., changes to the layout of a home affect audio-based sensing). MLIoT enables both the initial training and retraining and re-optimizations of the models based on user-driven corrective feedback.

3.2 Background

3.2.1 Large-scale ML Serving Systems

There are several general-purpose predictions serving systems from the industry and academia which aim to facilitate model deployment [1, 59, 60, 131, 225, 232]. These systems place the trained models in containers and optimize model inference requests. Clipper [59] aims to deploy pre-trained ML models in containers and optimize serving performance using request batching and caching to reduce latency. It also employs user feedback to select and combine the output of one or more deployed models to improve accuracy. Inferline [60] provisions and executes prediction pipelines subject to latency constraints, leveraging adaptive batching and autoscaling to reduce latency. TensorFlow Serving [1], a commercial grade serving system, is designed to deploy models as TensorFlow pipelines [86] which are executed in black box containers. Other

serving systems have focused on applications content recommendation systems [232], speech recognition [135] and activity recognition [131] all of which have highly customized, application specific models. Several commercial systems targeted towards IoT also exist such as Amazon AWS IoT greengrass [19], Google Cloud Vision AI [88], Amazon Rekognition [15] or Azure IoT [149]. Some of them leverage edge and cloud resources to serve models with low latency, and to save costs. These commercial systems are vertically integrated, focusing on serving predictions from a single model or framework, or specific hardware only. These systems fail to address one or more key requirements for IoT scenarios: focus on serving static pre-trained models, no adaptation to environmental changes, do not support heterogeneous devices, or have limited, if any, support for policies.

3.2.2 Large-scale ML Training Systems

Most of the training focused systems [51, 63, 207, 230] optimize for deep neural network models with many hyperparameters, which is very resource and time intensive. Project Adam [51] investigates distributed training based on available resources while Helix [230] and KeystoneML [207] use various techniques to optimizes the ML training workflow. Commercial systems such as Google Vizier [84], similarly optimize DNNs, focusing on a variety of techniques to 'tune' the network parameters to improve accuracy and performance. Google AutoML [87] views learning to build a network itself as a machine learning problem, applying techniques such as reinforcement learning for the task. These systems are geared towards producing high-quality and complex deep networks for domains such as object recognition, and translation. These training only systems assume large corpus of training data for an expensive, infrequent, training tasks and optimize for efficient distributed model training. In contrast, in IoT scenarios the available data is sparse to train complex models, the trained models need to be specific to the sensor sources, environmental context, application requirements and thus need more closely coupled (re-)training and serving systems.

3.2.3 ML Training/Serving Hybrid Systems

Several recent efforts aim to simplify ML development through a general-purpose machine learning system with both training and serving of models [12, 19, 28, 58, 60, 88, 109, 110, 138, 239, 240]. Some of these systems that share similar goals of MLIoT are the end-to-end "ML Platforms" that run at commercial settings. Systems such as Uber's Michelangelo ML [110] and Facebook's FBLearner Flow [109] serve as ML-as-a-service platform which is optimized for their internal use cases. Uber's Michelangelo ML is optimized for their real-time requirements, allowing production models to use features extracted from streams of live data. FBLearner Flow allows reusable ML workflow that allows ML models to be modified and reused in different products. On the other hand, Google's TFX [28], provides Tensorflow-based [86] toolkits for data preparation, and periodic model evaluation to improve performance and reliability and extends TensorFlow Serving [1] to serve the models with TensorFlow-based learners. Such systems generally run on the cloud incurring a higher cost for better workload environments and restrict users to a specific set of algorithms or libraries, so users are on their own when they step outside

these boundaries. They are not designed to adapt to unpredictable operational environments. Ultimately, many of these systems do not address the challenges specific to IoT applications.

3.3 System Design and Architecture

The overall architecture of MLIoT is illustrated in Figure 3.1. It comprises two logical components: the Device Selection Layer (DSL) and the Training-Serving Layer (TSL).

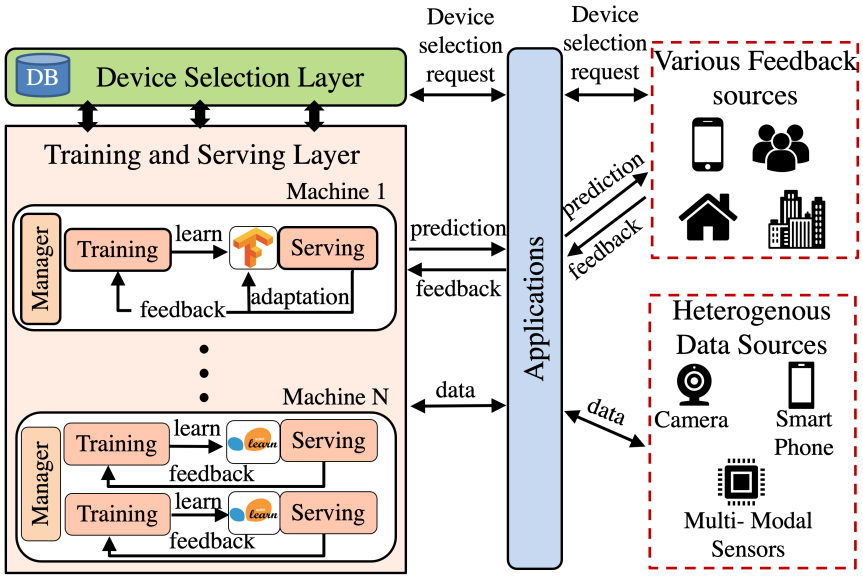


Figure 3.1: Overall System Architecture of MLIoT. The system consists of two components: the Device Selection Layer (DSL), managing devices and scheduling ML tasks, and the Training-Serving Layer (TSL), which instantiates workers for each ML task.

The DSL has several key roles. It serves as the central authority that manages all the devices or compute resources available to it for scheduling ML tasks. It is also responsible for handling any new request for ML tasks and based on the policy specified by the IoT application, selecting the appropriate resource to serve that task. The DSL also maintains configuration data for each training-serving task, which includes the application training data, the trained ML models, and several other parameters for serving those models. This functionality is important since the same task can be interrupted or restarted and can resume serving from where it left off using the state available at the DSL, including running on a different device.

The Training-Serving Layer (TSL), is responsible for instantiating the Training-Serving Workers (TSW) for each ML task sent to it. The TSL is also responsible for allocating and managing resources (CPU and memory limits) to individual TSWs. The training-serving workers are responsible for training models, optimizing them, and selecting the set of models for serving based on the specified policies. TSWs are also responsible for keeping track of performance and adapting the models to changes to the resource availability on the device they are running on, or on receiving corrective feedback. Individual TSWs report back metrics like model performance,

CPU and memory usage, etc to the DSL which has a holistic view of all the tasks it is managing and the devices they run on. We describe the TSL and the functionality provided by the TSWs in further detail in §3.3.2.

3.3.1 Device Selection Layer (DSL)

IoT application tasks comprise of training and serving ML models. How well these tasks run depends on the device capabilities such as the number of cores available for parallel execution, the amount of memory available, the relative performance of the cores (x86 vs ARM vs an accelerator), etc. Furthermore, IoT applications themselves may have different requirements. For example, an application that senses audio at home may need the ML predictions to be done locally on an in-home hub like a Raspberry-PI or a Samsung Hub. An intrusion detection application [71] may require low latency predictions, while an application that detects falls for the elderly requires high accuracy predictions to reduce false positives. Activity detection scenarios [130, 131] similarly need to be responsive, implying fast training times, to reduce user annoyance of having to wait. Ultimately, the DSL needs to holistically manage and monitor the compute resources at its disposal and schedule IoT tasks on different devices.

Benchmarking Device Performance: There are numerous devices and platforms that MLIoT can run on, often with different characteristics. These include inexpensive platforms like a Raspberry Pi, or Hardware accelerators for ML such as Google EdgeTPU [89], Nvidia Jetson [162] (or) Intel Neural compute [102]. Alternatively, MLIoT can also run on traditional VMs in the cloud on Google's Cloud Platform [88], or Amazon AWS [15]. To characterize their *relative* performance, MLIoT needs to benchmark each device. Since the IoT application workloads are not known apriori, we collect and use three different *representative* IoT datasets described in Section §3.1.1 - audio based activity recognition, a multi-modal sensor based activity recognition, and an image recognition application. Our insight is that the relative performance of devices on these representative datasets can be used for device selection of new IoT tasks. For these benchmarks, we train and serve a set of classical ML models as well as deep learning models (as applicable), on each device measuring several key metrics:

- **CPU and Memory Utilization:** We collect the average CPU and memory utilization over the entire benchmark execution and normalize the values between 0 and 1.
- **Prediction Latency:** We measure the average time taken by a training serving worker running on a device to receive a prediction request and respond to it for each benchmark.
- **Training Time and Accuracy:** We measure the training time and accuracy for a set of models, for each benchmark.

Notably, we collect these metrics when instantiating a Training-Serving Layer on each device. The overhead of collecting this data for device selection is low and to ensure that it does interfere with any other tasks, we reserve a fraction of the compute resources (10%) for benchmarking only.

Device Selection Policy: The DSL considers several aspects while scheduling tasks on devices. It uses the *benchmark metrics* from devices as mentioned above, as well as *device metadata* such

as the number of cores, memory, as well as the presence of accelerators. Since the available resources on a device change based on other co-located tasks on it, the DSL captures *run time metrics* periodically such as the CPU load (C), Memory (M) and Load Average (LA). This information is provided by the TSL running on each device, to the DSL periodically (every 1s). For these metrics, we calculate an Exponential Weighted Moving Average (EWMA) for a window size of 10 samples to reduce transient noise. Based on the benchmark metrics, the runtime metrics, and the device metadata the DSL exposes several policies for IoT developers to specify their application requirement. These policies are either *Static*, based on static values such as CPU core count etc., or *Dynamic* based on metrics that change such as the CPU load.

- **Static Policies:**
  - *GPU-CPU:* Use a machine with GPU (or) a CPU.
  - *Max-Min:* Select devices with the maximum or minimum of number of CPU cores, Memory or Load.
  - *Locality:* Select devices based on locality, such as an trusted edge device for privacy concerns or optimizing latency.
  - *And-Or:* Select devices based on a logical combination of different metrics mentioned above.
- **Dynamic Policies:**
  - *Threshold:* Select devices based on specifying *Atleast, Atmost or Equal* threshold conditions. e.g. Latency Atmost 40ms.
  - *Best Effort:* When no application requirements are specified, the best effort policy chooses the device with the most available resources across the runtime and benchmarking metrics.

The DSL compares the metrics it collects from different devices, with the policy requirements of each IoT application, evaluating them in order of arrival. If the application policy can be satisfied, the appropriate device is selected. However, in case the available resources cannot meet the policy requirements of an IoT application, the DSL sends a negative response back to the application.

Load balancing Training and Serving workers: In many cases, the DSL may be able to choose from multiple devices to meet individual IoT application requirements. In these scenarios, especially with an increasing number of concurrent applications, it is important to efficiently load balance tasks across devices. The DSL uses real-time metrics (e.g. CPU and Memory usage, and Load) as inputs to a dynamic load balancing algorithm based on resource weights [205] to address this challenge.

More formally our load balancing algorithm works as follows. Let's assume there are 'n' devices in a MLIoT deployment. The TSL on each of these 'n' devices reports its current system performance metrics periodically (1s). We calculate the load state using the following formula:

$$L_i = w_{cpui} \times cpu\_usage_i + w_{memi} \times memory\_usage_i + w_{loadAvgi} \times loadAvg_i \tag{3.1}$$

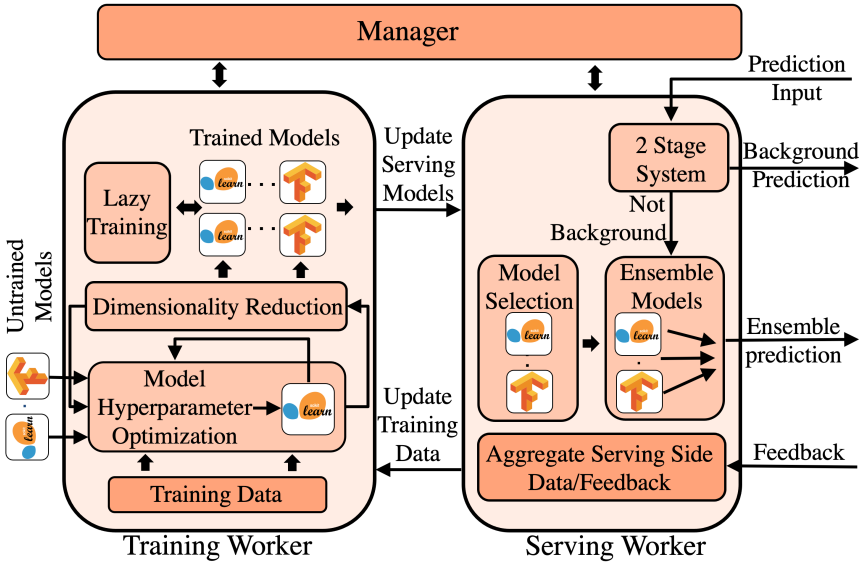


Figure 3.2: Overall architecture of the Training Worker (TW) and Serving Worker (SW). The TW is responsible for training models in parallel and optimizing hyperparameters using the input training data. Based on different policies, an ensemble of models is selected and sent to the SW. The SW uses the ensemble to make predictions using certainty estimation while aggregating serving side data for feedback from the user.

- L_i is the Load state of i*th* Training and Serving Layer.
- cpu_usage_i, w_{cpui} is the percentage CPU utilization with its corresponding weight.
- $memory\_usage_i, w_{memi}$ is the percentage Memory utilization with its corresponding weight.
- $loadAvg_i, w_{loadAvgi}$ is the system average load with its corresponding weight.

We calculate the weight dynamically at each fixed cycle to calculate load state precisely. To get the weights we find the minimum value of the metric and divide it by the metric value of the current machine as below:

$$w_{cpui} = \frac{min(cpu\_usage_1, cpu\_usage_2, ...cpu\_usage_n)}{cpu\_usage_i} \tag{3.2}$$

The DSL then selects the device with the least load for the particular IoT application.

3.3.2 Training-Serving Layer (TSL)

The Training-Serving Layer (TSL) runs on each device in a MLIoT deployment. The TSL spawns a separate manager process to create, monitor and manage a pair of Training Workers (TWs) and Serving Workers (SWs) for each IoT application as shown in Figure 3.2. The TSL uses Linux CGroups (control groups) [42] to allocate and enforce the resource usage for each Training and Serving worker. The Training Worker is responsible for the initial model training using labeled data, as well as retraining models when corrective feedback is provided by users to improve accuracy. The Serving Worker obtains the trained models from the TW, creates a model

ensemble and performs online predictions. The SW also collects user feedback on the ensemble-based predictions and forwards them to the TW. This close coupling of the TW and the SW for each IoT application is critical to improve accuracy and performance over time, and adapt to dynamic IoT environments. We discuss the Training Worker in §3.3.2 and Serving Worker in §3.3.2.

Training Worker (TW) A key goal of a TW is to streamline model training to provide high-quality ML models that are tailored to the specific IoT task. The TW uses generic ML model definitions to enable extensibility and allow use of multiple common ML frameworks and their algorithms. The TW selects models and applies various optimizations (discussed below) based on the device resources for faster training, and low latency serving.

TW Model Selection: Similar to Device Selection policies in Section§3.3.1 the training worker selects models based on application policy specification around metrics such as model accuracy and latency, as well as performance metrics such as CPU and memory usage. The goal with TW model selection is to create an ensemble of models to send to the Serving Worker. An example policy can choose three models with the lowest latency, to send to a SW. The TW handles all training related tasks, performing model selection and training, and then continuous model adaptation over time.

Serving Worker (SW) The Serving Worker performs online predictions using models received from the Training Worker, under the IoT application's policy constraints. The SW also provides a certainty estimate for each prediction based on the models included in the ensemble, for higher accuracy. It also supports a two-stage serving system that is specifically optimized for IoT environments, to improve accuracy due to environmental noise and reduce latency. The SW is also responsible for collecting corrective feedback to send to the TW for model adaptation.

Two-Stage Serving: Most continuous serving IoT scenarios are dominated by periods where no interesting events happens. For example, for activity detection, for 8 hours a day an occupant is likely asleep and in a specific room. In these cases, the IoT ML pipelines are essentially detecting ambient "background". Current IoT systems[130, 152] thus explicitly train for large amounts of background as one of the classes. Furthermore, transient "noise" in the sensor data, is also often misclassified as one of the trained events. Ultimately, this leads to higher latency since all models are evaluated for background and noise events, and lower accuracy when there is noise.

In MLIoT we propose a two-stage serving system. The *First Stage* is a binary classifier with two classes, namely the "Background" and all the other classes of interest for the IoT application. We empirically evaluate several ML models for the first stage and report results for different IoT workloads. We show that Logistic Regression (LR) [146] works best as the first state binary classifier, balancing accuracy and latency. The *Second Stage* uses the model ensemble obtained from the Training Worker and uses it for serving predictions. MLIoT's Two-Stage Serving reduces overall latency when events like background or noise occur.

Ensemble Based Model Prediction: After model selection, the SW uses the final set of models for weighted ensemble prediction, determining model weights based on validation accuracy.

Let the final list of models selected be $m_1, m_2, ..m_n$. We calculate the final ensemble based prediction as:

$$Prediction = argmax_c(\sum_{i=1}^{n} w_i(if\ f(x)_{m_i} = c)) \tag{3.3}$$

where $f(x)_{m_i}$ is the prediction made by model m_i for input x and w_i is the weight for m_i. The static weights $w_1, w_2, ..w_n$ for the models $m_1, m_2, ..m_n$ are calculated as:

$$w_i = e^{x_i}(\sum_{j=1}^{n} e^{x_j})^{-1} \tag{3.4}$$

where $x_1, ..x_n$ is the validation accuracy of model $m_1, ..m_n$.

Prediction Certainty Estimation: In several IoT use cases, especially those that actuate devices or send notifications based on ML predictions, knowing the confidence of the predictions can be very useful. To support this, MLIoT provides ensemble-based certainty estimates for each predictionby taking the weighted average of the probability of each class for each model.

$$certainties = (\sum_{i=1}^{n} w_i f_{m_i,c_1}(x), \sum_{i=1}^{n} w_i f_{m_i,c_2}(x), ..., \sum_{i=1}^{n} w_i f_{m_i,c_m}(x)) \tag{3.5}$$

where $f_{m_i,c_1}(x)$ is the probability that the model m_i predicts for the class c_j on the input x. The estimates are calculated by taking the square magnitude.

Model Adaptation: IoT environments are subject to changes in the ambient environment that affect model accuracy. In addition, users may want to update models with additional training data, including adding a new class or give corrective feedback for an incorrect prediction. MLIoT, performs model adaptation by sending this data to the Training Worker to retrain models. However, naively re-training all models with HPO and DR is expensive and potentially time-consuming. To overcome this challenge, we re-tune each of the models in the ensemble with new data. This significantly reduces the training time and is computationally less expensive. This feedback-based model adaptation allows us to account for changes to the environment and provide better results than using static models.

3.4 Evaluation

We now briefly summarize the results of our evaluation. A more detailed evaluation is available in our full paper [36].

3.4.1 Experimental Setup

Machine Learning Models: We choose several popular ML algorithms and frameworks to show that MLIoT is extensible to use a variety of models. To evaluate MLIoT we select five traditional

models (KNN, Ridge Regression, RandomForest, SVM-Linear, SVM-RBF) and two Neural Networks (MLP & XGboost). We integrated the implementations of these algorithms from SKLearn [173], TensorFlow [2], and PyTorch [172] into MLIoT.

IoT Benchmarks: We chose MNIST, a popular image-based object recognition dataset for the first benchmark[134]. For activity recognition based on rich multi-modal sensors, we collected data using the Mites devices [37] platform with 13 hardware sensors for a set of 16 common residential activities (classes). We use the same features proposed in Ubicoustics[131] wherein audio data from non-overlapping frames ($960ms$ each) is converted to a spectral representation to provide log-Mel spectrogram patches of 96×64 bins that form the input to all classifiers [97].

Table 3.1: Hardware platforms used in our experiments

Device	Type	Processors	Memory	Average RTT	Local /Cloud
M1	Raspberry-Pi 4	4 x ARM A72	4GB	~3ms	Local
M2	Intel NUC	4 x i5-5250U	8GB	~14ms	Local
M3	Virtual Machine	2 core	4GB	~128ms	Cloud
M4	Virtual Machine	8 core	16GB	~64ms	Cloud
M5	Virtual Machine	16 core	32GB	~84ms	Cloud

Testbed Hardware Platforms: To evaluate MLIoT we set up a five machine testbed (M1-M5 in Table 3.1) with different hardware configurations, resources, and network latencies (Average RTTs) representing both local (edge) and cloud compute resources. **M1** is an inexpensive Raspberry Pi4 (RPI4) [185] with 4 Cores, 4GB RAM, and 3ms RTT, augmented with the Google's Coral Hardware Accelerator[89] for testing our Device Selection Layer (Section §3.3.1). **M2** is an Intel NUC desktop with 4 cores, 8GB RAM, and a 14ms RTT . Notably, M1and M2are "local" devices since they are in the same LAN as the client, behind a NAT. **M3, M4, M5** are Virtual Machines (VMs) running on different physical servers. **M3** has 2 Cores, 4GB RAM and 129 ms RTT, **M4** has 8 Cores, 16GB RAM and 64ms RTT and **M5** has 16 Cores, 32GB RAM and 84ms RTT. Our test client machine, which executes all IoT Application traces is a Mac with a dual-core i7 processor and 16 GB RAM.

3.4.2 System Adaptation and Scaling

To assess the overall performance and adaptability of MLIoT, we evaluate MLIoT with different IoT application workloads and policy requirements Section §3.4.2. We then compare MLIoT with other academic and commercial ML systems.

System Adaptation: An overarching goal of MLIoT is to effectively schedule different IoT applications, that are concurrent, on different devices given their individual policy requirements. We categorize three typical application policy requirements: (a) strictly latency bound, with specified thresholds; (b) those that require "edge" devices in their local network for low latency and for privacy (e.g. speech recognition or activity recognition); and (c) those with more complex

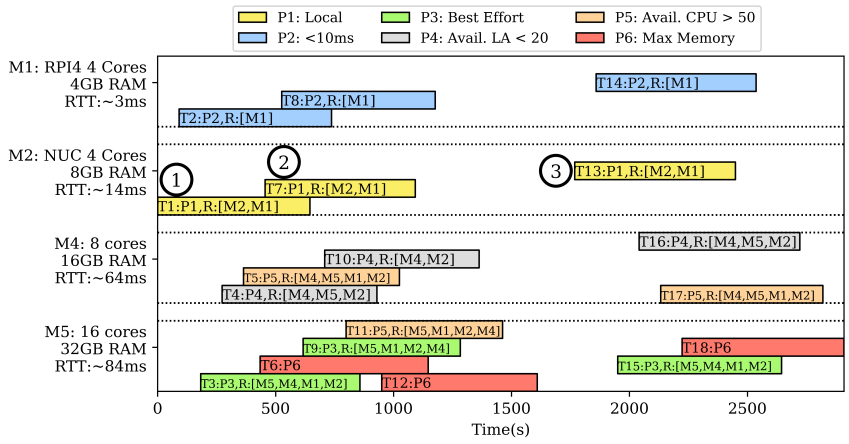


Figure 3.3: Timeline of the six application traces (Ts) where all the traces have diverse policy requirements. Traces have policy constraints on latency, resource and best effort.

requirements as a function of latency, and resource usages and accuracy. For this evaluation, we create a set of machine learning tasks using the six IoT application workload traces described in Section §3.4.1 , each with different representative policies attached (as discussed in Section 3.3.1). We use four devices (M1, M2, M4, M5) with different RTT values and configurations emulating an example MLIoT deployment (details in Table 3.1).

Figure 3.3 shows application traces with further diverse policy requirements based on on locality, compute resources, and latency metrics. These policy requirements include select devices with available CPU percentage (e.g. $> 50\%$) or a device with maximum available memory. At (1), we see that MLIoT selects M2 over M1 for T1:P1 to satisfy the locality constraint without a latency requirement. T4:P4 is executed on M4 as the load average for M4 was less than 20 and similarly, T5:P5 is run on M4 as the available CPU resources was more than 50. This is indicated by the rank list for T4:P4 as R = [M4,M5 and M2] and T5:P5 as R = [M4>M5>M1>M2]. In the 2nd iteration (2), we see that T11:P5 is run on M5 instead of M4 based on the available resources. Traces (T6:P6, T12:P6, T18:P6) with policy of picking a device with maximum available memory is always executed on M5 given its 32GB RAM configuration.

3.5 Conclusion

We design and implement MLIoT, an end-to-end Machine Learning System tailored towards supporting the entire lifecycle of IoT applications from training, and efficient serving to re-training based on user feedback. MLIoT integrates multiple distributed components and optimization techniques making our system adaptive, and handling the diversity of IoT use cases. MLIoT provides a flexible policy-driven selection of hardware platforms, ML models, and various optimizations for training and serving tasks. Our evaluations of MLIoT on several hardware devices, and for a set of expressive IoT benchmarks, show that MLIoT is able to service different policy objectives, balancing load across devices while maintaining accuracy and latency.

Chapter 4

Raising the Abstraction for Activity Inferences using Context Sensing Framework

Thus far, this proposal has focused on addressing the privacy risks associated with a general-purpose sensing platform by enabling features for data minimization, such as on-device edge-featurization of raw data, enabling the conversion of the low-level featurized data to high-level inferences. The Mites system showed the development of robust sensing hardware with features to enable edge processing all on the Mites device such that there is no way for raw data to ever leave the device, in addition to features for sensor control and metadata obfuscation. With MLIoT, we generate virtual sensors, converting the featurized data to high-level activity inferences using predictions from ML models. Overall, the ML models could use the featurized data from sensors on the Mites devices, such as an accelerometer and microphone, to train and predict activities such as *typing*, *talking*, or *mouse click*, converting the low-level featurized data to high-level inferences.

While these high-level activity inferences provide valuable insights for various applications, such as productivity or health monitoring, they can also be privacy-invasive as the application can collect sensitive inferences about individuals without their knowledge or consent. For example, in a productivity tracking application, if an ML model can infer with high accuracy individual activities such as *typing* or *talking*, this information could be used to infer other sensitive information, such as their work schedule or when they are likely to be away from home. In such cases, understanding the context of activity inferences can help to improve privacy for such applications. For example, if the productivity tracking application considers contextual information, like if the activities *typing* or *talking* is sent to the application only while the individual is in the context of *OfficeWork*. This approach makes the activity inference more accurate while protecting the individual's privacy. Furthermore, the contextual information provides the users with more control over their activity inferences by allowing them to specify which contextual information they are comfortable sharing based on the application.

A key challenge, however, is that human activity patterns are complex in nature and often require contextual information about the activity to be useful for downstream applications. For example, a wellness application that assesses an individual's productivity requires contextual in-

formation about an activity. An activity such as *talking* may or may not indicate if the individual is being productive, depending on whether it is happening in a context denoting office work or a different context of having a meal. Moreover, current HAR-based approaches primarily focus on using the sensor data from smart devices (such as phones, watches, or ambient sensors) to accurately infer human activities over timescales in the order of seconds (such as *Activity (A): talking, A: typing, A: jumping,* etc.), rather than extracting more semantically meaningful contexts (such as *Context (C): OfficeWork* or *C: Exercising*). Thus, understanding higher-level and semantically meaningful contexts of daily activities are crucial to support applications such as tracking productivity or wellness.

In this chapter of the thesis, we aim to further raise the abstraction for activity inferences. We propose TAO, a hybrid system that leverages OWL-based ontologies [90] and a temporal autoencoder-based clustering algorithm to detect semantically meaningful and richer contexts from complex real-world activity patterns (e.g., sequential and parallel). We design and build our custom ontology based on prior work [148, 220] and extend it to model various real-world activity patterns as complex activity relationships used to infer a wide range of contexts. To enable an extensible design, we model our ontology using SPARQL queries[200], allowing us to easily represent complex activity relationships and infer them to context definitions. We build a custom unsupervised clustering algorithm to model temporal contexts to identify recurring activity patterns and label these patterns using our ontology. TAO's hybrid approach handles sequential and parallel activities, accurately converting them into a rich set of contexts.

The rest of this chapter describes TAO at a high level. Additional details on our completed work in this space are presented in a full paper on this topic.

4.1 Challenges

Our overarching goal for TAO is to accurately translate fine-grained human activity patterns obtained from different activity recognition systems to meaningful contextual information that allows us to understand the user's behavior better. This contextual information can support various downstream applications in domains such as health, wellness, and human-computer interaction, including those that aid individuals in improving their quality of life, particularly those with cognitive or physical impairments. We motivate several potential use cases to illustrate the benefits of an accurate context-detection system.

4.1.1 Healthcare-based Applications

Context recognition in the healthcare domain can be used to monitor and support individuals with chronic conditions. For example, a system based on context recognition can help interpret a person's activities of daily living by looking at specific sequences of activities, such as *A: opening a medication box* followed by *A: drinking water* to detect adherence to medication. Similarly, inferring that an Alzheimer's patient left the kitchen to answer an incoming call (*A: phone ring*) but failed to resume lunch (*A: eating*) while in the context (*C: Eating a Meal*) afterward can help with memory augmentation tools. Such contextual information could provide reminders or prompts to help the person maintain a healthy routine and alert caregivers if they are not engaging

in activities necessary for their well-being while reducing the number of false alarms. Similarly, wearable fitness trackers and medical devices monitor a person's current activity and vital signs, such as heart rate and blood pressure. These vital signs can vary depending on the patient's activity level and other factors. With context recognition, a system can understand the context in which the vital signs were measured (e.g., C: Exercising or C: OfficeWork), allowing for more accurate attribution and possible notifications and interventions. For example, a change in the vital signs in one context may be acceptable while being a sign of stress in another. We believe that understanding such contexts would be very useful in providing more targeted and actionable interventions for addressing mental health issues [50].

4.1.2 Smart Building Applications

In smart building scenarios, a potential application of context recognition is to improve productivity and reduce stress. For example, suggesting tasks or actions based on the person's current context, providing reminders and prompts to help them stay on track, and automatically adjusting the environment to create a more conducive work environment. For example, if the system detects that an office occupant is in the same context for long hours, such as *C: OfficeWork*, while engaging in multiple activities, such as *A: talking* or *A: typing*, it can nudge the user to take a break to improve their productivity. Moreover, such a system in office buildings could improve energy efficiency and comfort for building occupants by changing environmental parameters based on context. For example, if the context is identified as *C: In a meeting*, the HVAC system of the room can be configured to allow more airflow into the office. Similarly, when the occupant is in the context of *C: Working* for some time, denoting focused work, their status can be set to busy automatically on software such as Slack to prevent interruptions. In contrast, when the context detected is *C: Taking a Break*, they can be marked as available to allow for impromptu social interactions. Note that a user in each of the example contexts above could be doing a wide variety of low-level interleaved activities (e.g., A: talking, A: using the PC, A: pacing, A: typing, etc.).

4.2 Background

Several prior works have proposed different modeling approaches for context reasoning architectures with a primary focus on activity recognition. Existing research in this space falls into three broad categories: (a) knowledge-driven approaches that focus solely on ontological reasoning; (b) data-driven approaches that rely on modeling temporal activity patterns, and; (c) approaches that combine knowledge and data-driven approaches. We refer the reader to [31, 175, 179] for an extensive survey on this topic and also compare and contrast prior works with our TAO system.

Ontology-driven Approaches: Researchers have proposed numerous ontology-driven approaches to model the semantics of low-level activity information and recognize user context [10, 11, 47, 48, 147, 148, 220]. Most ontological approaches use knowledge representation languages such as OWL [90] to define generic vocabularies for individual domains using low-level activity definitions. In such approaches, the context is represented as a set of axioms about entities and

resources that are further associated through properties and relationships, providing a uniform way of representing data. For instance, approaches in a smart home [47, 48, 220], contexts correspond to OWL individuals, and realization is used to determine into which context concepts a specific situation individual falls into. In addition to using an ontology, other approaches such as Context Aggregation and REasoning (CARE) middleware also use statistical reasoning for context inference (e.g., business meeting) based on the location or environment (e.g., office) and with at least two actors (e.g., employees) [10]. Other approaches extend their ontology such that their OWL 2 reasoning module incorporates temporal correlations of complex activities using rules and well-defined SPARQL queries that are essential in context recognition [147, 148]. While this approach to modeling complex relations between activities as context is useful, the definition of numerous activity patterns is often static and highly structured. In contrast, the TAO system allows us to model and reason over intricate, simple temporal dependencies between activities that indicate activity patterns that are sequential and parallel.

Temporal Clustering Approaches: Prior research has also proposed data-driven methods, such as using temporal clustering algorithms to identify patterns in multivariate timeseries data. These approaches focus on identifying sensor data patterns from multimodal sensors to detect activities in the space [115, 118, 137, 217]. Other approaches, on the other hand, focus on directly modeling individual behavior patterns (such as movement and activity routines) using sensor data from mobile and/or ambient sensors [3, 105, 114, 140, 187, 214, 245]. Other temporal clustering methods aim to identify a complex relationship between simple activity sequences when demonstrating a procedure such as *"how to change a tire"*[70, 82, 125, 231]. Their primary focus is to extract procedural knowledge about a particular kind of long-term activity (i.e., cooking, building models, etc.) to enable skills for AI agents[53, 247] or to understand human psychology[93, 141]. Other approaches use probabilistic and statistical methods to model temporal activity patterns and identify abnormal human behaviors [18, 139, 195]. While these data-driven pipelines identify activity patterns, they are limited to applications such as identifying anomalies and do not focus on capturing the semantic context of these activity patterns.

More recent approaches have proposed ML-based methods that use fine-grained sensor data to measure wellness indicators such as mood instability [156], productivity, and stress [13, 117] without requiring the need for context detection. However, such approaches rely on fine-grained data from several input sources, such as cameras, wearables, and smartphones. As a result, these approaches are susceptible to sensitive and noisy sensor data. More importantly, these approaches require manual user input to self-report productivity or stress every hour, which can be cumbersome. In comparison, TAO proposes an ontology-based approach to predict contexts accurately and automatically which can then be used to calculate wellness metrics.

Hybrid Approaches: Several recent efforts aim to combine both knowledge-driven and data-driven approaches to derive semantic relationships between activities for context inference [96, 119, 164, 192, 194, 196, 236]. COSAR [192] uses a statistical classifier that recognizes an activity which is then used to obtain context information using the ontological reasoner from the ActivO ontology models [193]. In other approaches [96, 164, 236], the sensor data is segmented based on activity relationships inferred from ontology, and a clustering model is trained to capture temporal patterns related to a context. These hybrid approaches are closely related to the

TAO system, showing the promise of data-driven pre-processing for activity inference in combination with ontology to improve context recognition. Such approaches, however, are easily affected by the noisy nature of sensor data streams and events, resulting in inaccurate context recognition. In addition, such an approach fails to identify complex activity patterns that can be used to define richer contexts.

4.3 TAO: System Architecture

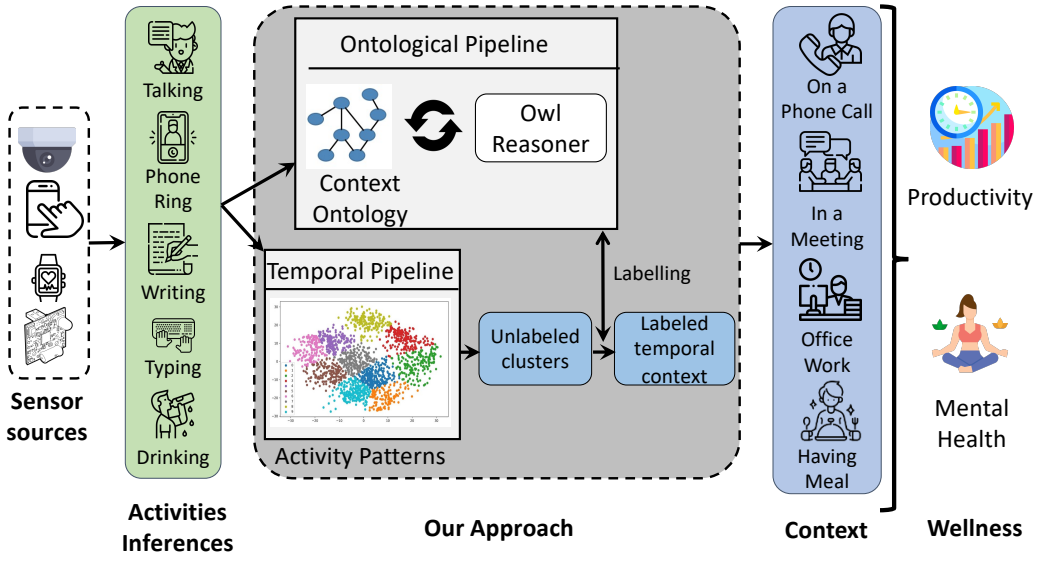


Figure 4.1: Overview of the TAO's system architecture. TAO leverages OWL-based ontologies and temporal clustering approaches to identify context from the stream of activities obtained from Human Activity Recognition(HAR) systems. The contexts detected by TAO are then sent to our example Wellness application which then infers productivity and stress.

4.3.1 Overview

Figure 4.1 shows the system architecture of TAO highlighting two key components namely the ontology pipeline (§4.3.2) and the temporal pipeline (§4.3.3). Our ontological pipeline is responsible for providing a standard vocabulary for modeling activity-related information, such as domain activity classes, actors, etc., and for formally describing the complex relationships between the activities (activity patterns) as Contexts (§4.3.2). In addition, it also provides a method to derive contexts from complex activity patterns using the OWL 2 reasoning paradigm and the execution of SPARQL CONSTRUCT queries (§4.3.2). Our temporal pipeline (§4.3.3) is responsible for learning context representations based on the complex activity patterns that happen over a period of time (e.g. 5mins, 10mins, 30mins, 1hr). To enable this, our temporal pipeline uses a novel featurization technique to model activity relations and a deep-learning-based unsupervised

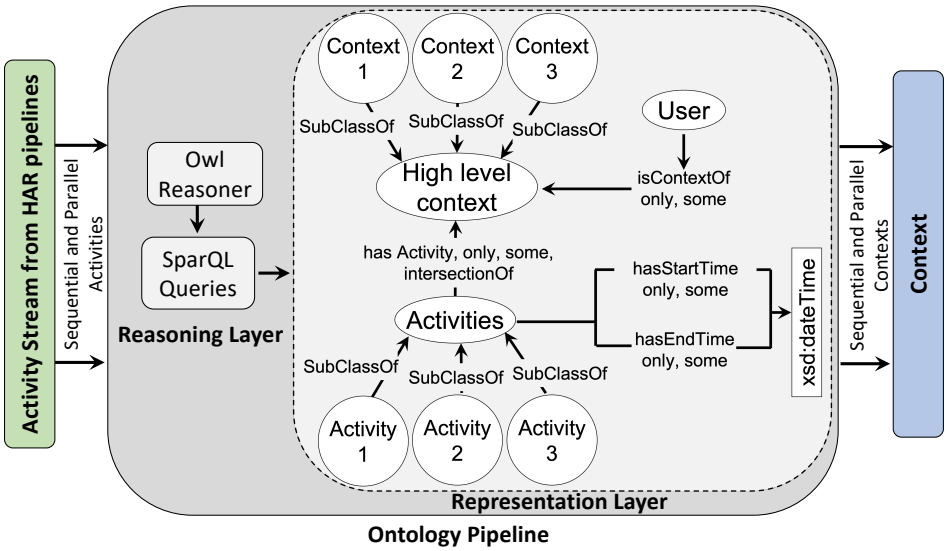


Figure 4.2: Overview of our ontological pipeline within TAO.

learning approach to model activity patterns. The unlabelled clusters in our temporal pipeline utilize the ontological pipeline to label these representations to infer existing and new contexts. Overall, our TAO system uses the activity inference from human activity recognition pipelines which are then fed into TAO, specifically the ontological and the temporal pipelines, to assess the activity patterns and generate semantically meaningful and richer contexts that can be used for applications such as indicating wellness such Productivity or Stress. We describe each of these components in further detail.

4.3.2 Ontological Pipeline

The architecture of our ontological pipeline is shown in Figure 4.2. It comprises a Representation Layer and a Reasoning Layer. We designed our representation layer such that it consists of a dense representation of complex activity patterns which we see in daily life and can be interpreted into contexts. In addition, we wanted to ensure that we modeled the context information that is accessible using simple queries of activity relationships. This is primarily useful in cases where we are performing we want to interpret a context based on a complex activity relationship that consists of several constraints, such as time duration or order of important activities. To enable this, we built our custom ontology framework based on the prior work[148, 220] to provide a dense vocabulary for formally describing the context as complex relations among activities. We describe the representation layer further in §4.3.2. We design our reasoning layer such that we can derive contexts by interpreting the relationships between the defined activities in our custom ontology. The reasoning layer uses an OWL 2 reasoner [90] to model complex activity patterns, e.g., class subsumption, sub-properties, inverse properties, etc., while the context inference is realized by custom-defined SPARQL queries [200].

Representation Layer: The goal for the representation layer is to design an ontology to provide a standard definition of vocabularies that models contexts as relationships between complex

activity patterns that happen in our daily life. These contexts include (1) contexts based on sequential activity patterns, (2) contexts based on parallel activity patterns, and (3) contexts based on a variation in the duration of activity. For example, for a context *C: OnAPhoneCall* which requires the *A: phone ring* activity to be sequentially followed by the *A: talking* activity. Similarly, often multiple activities may occur simultaneously, such as *A: typing*, *A: talking*, or *A: chewing*, which indicates that multiple contexts are happening at the same time, such as *C: OfficeWork*, *C: HavingMeal* and *C: InAMeeting*. Moreover, context inference also varies based on the duration of activities. For example, both contexts *C: OnAZoomCall* and *C: OnAPhoneCall* require the activity patterns to be sequential (*A: phone ring* to *A: talking*); however, the duration of the activity *A: talking* (5mins vs 30mins) differentiates these contexts. Thus, to model these complex activity patterns as contexts, we designed the representation layer with a custom ontology framework that incorporates the concepts proposed in the Meeting Minds Ontology [220] and the lightweight Domain Activity Ontology [148]. Specifically, we leverage the vocabularies defined by the Meeting Minds behavioral scientists [220] for contexts as activity combinations and the definition concepts from Domain Activity Ontology [148] to enable capturing of time duration information of activities. Combining these approaches allowed us to model a wider range of context as an intersection of multiple activities and relationships beyond what was possible by these individual approaches.

The activities of daily life are represented as instances of the *Activity* class and they are linked to ranges of time through the use of the *hasStartTime* and *hasEndTime* datatype properties. The instances of the *Contexts* class models the relationship between the instances of the *Activity* class using the object properties *hasActivity* along with *intersectionOf, some, only, union* object definitions. For example, the context *C: OfficeWork* is an instance of the class *Context* and models the relationship between instances of the *Activity* class such as *A: typing* and *A: writing* using *intersectionOf* object properties.

Reasoning Layer: Our reasoning layer derives context by meaningfully interpreting the relationship between the primitive activities specific to the context using a combination of standard OWL reasoners and SPARQL queries. The OWL reasoners determine whether or not the ontology is consistent and identify subsumption relationships between classes, such as Context or Activities. The SPARQL query language then allows customized queries to interpret Context. We used this combined approach to design our reasoning layer primarily because an OWL reasoner by itself cannot support query-like operations that are required to interpret contexts. Combining both the reasoner and SPARQL queries, on the other hand, can render the context of an activity or multiple activities easily. Thus, TAO uses the OWL 2 reasoner to formulate the context and activity relationships, and we then use the SPARQL queries to query the combined pieces of information (intersections). More specifically, the semantics of the ontology, e.g., property restrictions, subproperties, inverse properties, etc., are handled by the OWL 2 reasoner, whereas domain-specific SPARQL queries realize the context recognition.

SPARQL context interpretation queries: The SPARQL query language enables ease for querying graph patterns along with their conjunctions and disjunctions. The SPARQL in the TAO system is defined in terms of a CONSTRUCT and a WHERE clause: the CONSTRUCT clause defines the graph patterns, i.e., the set of information that should be returned to upon the success-

```
CONSTRUCT {
?x a :OfficeWork .
    :hasActivity ?x;
    :hasActivity ?y;
}
WHERE{

?x a typing;
?y a talking;
BIND((URI(?x, ?y) as ?new) .
NOT EXISTS (?new a [] .)

}
```

Figure 4.3: Query to infer *A: typing*

```
CONSTRUCT {
?x a :OnAPhoneCall .
}
WHERE{
    ?x a phoneRing;
        :hasStartTime ?st;
        :hasEndTime ?et.
    ?t a talking;
        :hasStartTime ?st1;
        :hasEndTime ?et1.
    FILTER (contains(?st, ?et))
    FILTER (before(?st1, ?et1))
}
```

Figure 4.4: Query to infer *C: On a phone call*

ful pattern matching of the graph in the WHERE clause. We show two simple examples for the graphs-based SPARQL queries in Listings 4.3 and 4.4. The first listing 4.3 shows how we query handles the composition semantics of *C: OfficeWork* context, using the classes *A: typing* and *A: talking* as its sub-activities. Similarly, listing 4.4 shows modeling of *C: OnAPhoneCall* context using the *A: phoneRing* and *A: talking* as sub-activities with time duration information of how long each activity would be performed. These queries can be updated to infer context based on complex dependencies of underlying activity relationships.

4.3.3 Temporal Clustering Pipeline

Ontological approaches work well to detect instantaneous contexts based on short sequential patterns or parallel occurrences of activities. However, in real life, we regularly switch between contexts. At one time, we can be in multiple contexts, i.e., reading and having a meal at the same time or listening to music while vacuuming, etc. These settings with multiple contexts provide an opportunity to derive richer context information by capturing repetition across interleaved sequential and parallel activity patterns. We present a temporal clustering-based approach to detect such interleaved and recurring patterns. We define activity detection pipeline as a set of (one or more) models that utilized raw data from any sensor sources (see Figure 2.1) and outputs instantaneous activity inference(s) for a given timestamp (i.e., typing, talking, drinking, etc.). These outputs are combined into a single activity stream which consists of *timestamp-activity* inference pairs. This activity stream is an input for our context prediction model. Further, we parameterize the input activity stream with two parameters, (a) *stacking window*, and (b) *lag window* (see Figure 4.5). We define the *stacking window* as the time interval to wait for receiving activity inferences. Typical time intervals for a *stacking window* are from a few seconds (1 second, 5 seconds, etc.) to a few minutes (1 minute, 5 minutes, etc.) based on the output rate of the activity detection pipeline. Activities received in one *stacking window* are considered to happen in parallel . i.e., in a sample activity stream shown in Figure 4.5, we see that we receive one activity inference between 0-2 minutes (*A: eating*), no activity inferences between 4-6 minutes, and multiple activity inferences between 8-10 minutes (*A: drinking* and *A: writing*). To capture patterns over the sequence of activities, we observe the incoming activities for a time

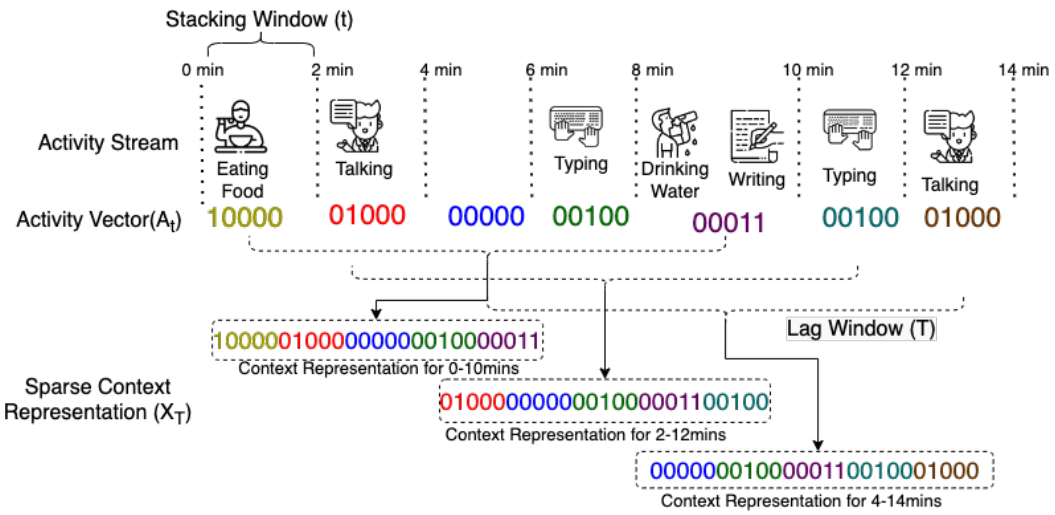


Figure 4.5: An example of context representation (five activities only) for 2 min stacking window(t) and 10 min lag window(T).

interval (defined as the *lag window*), which is longer than the *stacking window*. Typically, a *lag parameter* is set to a multiplier (5x, 10x, 30x, etc.) of the *stacking window*.

The sequence of (a set of parallel) activities happening in a *lag window* is used for context prediction at the end of the *lag window*. Figure 4.5 shows how we encode activities in a *stacking window*(t) as a binary encoding A_t of size N, where N is a set of all possible activities, with each positional argument being 1/0 based on whether that activity is detected. Next, we stack these binary vectors horizontally for a *lag window* (T) to create a sparse representation of activity patterns X_T seen in the *lag window*. This is the final input to our temporal clustering (See Figure 4.7), which consists of three components, (i) *Representation Trainer* which learns meaningful embeddings from context representation X_T, (ii) *Cluster Trainer* which clusters these embeddings into distinct context clusters, and (iii) *Context Labeler* which derives labels for prominent context clusters using the ontology reasoning layer.

Representation Trainer: A naive way to identify recurring temporal patterns is to cluster these sparse representations directly. However, positional dependence of activities leads to significantly different representations for similar contexts. i.e., *A: jumping* followed by *A: jogging* would be quite different from *A: jogging* followed by *A: jumping*. One way to reduce this bias is to featurize context representations to encode sequential and parallel patterns explicitly. Finding a fixed approach for such featurization is not generalizable as the importance of such highly localized patterns can differ for different user sets. Rather, we built a *Representation Trainer*, which learns to featurize our input context representation X_T (see Figure 4.5). It utilizes an autoencoder-based approach, an unsupervised neural network that learns how to compress and encode data efficiently and then how to reconstruct the data from the reduced encoded representation to a representation as close to the original input as possible [242]. The encoded data, i.e., dense context embedding Z_T, is further used as initial input for our *Cluster Trainer*. A basic autoencoder [222] maps input to a denser latent representation but does not guarantee to encode

sequential or parallel patterns. Another baseline approach is to use Long Short Term Memory (LSTM) for modeling sequential patterns [100]. However, based on our initial experiments when designing the temporal pipeline, we found that LSTM-based autoencoders failed to learn context representation effectively for our use case. One major reason is that LSTMs need dense input data and longer sequences to learn meaningful representations. Thus, they fail to model temporal patterns effectively from sparse context representation.

Based on our findings from baseline approaches, we used a specialized network architecture, Temporal AutoEncoder (TAE) [142], that creates dense embeddings, encoding temporal information over sequences of activity stream in X_T. TAE can effectively capture temporal relationships in multivariate time series data. Figure 4.6 shows an architecture for the TAE encoder and decoder. The first level of network architecture consists of a 1D convolutional layer, which extracts short-term features (waveforms), followed by a max pooling layer. This casts time series into a more compact representation while retaining the most relevant information. This step helps learn local patterns in the sparse context representation, thus modeling information from activities happening in parallel. These compact representations are then fed into the second layer, consisting of two bi-LSTM cells, to learn temporal changes in both time directions. The Bi-LSTM layers help in capturing sequential patterns of activities in final embedding. Finally, reconstruction is done by an upsampling layer followed by deconvolution later to obtain a reconstructed signal. This architecture has been tested with time series information across various domains in literature [142]. Our *Representation Trainer* pre-trains a TAE architecture with a binary cross entropy [61] loss function to learn context embeddings Z_t. These context embeddings are provided as an initial representation to derive optimal cluster count and later cluster a variety of activity patterns into context clusters with our *Cluster Trainer*.

Cluster Trainer: A direct approach to derive contexts from embeddings Z_T is to use partition or hierarchical clustering approaches to derive prominent contexts based on cluster centers. However, these approaches do not utilize data-specific patterns in activity streams (X_T) to enhance separability in centroids for better clustering. Existing literature in clustering methods has shown superior performance by jointly training autoencoder for latent representation and clustering error[243]. Our *Cluster Trainer* uses a similar, combined training approach for cluster assignment, which is inspired by Deep Temporal Clustering (DTC) [142] and Deep Embedded Clustering (DEC) methods [229]. In this approach, we iteratively update cluster centroids to optimize for the separability of clusters and fine tunes weights in the *Representation Trainer* to ensure that learned context embeddings are best suited to separate context representations X_T into given categories.

One of the challenges of using such a combined training method is to figure out an optimum number of clusters. As the number of recurring activity patterns is different for different settings, we need a method to figure out cluster count based on given context representations X_T. Spectral clustering methods (i.e. DBSCAN, OPTICS, HDBSCAN, etc.) provide a natural way to figure out optimal cluster count based on other parameters like distance threshold for neighboring points, minimum clusters in a sample, etc. However, we observed that spectral clustering methods are sensitive to the subspace of embeddings Z_T and their hyperparameters. For different kinds of activity streams and length of context representation, underlying subspace Z_T varies significantly, thus leading to a wide range for the number of optimal clusters, whereas the number

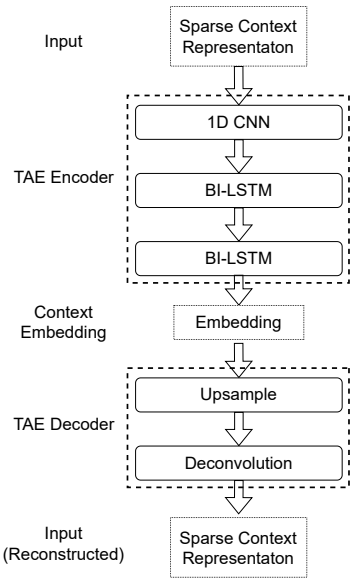


Figure 4.6: Architecture of temporal autoencoder (TAE).

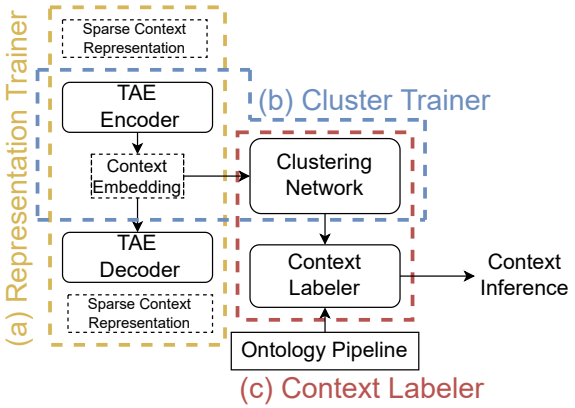


Figure 4.7: Overview of TAO's Temporal Pipeline. Representation Trainer(a) pretrains TAE Encoder-Decoder to learn dense context embedding from sparse input representation, then Cluster Trainer (b) uses TAE-Encoder and learned embeddings to discover optimum context cluster count and their sparse representation, and finally, Context Labeler(c) uses ontology to provide labels to generated context clusters.

of meaningful interleaved patterns for a given set of activity streams is finite. Thus, we used a simpler method using silhouette scores[238] with K-Means clustering over context embeddings Z_T. One caveat of using this method is that partition-based clustering assumes convex cluster boundaries, which is alleviated by fine-tuning cluster membership of context representations X_T based on the deep clustering method in the next step.

Next, we initialize centroids with K-Means clustering with optimal cluster count over learned context embeddings Z_T, followed by iterative training with an unsupervised method that alternates between two steps till it ends. First, we compute q_{ij}, which is the probability of assignment of input X_i belonging to cluster j based on the distance of context embedding Z_i and centroid w_j. The closeness is evaluated using complexity invariant distance (CID) [27], thus allowing a generalizable distance metric across various complex activity streams. It is based on a Euclidean distance (ED), corrected by the complexity factor (CF) of two representations (Z_i and w_j)

$$dist(Z_i, w_j) = CF(Z_i, w_j) * ED(Z_i, w_j)$$

$$CF(Z_i, w_j) = \frac{max(CE(Z_i), CE(w_j)}{min(CE(Z_i), CE(w_j))}$$

$$CE(x) = \sqrt{\sum_{t=1}^{N-1}(x_{t+1} - x_t)^2}$$

where $CE(x)$ is the complexity estimate of a sequence x. We normalize these distances $dist(Z_i, w_j)$ into probability assignments q_{ij} using a Student's t distribution kernel with a single degree of freedom [216].

$$q_{ij} = \frac{(1 + dist(Z_i, w_j))^{-1}}{\sum_{j=1}^{k}(1 + dist(Z_i, w_j))^{-1}}$$

Second, we train the *Cluster Trainer* iteratively by minimizing KL divergence loss between q_{ij} and target distribution p_{ij} to strengthen high confidence predictions and normalize the losses to prevent distortion of context embeddings, which is

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j=i}^{k} q_{ij}^2/f_j}$$

where $f_j = \sum_{i=1}^{n} q_{ij}$, which is derived empirically [99, 229]. Finally, we compute KL divergence loss:

$$L = \sum_{i=1}^{n}\sum_{j=1}^{k} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

We use a weighted combination of KL divergence loss and *Representation Trainer* loss to optimize centroids and fine-tune the embedding encoder weights. This provides us the final cluster centroids $w_j^{*(z)}$ in the subspace of context embedding Z_T, which are fed back into the *Representation Trainer* to reconstruct $w_j^{*(x)}$, a representation of cluster centroids in the subspace of sparse context representations(X_T's).

Context Labeler: Finally, $w_j^{*(x)}$ is the context representation for each cluster and defines the context for all X_T belonging to cluster j. However, when converted into timestamped activities A_t^*, these representations can be unnecessarily long and interleaved with activities from multiple contexts. A simple example can be a cluster representation shown in Figure 4.5 between 4-14 minutes, i.e., *A: typing ¿ A: drinking+A: writing ¿ A: typing ¿ A: talking*. We use a heuristic method to break this sequence of activities and run multiple queries through our ontology to detect semantically meaningful contexts.

A naive approach is to query our ontology to provide context for all possible subsets of sequential, parallel, and single activities. It will provide multiple contexts (i.e., for our example *C: In a meeting*, *C: Having meal*, *C: Office work*, etc., whereas the actual context is only *C: Office work*). To reduce inaccurate contexts, we use two heuristics that work well in practice. First, we only find contexts from all sequential and then parallel activity patterns of two activities. i.e., in our example, an ontology query for (a) *A: typing*, (b) *A: drinking+A: writing*, and (c) *A: typing ¿ A: talking* will return *C: Office work*. If we do not find any meaningful contexts, we find contexts based on single activities in the set of interleaved activities. Second, we use our ontology to find the subset of activities that identify to a particular context (i.e., only *A: typing* would indicate the *C: Office work* context, in contrast to only *A: talking* could indicate multiple contexts like *C: Having meal*, *C: Office work*, etc.). We attempt to find context labels based on interleaved sequences of these subsets of activities first. If we do not find meaningful contexts, we combine the contexts from all possible subsets of sequential, parallel, and single activities.

If new activities are found in activity streams not modeled in the ontology, the ontology cannot provide contextual information even with the entire subset of relations. In such cases, we can ask users to label the context based on their understanding manually. These patterns and provided contexts can be used to update the ontology to capture a richer set of contexts over time. We keep this as a future extension of our work and scope our current work to a set of activities modeled by our extended ontology.

4.3.4 Putting it All Together: The TAO System

We combine context predictions from the ontology and temporal pipeline to give final context predictions. A simplistic way to combine contexts from both pipelines is to take the union of unique contexts predicted for a given time period. However, this method leads to an overestimation of the context for activities. One primary reason for overestimation is the presence of ambiguous activities(i.e., activities that span across multiple contexts like *A: Sitting* or *A: talking*), which leads ontology to predict all the possible contexts, i.e., *C: Office work*, *C: Amusement* and *C: Having meal*, etc. To alleviate this issue, we opportunistically select output from one of the pipelines based on a simple metric, the number of contexts predicted by each pipeline for a given time period. An ontology is deterministic in nature as it consists of pre-defined rules for mapping activities to contexts. Thus, the ontology pipeline provides more reliable context detection than the temporal pipeline for scenarios where we only see a single activity in the activity stream or a well-defined sequential set of activities. On the other hand, the temporal pipeline is more reliable in a multi-context setting, as it predicts multiple contexts only when corresponding activity patterns are recurring in activity streams. In comparison, the ontology provides multiple context predictions due to the presence of ambiguous activities (like *A: Sitting* or *A: talking*). Thus, we choose an output from the temporal pipeline when there are multiple context predictions from either of the pipelines.

4.4 Evaluation

In this section, we evaluate the TAO's ability to accurately identify context from different activity patterns using representative HAR datasets as well as a real-world study. Our evaluation aims to answer the following questions:

- TAO's goal is to *accurately* detect contexts from different activity patterns. *RQ1: How well do the components of TAO, namely the ontological and temporal pipelines, detect contexts from daily activities? How well does TAO's hybrid approach that combines both pipelines detect contexts?*
- TAO is also *robust* in detecting contexts in real-world settings. *RQ2: How well does TAO perform in a real-world deployment, and how does the accuracy of an underlying activity detection pipeline impact TAO's performance?*

We now briefly summarize the results of our evaluation. A more detailed evaluation is available in our full paper [38].

4.4.1 Evaluation Setup

To evaluate our TAO framework, we benchmark it on two human activity datasets (*Extrasensory* [215] and *Casas* [57]) and on a real-world activity detection pipeline. The dataset based evaluation allows us to evaluate performance of TAO on continuous activity streams across multiple days (longitudinal study), and compare TAO with other approaches that have also been evaluated on the same datasets. The *Extrasensory* dataset consists of activity and location labels of smartphone sensor data from 60 participants across several days (about 7-14 days per participant). The activity labels per participant are at 1 minute intervals, and there are 34 unique activity labels across all participants. The *Casas* dataset collects data at 1 second granularity from ambient sensors deployed in 30 different smart homes across 1.5 months. This data is manually labeled by researchers, and entire dataset consist of 42 unique activity labels.

Ground-Truth Context Annotation: Most public datasets have activity annotations on various sensor data. A few public datasets have context annotations from sensor data. But, none of these datasets have an activity for context mapping. Thus, we created our own context labels using annotated activities from these datasets. We used the activity labels to create timestamped activity traces (at a participant level in *Extrasensory* and at a home level in *Casas*). We identified 14 different contexts from these activity labels, and two researchers independently annotated 90 days(randomly sampled) of activity data across both datasets. We annotated a total of 1300 hours of data across 1800 annotated instances of contexts.

Performance Metrics: To evaluate the accuracy performance of TAO, we use two metrics: (1)*Jaccard similarity coefficient* (JC) [106] and (2)*Precision* and *Recall*. The *Jaccard similarity Coefficient* (JC) computes the degree of set overlap between the annotated ground truth context and the detected context. If $Context_{Det}$ is the set of contexts detected, and $Context_{GT}$ is set of contexts present in ground truth for a given timestamp, JC is given by:

$$JC(Context_{Det}, Context_{GT}) = \frac{|Context_{Det} \cap Context_{GT}|}{|Context_{Det} \cup Context_{GT}|}$$

The JC ranges from 0-1, i.e., 0 when there is no overlap between ground truth and detected contexts and 1 if ground truth and detected context(s) are identical. Further, JC penalizes methods that detect contexts that are not present in the ground truth (larger value of union of sets) and for not detecting some contexts present in the ground truth (smaller value for the intersection of sets). *Precision* and *Recall* is calculated based on True Positives (TPs), False Positives (FPs), True Negatives (TNs) and False Negatives (FNs) at a context level. A specific detection for a context (say *C: OfficeWork*) is considered as TP if it is both detected and present in the ground truth, an FP if it is detected but not present in the ground truth, and an FN if it is not detected but present in the ground truth. The Precision and Recall metrics at the context level are then calculated as $P = TP/(TP + FP)$ and $R = TP/(TP + FN)$. For overall accuracy, we use an average of *Jaccard Similarity Coefficient* (JC) across all detection events and the weighted average of *Precision* and *Recall* at context level, weighted by a number of times a particular context that appears in the ground truth. In addition, we calculate the F1 score metric to compare our system with prior approaches.

4.4.2 Evaluation of TAO's Ontological, Temporal and Hybrid Pipelines

We evaluate the components of TAO, the ontology, and the temporal pipeline to measure their accuracy performance in isolation. For this evaluation, we use the two HAR datasets and our ground truth labeled contexts. Specifically, we highlight the differences in terms of accuracy (JC), precision, and recall of context prediction, as applicable, when we use the TAO system with other datasets.

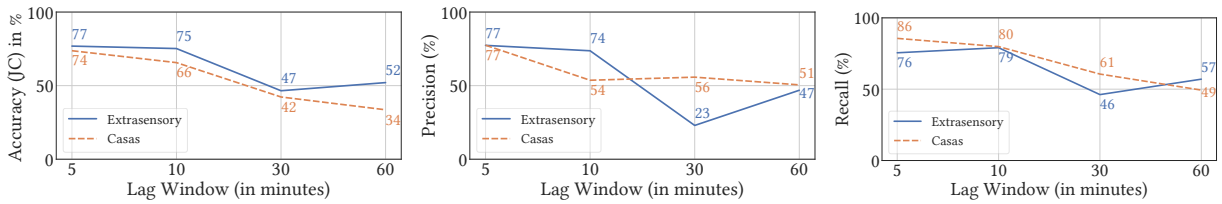
Ontological Pipeline: We evaluate the TAO system's ontological pipeline in terms of accuracy and richness in capturing contexts present in the two HAR datasets with rich activity patterns. We then compare our ontological approach with prior ontological approaches, such as the Meeting Minds (MM) context ontology [220], which models context as the relationship between activities. For a fair comparison, we update the activity labels from the datasets to be consistent with the activity instance definitions in the respective ontologies such that all the activities and their corresponding timestamps can be used for querying the ontology. Then, these activity labels are posed to the respective reasoners, and the corresponding contexts are inferred. Table 4.1 compares the accuracy (measured by the *Jaccard Similarity Coefficient* (JC)) and the contexts detected by each approach, categorized into sequential and parallel contacts. We observe that for both datasets, our approach detects contexts at a higher accuracy (*Extrasensory* - 53.94%, *Casas* - 72.8%) when compared to the MM approach (*Extrasensory* - 0.03%, *Casas* - 11.03%). We further observe that in the *Extrasensory* dataset, the low accuracy of the MM approach is primarily due to its inability of ontology to infer contexts from activity patterns that are parallel (e.g., *A: talking*, *A: eating*, *A: tv* at the same time). In comparison, our approach can detect such contexts with high accuracy(60%). In addition, we observe that in the CASAS dataset, which consists of sequential activity patterns, our approach still outperforms MM (TAO 72.08% vs MM – 11.03%). This is because our approach consists of a denser vocabulary of activity relationships and contexts compared with prior work. Overall, we show that our ontological approach performs much better than prior approaches to detect an accurate and richer set of contexts.

Table 4.1: Comparing the accuracy (JC) and the total contexts detected by TAO's ontological pipeline only with prior work on two HAR datasets. Our approach detects different contexts (sequential and parallel) at a higher accuracy (72.8% – *Casas* and 53.9% – *ExtraSensory*).

Dataset	**Ontological Approaches**	**Sequential Contexts**		**Parallel Contexts**		**Total Contexts**	
		Acc(JC)	Count	Acc(JC)	Count	Acc(JC)	Count
ExtraSensory [215]	MeetingMinds [220]	20.3%	976/4585	–	–/17265	0.03%	976/21850
	TAO (Ontology only)	**24.6%**	1138/4585	59%	10648/17265	**53.94%**	11786/21850
CASAS [57]	MeetingMinds [220]	11.03%	38949	NA	NA	11.03%	38949/182604
	TAO (Ontology only)	**72.80%**	153022/182604	NA	NA	**72.80%**	153022/182604

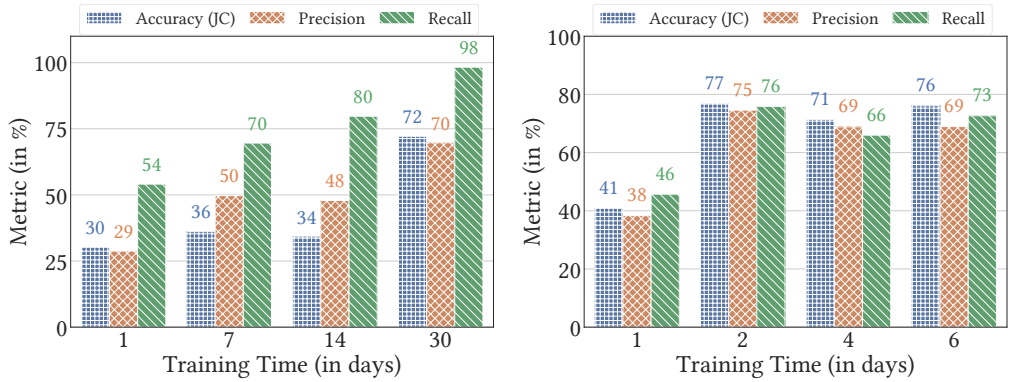
Temporal Pipeline: We evaluate the performance of TAO's temporal pipeline across two key metrics, (1) *lag window*, which is the duration of the activity stream used for creating a sparse context representation(X_T), and (2) the amount of training data used for learning context clusters.

Optimizing the Lag Parameter: Figure 4.8 shows the different performance metrics of models



(a) Accuracy (JC) comparison (b) Precision comparison (c) Recall comparison

Figure 4.8: Comparing the accuracy (JC), precision, and recall of context detection for TAO's temporal pipeline for different duration of *Lag Window*, across the *Casas* and the *Extrasensory* dataset. Our approach has high accuracy(65%-75%) for short (5 and 10 minutes) *Lag Window* in comparison to long (30 and 60 minutes) *Lag Window*. We also observed for both datasets, recall of context detection is consistently higher than the precision of context detection.

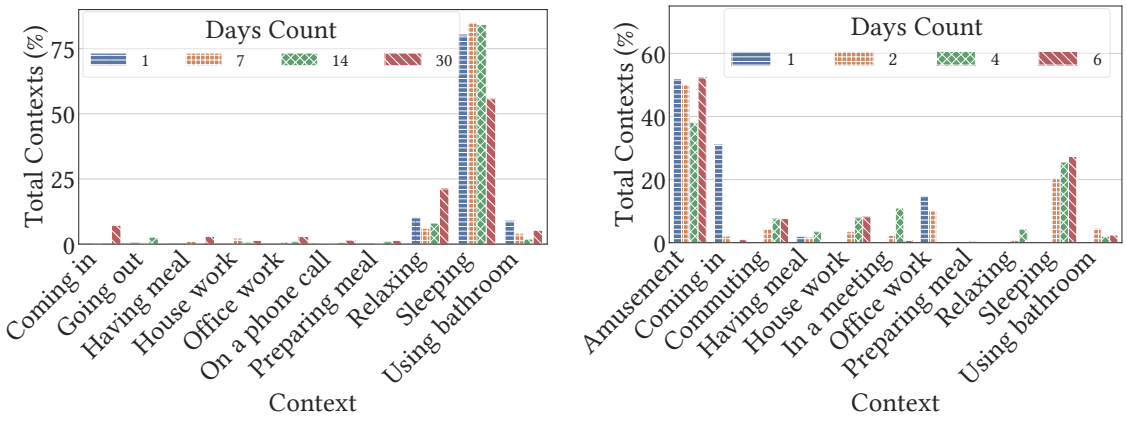


(a) Casas Dataset (b) Extrasensory Dataset

Figure 4.9: Comparing the accuracy (JC), precision, and recall of context detection for TAO's temporal pipeline as the data (in days) available for training for a single user increases. For the *Casas* dataset, our approach shows a consistent increase in accuracy, precision, and recall values as the training data increases. For the *Extrasensory* dataset, our approach shows high accuracy within two days of training data.

trained on the *ExtraSensory* and the *Casas* datasets for different *lag parameters*: 5 min, 10 min, 30 min, and 60 min respectively. We see that accuracy (JC) decreases across both datasets as we increase the lag parameter. This shows that a lag parameter of 5 minutes is sufficient to capture various activity patterns happening across multiple contexts. Larger values (10min, 30min, 60mins) miss shorter duration contexts (i.e., *C: Using bathroom* between *C: Office work*) or overestimate context presence across a larger window (i.e., *C: Office work* and *C: Having meal* lasting for the entire 60-min window). We see more dips in precision than recall, highlighting that shorter duration contexts are overestimated with larger lag values rather than shorter duration contexts being missed. We set the lag parameter to 5 mins based on our findings to provide the most accurate contexts for all subsequent evaluations.

Accuracy with Incremental Data: Figure 4.9 shows how the performance measures of our temporal pipeline change as the data used for training is increased, expressed as the number of days. To show this, we use data from two users chosen randomly , one from each dataset. The user from the *Casas* dataset has 45 days of ground truth data. Thus we trained different models



(a) Casas Dataset Context Distribution

(b) Extrasensory Dataset Context Distribution

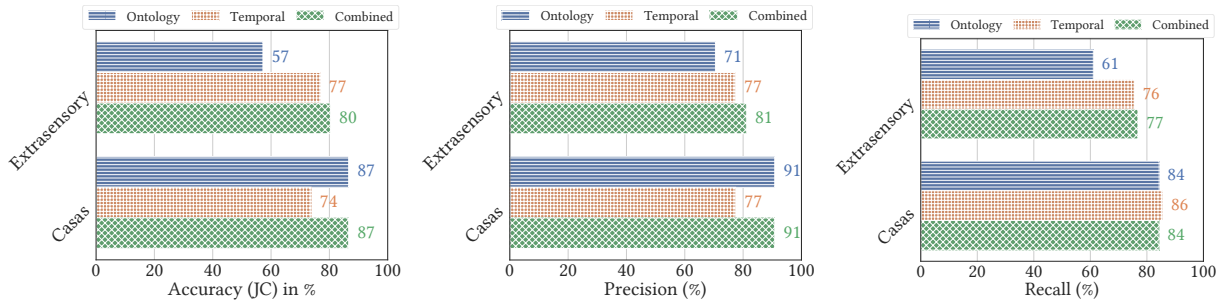
Figure 4.10: Comparing the distribution of predicted contexts learned over 45 days (1/7/14/30/45 days) for Casas and six days (1/2/4/6 days) for Extrasensory. In Casas, we see that with 30 days of training data, our pipeline can detect a wide range of contexts more accurately (72% JC). For the *Extrasensory* dataset, our approach shows that it can detect several contexts within two days of training data

with 1-day, 7 days, 14 days, and 30 days of training data. All trained models are then tested on the same one week of data to maintain consistency across users, which is not included in any training data. We use a similar protocol for the data from the user in the *ExtraSensory* dataset, except we trained different models with 1 day, 2 days, 4 days, and 6 days, and tested it on the same remaining two days in the end. We observe that for the *Casas* dataset user, we see a consistent increase in recall of context detection as training data increases. We get a good recall for contexts for a single day of training. However, the precision of context detection is low. For the user from the *ExtraSensory* dataset, we observe that precision and recall improve as we go from 1 to 2 days of training and then remain around the same as additional training data are used. The small dip in performance is an artifact of relatively less testing data. Figure 4.10 shows the distribution of contexts detected from the activity patterns. We see that in Casas, which predominantly consists of sequential activity patterns, with 30 days of training data, our pipeline can detect a wide range of contexts more accurately (72% JC). The context that is majorly detected is *C: Sleeping* since this activity happens *A: laying down, A: sleeping on bed* for a major period of the day. For the *Extrasensory* dataset, our approach can detect several contexts within two days of training data ranging from Amusement to Office Work.

4.4.3 Evaluation of TAO's Performance

We evaluate the components of TAO, the ontology, and the temporal pipeline together to measure their accuracy performance.

Overall performance: Figure 4.11 shows the overall performance of our ontological approach and the temporal approach in isolation and then TAO's hybrid approach that combines them. For this evaluation, we split users into separate training and testing sets, and we trained one



(a) Accuracy(JC) (b) Precision (c) Recall

Figure 4.11: Comparing the Accuracy (JC), Precision, and Recall of context detection on the two datasets.

model per dataset. For the *Casas* dataset, we trained on data from 28 out of 30 users, and then evaluated the remaining 2 users. For the *ExtraSensory* dataset, we used 53 out of 60 users for training and tested on remaining 7 users. We observe that the temporal pipeline by itself has around 75% accuracy (JC) across both datasets. However, the ontological approach by itself has significantly higher accuracy for the *Casas* dataset as compared to the *ExtraSensory* dataset. This is attributable to the different types of activity patterns in the two datasets. In *Casas*, most of the activities happen sequentially (i.e *A: toilet>A: eating>A: washing dishes*), which leads to simpler mappings to context (i.e *C: UsingBathroom>C: HavingMeal>C: HouseWork* etc.). However, in the *ExtraSensory* dataset, most activity patterns are interleaved for a long period of time (i.e *A: walking>A: with friends+A: talking>A: talking*) and users are in multiple contexts simultaneously (i.e *Commuting, Amusement* etc.). Due to this, our ontological only approach could not detect these contexts accurately. However, TAO's temporal pipeline tries to learn recurring patterns in the datasets instead of using a static mapping, thus can reason well about these patterns. TAO's hybrid approach performs better in terms of its accuracy (JC), which is slightly better than average. It has a lower precision and a higher recall across individual components. This shows that our combined approach is detecting all contexts present in the ground truth, but it is overestimating individual contexts.

4.5 Conclusion

We present TAO, a context detection system that combines ontological and deep unsupervised clustering approaches for inferring a rich set of contexts from a wide variety of daily activities. The TAO system models the different activity patterns sequential, parallel, or interleaved activities as context information using the OWL-based ontologies. The temporal pipeline uses an unsupervised clustering algorithm to detect context from new activity patterns and automatically extends our ontology based on new activity patterns. Our system is agnostic to underlying activity detection mechanism and set of activities detected, making it usable across various sensing systems to derive semantically richer information from activity streams. We showed that the TAO system performs well across multiple settings we explored using two well-known public datasets and our real-world study.

Chapter 5

General-Purpose Edge Audio Filter for Privacy-Preserving Activity Recognition

Throughout this thesis, the focus has been on devising innovative approaches to support general-purpose sensing systems, catering to diverse application needs and stakeholder requirements. As we delve deeper into enhancing the capabilities of IoT sensing frameworks, a critical aspect emerges the need for privacy-preserving solutions, particularly in the context of sensors such as audio that may capture privacy-sensitive information. Audio-based ambient sensing approaches are increasingly prevalent in various application domains, such as personal health monitoring [119, 133, 167], ambient environmental sensing [22, 37, 104, 155] and energy efficiency optimization [49, 211] showcasing its growing significance in enhancing our overall quality of life. However, the increased reliance on audio data in these applications has raised substantial privacy concerns, especially considering the inherently sensitive nature of audio data and its potential to capture human speech. To address these concerns, a common strategy is to apply a combination of audio data featurization and speech filtering approaches, preferably at the edge, to reduce privacy concerns while still ensuring the utility [49, 129, 133, 143, 155]. These methods aim to process raw audio signals locally, extracting useful information while preventing speech-related data from leaving the device. However, with the emergence of deep-learning-based Automatic Speech Recognition (ASR) models [21, 92, 181], new privacy challenges have arisen. While featurized audio files may not be intelligible to humans, ASR models trained on such data can potentially reconstruct speech content.

In this chapter of the thesis, we systematically evaluate the privacy risks and utility trade-offs associated with Fast Fourier Transform (FFT) techniques applied to audio sensor data in general-purpose sensing systems. Audio sensors are particularly valuable for activity modeling but also pose privacy risks by potentially revealing sensitive information such as conversations. FFTs are commonly used in audio processing to transform raw data into frequency domain features suitable for machine learning applications [133]. While these techniques have demonstrated effectiveness in protecting audio privacy, modern ASR models like Wav2Vec [21] and Whisper [181] challenge the assumption of privacy. Our analysis of existing privacy-focused filtering approaches [104, 129, 133, 155] reveals residual speech information that fine-tuned ASR models can still recognize, highlighting ongoing privacy risks despite filtering efforts.

To overcome these challenges, we present Kirigami, a lightweight edge-compatible speech

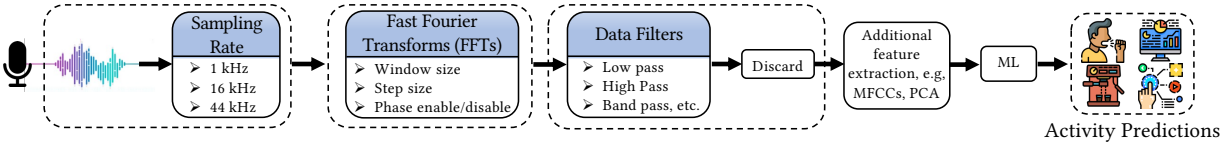


Figure 5.1: An architecture of an audio-based activity recognition approach that takes featurized audio as inputs for applications such as cough recognition or event detection

filter that effectively removes probable speech content while preserving non-speech content to maintain high utility value for activity recognition applications. Unlike existing solutions that may still reveal residual speech information, Kirigami takes a more conservative approach of completely discarding likely audio content on the edge. For this reason, we believe that Kirigami will remain effective even as ASR models become increasingly sophisticated in the future. Furthermore, we demonstrate Kirigami's effective adaptation to real-world environments through an innovative background masking method, enhancing its ability to filter ambient sounds before processing speech events. Moreover, Kirigami allows adaptability through custom post-filter featurization methods, allowing users to users to customize the filter for specific application needs using techniques like Log-Mel Spectrogram and Mel-Frequency Cepstral Coefficients. This flexibility enhances Kirigami's applicability across a range of scenarios. Our results demonstrate that Kirigami is highly effective in suppressing human speech inference even when using fine-tuned ASR models. Kirigami can run on low memory, is computationally efficient, and has been tested and verified on embedded hardware platforms, making it a viable solution for real-world IoT use cases.

The rest of this chapter describes the preliminary background exploration of previous works that enhance privacy in the audio sense, recent progress in speech recognition, and our initial approach to addressing the systematic characterization of privacy and utility tradeoffs.

5.1 Challenges

In this section, we introduce common audio-based ambient sensing solutions and discuss their architecture, focusing on their feature engineering approaches to denature the audio data.

5.1.1 Microphone for Activity Recognition

Microphones are widely used in ambient sensing to support various applications surrounding Human Activity Recognition(HAR), such as health monitoring [133, 144, 234, 244], monitoring the number of people present in a building, and identification of room occupancy and activities of people [49, 132, 206]. In addition, audio capture can be utilized for assistive services, particularly for populations with hearing disabilities, where it can be used for audio scene analysis, audible event alerts, and new wearable devices that work in conjunction with microphones inside smart buildings [174]. In such applications, the collected ambient audio data is often denatured to ensure sensitive information, such as speech, is not collected or sent to the cloud.

A typical ambient sensing solution with a microphone consists of three main sub-components: *audio sub-sampling*, *audio featurization*, and *data filters*, as illustrated in Figure 5.1. The audio sub-sampling component records the time domain ambient audio at different configurable sampling rates based on the application. This raw audio data is then passed through an audio featurization algorithm to convert them into frequency domain signals to reduce the data dimensionality and in some cases to denature the data. In general, Fast Fourier Transforms (FFTs) are used to convert the raw audio signal into a frequency domain representation critical for extracting useful information from the audio signal. Next, various data filtering approaches, including low pass filters, can be employed to eliminate mid to low-frequency bands containing speech information or background noise from the processed signals. The filtered FFT data is then subject to further feature extraction using various audio processing techniques such as Mel Frequency Cepstral Coefficients (MFCCs), filter banks, or spectral features. These extracted features are then used by ML models to classify and recognize audio based events or activities.

Privacy v.s. Utility Tradeoffs for Speech Filters: While featurized data can be sent to a cloud backend, where speech can then be filtered, it is generally considered better to do this on the edge [35] for privacy to prevent speech data from being sent in the first place. In addition, any filtering approach needs to balance privacy (i.e. detecting actual speech segments) and utility (i.e. avoiding filtering non-speech segments) to be useful for real-world activity recognition. In general, audio-based speech filtering approaches can be categorized into four types: *time domain based*, *frequency domain based*, *feature-based*, and *model-based*.

Time domain-based filtering involves techniques such as reducing the audio signal's sampling rate or calculating statistical values such as minimum, maximum, and std-dev for a time period of values. While doing so may be effective at protecting speech privacy, it reduces the utility for downstream HAR applications since sub-sampling can remove important features of the audio signal (e.g. high frequency signals) and not provide enough information to detect activities. Feature-based filtering, extracts specific speech features to speech, such as a spectral envelope or harmonic structure, and use them to remove speech segments. This approach can still affect utility, as it may filter out non-speech sounds. Alternatively, a model-based approach uses ML models to recognize and filter speech. This approach can be highly effective but is computationally expensive, and not available on edge devices with limited computational power and storage.

5.1.2 Phonemes in Audio

A phoneme is a perceptually distinct unit of sound, that can distinguish one word from another in a specified language. When speaking, various phonemes can be produced by adjusting the air passage in the vocal tract. Consonant sounds result from restricting the airflow, such as using different lip, tongue, or teeth positions, whereas vowel sounds occur when the airflow is less restricted and the mouth is more open [163]. Understanding the phoneme structure of a language is crucial in various areas, such as linguistics, speech recognition, and natural language processing. Typically, the English language is composed of 44 phonemes, including 24 consonants and 20 vowels [34].

The 39-phoneme set illustrated in Figure 3, derived from the Carnegie Mellon Pronouncing Dictionary (CMUdict) [10], is widely employed in various ASR and NLP applications. Figure

aa *odd*	ae *at*	ah *hut*	ao *ought*	aw *cow*	ay *hide*	eh *ed*	er *hurt*	ey *ate*	ih *it*	iy *eat*	ow *oat*	oy *toy*
b *be*	ch *chip*	d *dog*	dh *thee*	f *fee*	g *green*	hh *he*	jh *gee*	k *key*	l *leaf*	m *me*	n *knee*	ng *ping*
p *pee*	r *read*	s *sea*	sh *she*	t *tea*	th *theta*	uh *hood*	uw *two*	v *vee*	w *we*	y *yield*	z *zebra*	zh *seizure*

Figure 5.2: The 39-phoneme table of CMUdict [66]

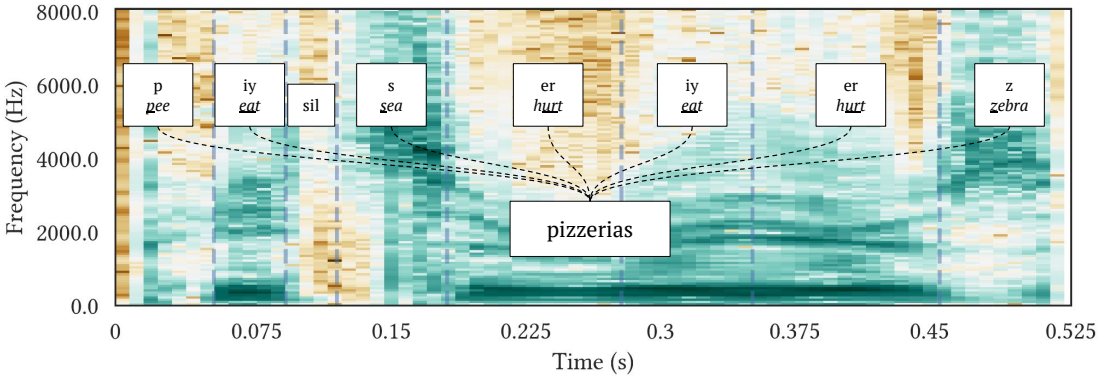


Figure 5.3: The phoneme and grapheme of an example word "pizzerias".Phonemes are the individual speech sounds composing words, while graphemes are the corresponding letters or letter groups representing those sounds.

5.3 shows the unique frequency spectrum signature that indicates the corresponding phoneme. For example, the frequency spectrum for the sound *z* shows that almost all of the frequency spectrum values are activated in comparison to the other phoneme information. Overall, phonemes as features play an important role in speech recognition tasks.

5.1.3 Deep-learning-based Automatic Speech Recognition Models

Recently, there have been significant strides in Automatic Speech Recognition (ASR) models making them even more accurate. Deep learning models, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer models, have been trained on large speech corpora and have demonstrated state-of-the-art performance on a variety of benchmark datasets [80, 170]. More recently, large ASR models such as Wav2Vec [21] have been trained on several hundred thousand hours of multilingual speech data, increasing their robustness to different accents, background noise, and diverse languages. Leveraging the advantage of pretraining, these models can quickly learn general representations of speech patterns and phonetic features (see Fig 5.3), even with low-fidelity or denatured data, as well as handle variations in speaker accent and speech rate. For example, Whisper [181] combines an encoder-decoder model with a context network to improve the modeling of long-term dependencies in speech. Wav2Vec uses contrastive learning to train a model to distinguish between a correctly aligned speech segment and a randomly sampled corrupted segment [21]. Similarly, other transformer-

based models such as BERT [65] and GPT models [77] can also be fine-tuned to achieve impressive results for many down-stream tasks, including ASR [101, 246]. These advancements pose a significant challenge to edge speech featurization, and filtering approaches as modern ASR systems can be fine-tuned to potentially identify speech content even from transformed audio data, as we show in this paper.

5.1.4 Threat Model

Based on our review of the relevant literature on audio privacy techniques used in several activity recognition applications [37, 104, 155, 234], we consider a sophisticated adversary (following the Dolev-Yao model [67]) who has full knowledge of the audio featurization methods used by the device or application. We also assume the adversary wants to extract speech from the featurized and transformed data sent by a device that senses audio. We assume that the adversary does not have direct access to compromise the device itself (i.e., it cannot change its firmware). We also assume that the adversary has the knowledge to replicate the same audio transformation on the speech datasets to use as training data. The adversary can also try to invert any transformation using approximation techniques, for example, using inverse Fourier Transforms or inverse Principal Component Analysis (PCA). The adversary has access to public speech datasets, such as TIMIT [80] and Librispeech [170], to public ASR models [186], and can even fine-tune these models. Finally, we assume that the adversary has access to featurized/filtered audio. In a real-world scenario, an adversary capable of launching such an attack can be, for example, an application that uses audio data to identify activities such as cough, an honest-but-curious audio-to-HAR cloud API service provider, or an external hacker. Based on these assumptions, we consider three potential adversarial scenarios:

S1 - Scenario requiring low effort: An adversary downloads a pre-trained ASR model (no fine-tuning). Then, they try and reverse-engineer the featurized and filtered audio data using an inverse PCA or inverse FFT. The resulting data is in a format that the various ASR models expect, and the adversary passes the data to them to infer speech segments.

S2 - Scenario requiring moderate effort: An adversary downloads a pre-trained ASR model. They fine-tune the ASR model by replicating the same audio featurization techniques used on an annotated speech dataset to create a training set. The adversary also creates a pipeline to transform the featurized audio data into the same shape as the ASR model requires. During training, the adversary re-trains a small subset of the layers on the ASR model with pre-trained weights loaded while freezing the gradients of the rest of the model. Finally, the adversary processes the target audio data into the same shape dimensions as the original ASR model to infer speech.

S3 - Scenario requiring high effort: An adversary downloads a pre-trained ASR model. To fine-tune the ASR model, the adversary replicates the audio featurization techniques on an annotated speech dataset to create a training set. The adversary also creates a pipeline that can transform the featurized audio data into the same shape as the ASR model requires. The adversary trains *all the layers* on the ASR model with pre-trained weights loaded. Finally, the adversary processed

the target audio file into the same shape dimensions as what the original network was trained on.

5.2 Background

Prior works that use audio for activity recognition have proposed different methods to protect audio privacy while preserving the utility of detecting activities.

Table 5.1: Summary of evaluated speech filtering approaches

Filtering Approach	Fourier Transform Configuration		Filter Type	Filter Summary
	Window-Size	Stride-Size		
CoughSense [133]	512	256	Feature	STFT concatenation (150ms), PCA (10 components)
Synthetic Sensors [129]	256	128	Frequency Domain	Reduced FFT (10 windows/s)
PrivacyMic [104]	256	128	Frequency Domain	Low Pass Filter (<300 Hz)
SAMoSA [155]	600	30	Time Domain	Subsampling (1kHz)

5.2.1 Audio Privacy Filters for Activity Recognition

Researchers have proposed various audio filtering methods to remove speech information from audio, including data degradation techniques to sample FFTs at a lower rate or completely drop FFT data from a certain frequency band. These approaches aim to protect user speech data while allowing other audio data that are useful for activity recognition, but their efficacy varies significantly, and their limitations must be considered. Table 5.1 shows the summary of different speech filtering approaches presented in the prior work.

Coughsense [133] is one of the earliest works to propose a cough detection system that utilizes a low-cost microphone to detect coughs accurately in real-time. They reduced speech intelligibility by aggressively aggregating 150ms of sound and extracting ten principal components from principal component analysis on cough sounds. They showed that their system could classify coughs with high accuracy. Iravantchi et al. [104] proposed a daily activity recognition system that utilizes inaudible frequencies in the audio signals to preserve privacy. Such an approach requires special microphone sensors that capture ultrasonic and infrasonic sounds and the usual microphone that collects sounds in the audible range. Filters can be implemented on microphone hardware to filter out frequencies from 300 Hz to 8kHz.

Another line of work focuses on more generic transformations to hide speech in audio. Chen et al. [46] suggested a method to filter speech from audio by replacing the vocal tract transfer function of vowel regions in audio with the transfer function from prerecorded vowels. SoundShredding [126] proposed a privacy-preserving audio transformation in which the order of frames from MFCC features is randomized. The commonly used method of evaluation in these approaches includes recruiting participants to listen to the processed audio clips and examine if any speech content can be picked or to rate the extent of clarity of the audio clips. Another approach is to pass through an existing speech-to-text service such as Google Speech Recognition. While the ability to recognize speech was shown to be limited after using these transformations was evaluated under human listening experiment [46, 104, 133, 155] or passing through existing speech-to-text API [104, 155], the threat from recent powerful machine-learning-based ASR models was not considered.

5.3 Feasibility of Inferring Speech From Filtered Audio

Two reasons to evaluate the feasibility of ASR models to infer speech from featurized audio are: (a) our observation of the residual phoneme information available in the output filtered data from prior approaches, and (b) the ability of the deep learning-based ASR models to be fine-tuned and learn from featurized data. In prior approaches, a speech filter such as a time or frequency-domain filter is applied to an audio signal. These approaches often remove certain frequency components associated with human speech. However, not all phoneme information is removed, and some residual information may remain in the filtered signal. Figure 5.4 shows the spectrogram of the data after prior filtering approaches are applied. This residual phoneme information can be seen in the form of different acoustic characteristics, such as spectral shapes specific to spoken words or phonemes. For example, the FFT output after applying the CoughSense filter [133], for the phoneme *"iy"* has unique spectral patterns still present around the 1 kHz frequency range. Similarly, for PrivacyMic [104], we can see that unique spectral patterns are present around the 250 Hz frequency range for the same *"iy"* phoneme. An adversary can exploit this residual phoneme information by using ASR models. An ASR model trained on raw audio can still be fine-tuned for featurized audio and does not need to be trained from scratch. These models can learn complex patterns and representations from data through training over a large speech data set such as TIMIT [80].

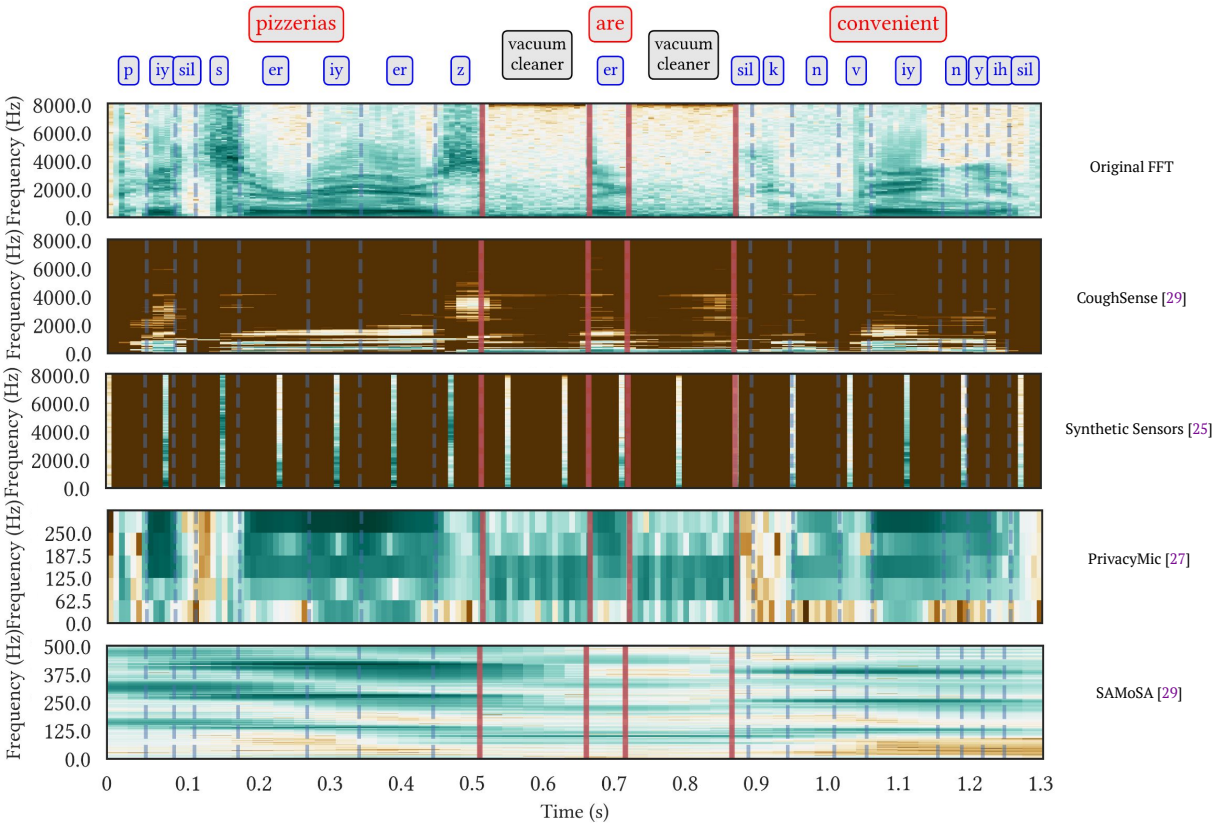


Figure 5.4: FFT Spectrogram of several privacy filters proposed by prior work.

Next, we describe all ASR models that we evaluated and detail the procedure to mimic speech inference attacks. We tested two kinds of models based on the type of inferences the models made: phoneme and word. We summarize the ASR models in Table 5.2.

Table 5.2: The list of evaluated ASR models against audio privacy filters.

Attack Model	Inference Type	Pre-Training (Hours)	Fine-Tuning (Hours)	# Parameters	Metric
CRDNN	Phoneme	5 (labeled)	5 (labeled)	10M	PER
Wav2Vec Transducer	Phoneme	53.2k (unlabeled) + 960(labeled)	5 (labeled)	318M	PER
Whisper AI (pretrained)	Word	680k (labeled)	0	769M	WER
Whisper AI	Word	680k (labeled)	5 (labeled)	769M	WER

5.3.1 Fine-Tuning Phoneme-based Speech Inference Models

Phoneme prediction models can be utilized to infer phonemes from featurized audio data. As opposed to word-level speech recognition, phoneme recognition offers the benefit of having considerably fewer prediction targets (e.g. 39 phonemes in CMU-Dict [66]), alleviating the concerns about the size of the vocabulary [39]. Moreover, we speculate that if only part of a word can be inferred, phoneme-based models might provide a chance for the human attacker to infer the complete word based on the context with only the predicted parts.

Convolution Recurrent Deep Neural Network (CRDNN): The CRDNN model combines Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Multi-Layer Perceptrons (MLPs). The CNN layers extract features from spectrograms of raw audio, while the RNN layers allow the network to find sequential information for phoneme prediction. Connectionist Temporal Classification (CTC) [92] is integrated into the architecture, allowing the model to handle varying lengths of input and output sequences without requiring explicit alignment.

We adopted the CRDNN model to infer phonemes from filtered speech data based on implementation from SpeechBrain [186]. We convert the audio features obtained from different audio filtering techniques, through rescaling and cropping, into spectrogram-shaped representations with 40 features at each time window.

Table 5.3: Example inference results and Phoneme Error Rate (PER) from the CRDNN model.

Original Segmented Phonemes	Privacy Filters	Predicted Segmented Phonemes	PER
pizzerias are convenient for quick lunch p-iy s-er-iy-er-z er-k-n v-iy-n-y-ih f-aa-r-k-w-ih k-l ah-n-ch	CoughSense [133]	p-iy-t-ih-ay iy ih-z-ih-k-m b-ih-n-y-ih f r-ay-k-w-ih m-ah-n-s	44.12%
	Synthetic Sensors [129]	s-t-ih-r-iy ih-z-k-ah-m p-iy-n f-aa-r-k-w-ih-l-ah-n s	50.00%
	PrivacyMic [104]	p-iy p er-r-iy-ih-z-ih k-m-b-ih-l-ah f-aa-r-t-r-ae n-ah-jh	52.94%
	SAMoSA [155]	dh-ah-p-r-aa p-er-d-ih-s p-l-ih-n t-ih-k ih-n-d-ih-s t-r-ey dh-ah-p-r-aa p-er	97.05%
december and january are nice months to spend in miami d-ih-s-eh-m b-er-ng-jh-y-ae-n y-uw-er ih-n ay-s-m-ah-n-th s-t-ih-s p-eh-n-ih-n m-ay-ae m-iy	CoughSense [133]	d-ih-s-ih-m-er-z eh-n-iy er m-aa-s m-ah-n s-d-ih-s p-ah n-ih-m aa-ih n-iy	45.45%
	Synthetic Sensors [129]	dh-ih-s-ih m b-er-ih-z eh m r-eh r-ih-n-ay-s m-ah-n t s p-r-ih-n-ih m-ay-ih-n m-iy	43.18%
	PrivacyMic [104]	dh-ah s-ih-ng-g-er-n jh-eh-n-er-l-iy ih-m-ay-s m-ih-n-s t-ih-s-p-ih-n-ih-ng l-ay-b-l-iy	45.45%
	SAMoSA [155]	dh-ih-s-p-aa-r k-ih-n-t-ih k-s-p-er d-ih-s-t-r-ey dh-ih-s-p-eh-r-ih k-ih-n-t-ih k-ih-n	77.72%
basketball-can-be-an-entertaining-sport b-ae-s-k-ih b-aa-l-k-ih-n b-iy ih-n eh-n-t-er ch-ey-n-ih-ng s-p-aa-r	CoughSense [133]	b-r s-t-ih b-r-k-ih-n b-iy-ih-n t-ih k-ey-m-ih n-s p-r-ay	32.43%
	Synthetic Sensors [129]	dh-ae-f-ih-l-aa k-ih-n m-ey-n er s t-ey n-ih-ng-z b-aa r	48.65%
	PrivacyMic [104]	dh-eh-s-t-ih b-ae-k-ih-n w-ah-n ih-n-t-er t-uw ih-n-iy s-t-r-ey s	54.05%
	SAMoSA [155]	dh-ih-s-p-aa-r t-ih s-p-l-ih-n t-r-iy-k ih-n-d-ih-s t r-iy-k-ih-n	72.97%

Transducer with Pretrained Wav2Vec 2.0: Transducer models, also known as RNN-T (Recurrent Neural Network Transducer) models, are a type of end-to-end ASR system that directly map input speech features to target text without requiring any explicit alignment between them [91].

These models consist of an encoder, a decoder, and a joint network, which together predict the output sequence in an autoregressive manner. Wav2vec 2.0 is a self-supervised pretraining method that learns powerful speech representations from raw audio waveforms by exploiting the temporal structure of the data [21]. Studies have shown that a pre-trained Wav2vec model leads to better performance than using handcrafted features, such as the Mel-frequency cepstral coefficients (MFCCs) or filter banks, as the ASR models can benefit from the rich and expressive features that wav2vec learns from large amounts of unlabeled audio data [75, 186, 226].

We used an implementation of the Transducer model from SpeechBrain [186] and fine-tuned the model starting from an existing checkpoint trained using the TIMIT dataset [80]. We used the Wav2Vec2-Large-LV60 to extract features from audio inputs, which contains 317M parameters and is pre-trained on 53.2k hours of unlabeled audio data and 960 hours of speech data [21]. While the resulting filtered audio from many audio privacy-focused featurization techniques transforms audio into the frequency domain, with the Wav2Vec 2.0 encoder, the Transducer model takes waveforms as inputs. To make the model compatible with the spectrogram-like shape resulting from different featurization approaches, we applied the Inverse Fast Fourier Transform (IFFT) to obtain a waveform representation from the spectrogram-shaped representations. All weights of the model are fine-tuned using the TIMIT dataset.

Phoneme Post-Processing: Directly interpreting the phoneme outputs might still be challenging for inexperienced adversaries. To enhance the speech inference practicality and to better understand the privacy implications of the tested audio filtering techniques, we perform the following post-processing on the phonemes output. Our first step is to segment the phoneme predictions into groups, in which each group of phonemes likely represents a word. We trained a bidirectional LSTM sequence tagging model to segment the phonemes using the ground truth TIMIT phonemes and words, which achieved 98.6% per-tag accuracy. In addition, we used a heuristic-based approach that breaks at 'sil' (silence) phonemes unless there are less than four consecutive predicted non-silence phonemes in prior, which is likely due to prediction error. After segmenting the entire phoneme sequence prediction, we used Pincelate [178], an open-source tool that performs phoneme-to-grapheme and grapheme-to-phoneme conversion, to spell the probable word for each segment of phonemes. Although the spelling of the sentence is imperfect due to the phoneme inference errors, it still provides hints to infer the speech content. Table 5.3 shows the inferred segmented phonemes using the CRDNN model. In some cases, even after filtering, the inferred phonemes can sound very similar to the original sentences. For example, when using the CoughSense [133] filtering approach, the CRDNN model was still able to capture *p-iy-t-ih-ay iy ih-z*, which is very close to the pronunciation of the word *pizzerias*. When the PER value increases, transcribing the words is not as straightforward as one needs to spell the words and consider which phonemes might be incorrect predictions. The output *dh-ah s-ih-ng-g-er-n* for PrivacyMic [143], for instance, still sounds similar to the word *December*, but directly identifying the word without knowing the original sentence can be challenging.

5.3.2 Fine-Tuning Word-based Speech Inference Models

Whisper is a Transformer-based encoder-decoder model, more commonly known as a sequence-to-sequence model [181]. Unlike Wav2Vec, which was primarily trained on unlabelled data in

Table 5.4: Sample speech inference results and Word Error Rate (WER) from the Whisper model

Original sentence	Privacy Filters	Whisper (Fine-tuned)	WER
pizzerias are convenient for quick lunch	CoughSense [133]	pitcheeers are convenient for a quick lunch	33.3%
	Synthetic Sensors [129]	his barriers continued to overlap	100%
	Privacy Mic [104]	peculiar is a conveyor for a quick lunch	83.3%
	SAMoSA [155]	people often go for in quick evening	83.3%
december and january are nice months to spend in miami	CoughSense [133]	decembers are nice mountains to spend in miami	40%
	Synthetic Sensors [129]	december and jan are make sure you save money to visit my website	90%
	Privacy Mic [104]	the figure here may attach to the spring and water	100%
	SAMoSA [155]	decide and jan are moving to the may	70%
basketball can be an entertaining sport	CoughSense [133]	basket bowl can be an immediate sport	50%
	Synthetic Sensors [129]	basketball can be found in video game	83.3%
	Privacy Mic [104]	bask be an enter	66.7%
	SAMoSA [155]	ballers are on an extreme sport	66.7%

an unsupervised manner, Whisper was trained on 680k hours of labeled speech data, such as LibriSpeech [170] using extensive supervision with 769M parameters. We used the pre-trained Whisper *medium* checkpoint and then fine-tuned the model to the different types of audio features obtained from the prior filtering techniques discussed in the previous section. Since the Whisper model expects log-Mel spectrogram as input, we convert the audio features obtained from different audio filtering techniques to log-Mel spectrograms. We then use this information to fine-tune the model. During the fine-tuning step, Whisper's parameters are updated to match the specific characteristics of the target word prediction, such as its phonetic spectral properties.

Table 5.4 shows examples of sentence predictions from Whisper pre-trained and fine-tuned ML models when we apply different filtering techniques. In certain cases, the predicted sentences are similar to the original sentences. For example, in certain instances, even after applying the CoughSense [133] filtering approach, the fine-tuned whisper model successfully predicted all the words except for "*pitcheeers*" in the example "pizzerias are convenient for quick lunch." Furthermore, it is observed that as the WER value increases, the distinction between WER values becomes less clear in terms of what information they may reveal. For instance, although an example sentence prediction from PrivacyMic has 90% WER, the prediction from Synthetic Sensors with the same WER still reveals some of the original speech information (such as words like "december" and "jan"). In addition, we found WER may appear high for short sentences, due to the number of words normalizing it, and thus predicting only a few incorrect words will be enough to raise WER to a high value.

5.3.3 Need for PER and WER contextualization

Notably, as the above results show PER and WER values by themselves are only part of the story in terms of understanding the potential privacy concerns with the parts of the original speech that may still be reconstructed using ASR approaches. In addition, all the words in a sentence are not the same in terms of what they reveal about the conversation and different sentences with similar WER/PER values may lead to less (or more) privacy concerns. Finally, the data and examples for different featurization approaches mentioned in this section are merely illustrative to show what is possible by re-tuning some of the ASR models. In Section 5.5.2 we provide a detailed evaluation of Kirigami as compared with various prior approaches on a larger corpus of speech data in terms of average PER and WER values. Furthermore, to contextualize different PER and WER ranges in terms of what they can still reveal, we performed a separate user study, the results

of which are reported in Section 5.5.4.

5.4 Kirigami: Lightweight Speech Filter

As shown in the previous section, prior approaches on preserving user privacy are susceptible to inferring speech with the latest state-of-art fine-tuned ASR models. A key reason for this is that these approaches focused on degrading data or utilizing feature-reduction strategies to filter potential speech segments. However, modern ASR models such as Whisper [181] are trained on a broad spectrum of acoustic features and linguistic contexts that can take advantage of any residual speech information, such as phonemes, making them less susceptible to conventional privacy-preserving techniques. More importantly, as the development and optimization of ASR models progress in the future, their reliance on any residual speech segments to enhance ASR performance increases, highlighting the necessity for new strategies in preserving privacy. Consequently, our approach focuses on the detection and removal of data segments containing speech-related information, including phonemes. This ensures a more robust mechanism to safeguard user privacy within the evolving realm of ASR technology. Our design of Kirigami is based on a set of key insights. First, the detection of speech information (phonemes) can be modeled as a binary classification task, for which shallow machine learning models may suffice in terms of reasonable accuracy. Second, these shallow ML models can be deployed on a wide variety of hardware as they are memory and computationally efficient. Third, the Kirigami filter can promptly discard detected speech segments at the edge to safeguard speech privacy, allowing full-featured FFT data to pass through when non-speech segments are detected, thereby optimizing utility performance.

5.4.1 Machine Learning-based Kirigami Speech Filter

Our proposed solution of a speech filter on the edge involves constructing a lightweight yet efficient real-time speech detector. The overall objective of the speech detector is to classify each time frame of the Short-Time Fourier-transformed (STFT) audio data as either speech or non-speech. Formally, let $X \in \mathbb{R}^d$ be a d-dimensional vector representing a time frame of STFT data. For example, $d = 128$ when the window size of the STFT is 256. The task of the speech detector is to learn a mapping $f : \mathbb{R}^d \rightarrow 0, 1$, where $f(X) = 1$ indicates speech and $f(X) = 0$ indicates non-speech. Once a time frame is identified as likely speech (i.e., $f(X) = 1$), that particular time frame is discarded.

In Kirigami, we use a Logistic Regression (LR) model for real-time speech detection. Logistic Regression is a well-established shallow ML model typically used for binary classification tasks and is also resource-efficient. To build an LR model, we first normalize the STFT features using the L-1 norm across all frequency components for each time frame. The normalization step ensures that the influences of volume variation from the audio signal are reduced.

Thus we formally represent the Logistic Regression model used for binary speech detection as follows:

$$g(X) = \sigma(W \cdot \frac{X}{||X||_1} + \beta_0) \tag{5.1}$$

, where $||X||_1 = \sum_{i=1}^{d} |x_i|$ is the L-1 norm of X, W and β_0 are the weight coefficients and intercept learned from the training data, and σ is the logit function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{5.2}$$

The decision $f(X)$ of whether a time frame should be removed is based on comparing the model prediction against a threshold value τ, which can be represented as:

$$f(X) = \begin{cases} 1 & \text{if } g(X) \geq \tau \\ 0 & \text{if } g(X) < \tau \end{cases} \tag{5.3}$$

In our training process, the LR model was developed using the TIMIT [80] dataset for speech data and the ESC50 [177] environmental sound dataset for non-speech data. We opted for the TIMIT dataset for speech data due to its inclusion of multiple hours of phonetically transcribed sentences, enabling in-depth analysis and modeling of speech sounds. Additionally, we opted for the ESC50 dataset for non-speech data, aligning with our objective of recognizing events and activities. The sound samples from ESC50 provide valuable training data for the LR model to effectively distinguish and preserve sounds associated with various activities. To enhance the diversity of our dataset, we created an additional speech dataset where we overlay sounds from ESC50 on top of the TIMIT speech audio. This augmentation aims to enrich the training data, enabling the LR model to better generalize and perform effectively across a range of real-world scenarios. We apply STFT to the audio samples from these sources to transform them into the frequency domain (FFTs). Subsequently, we label each time frame as positive (i.e., speech) or negative (i.e., non-speech) depending on the source. We balanced the dataset to have an equal number of speech and non-speech samples. In total, our dataset comprises 20000 samples, which are randomly split into three subsets: 80% for training, 10% for validation, and 10% for testing. Through supervised training, the LR model learns to classify each time frame as speech or non-speech, thereby removing the time frames that are likely speech. Overall, our Kirigami LR model (using $\tau = 0.5$) achieved a speech recognition accuracy of 76.44%, indicating the effectiveness of our method in accurately identifying and classifying speech segments. It's important to note that we are not aiming for perfect classification accuracy, and our goal was to make the model configurable to balance privacy or utility requirements, depending on the use case. This adaptability allows for a nuanced and tailored approach, where the balance between accuracy and the desired outcome can be fine-tuned to align with the overarching goals of the application or system. We elaborate further in Section 5.4.2 on how the Kirigami LR model can offer sufficient privacy protection with appropriate threshold values while preserving adequate utility value.

5.4.2 Configuring Privacy vs. Utility Tradeoffs

Figure 5.5 provides an illustration of the trade-off between privacy and utility as we configure the Kirigami filter to have different values for τ. In the figure, the first part contains pure speech ($t = 0s$ to $t = 1.86s$), speech data overlaid with a vacuum cleaner sound ($t = 1.86s$ to $t = 3.72s$), and a vacuum cleaner sound ($t = 3.72s$ to $t = 5.6s$).

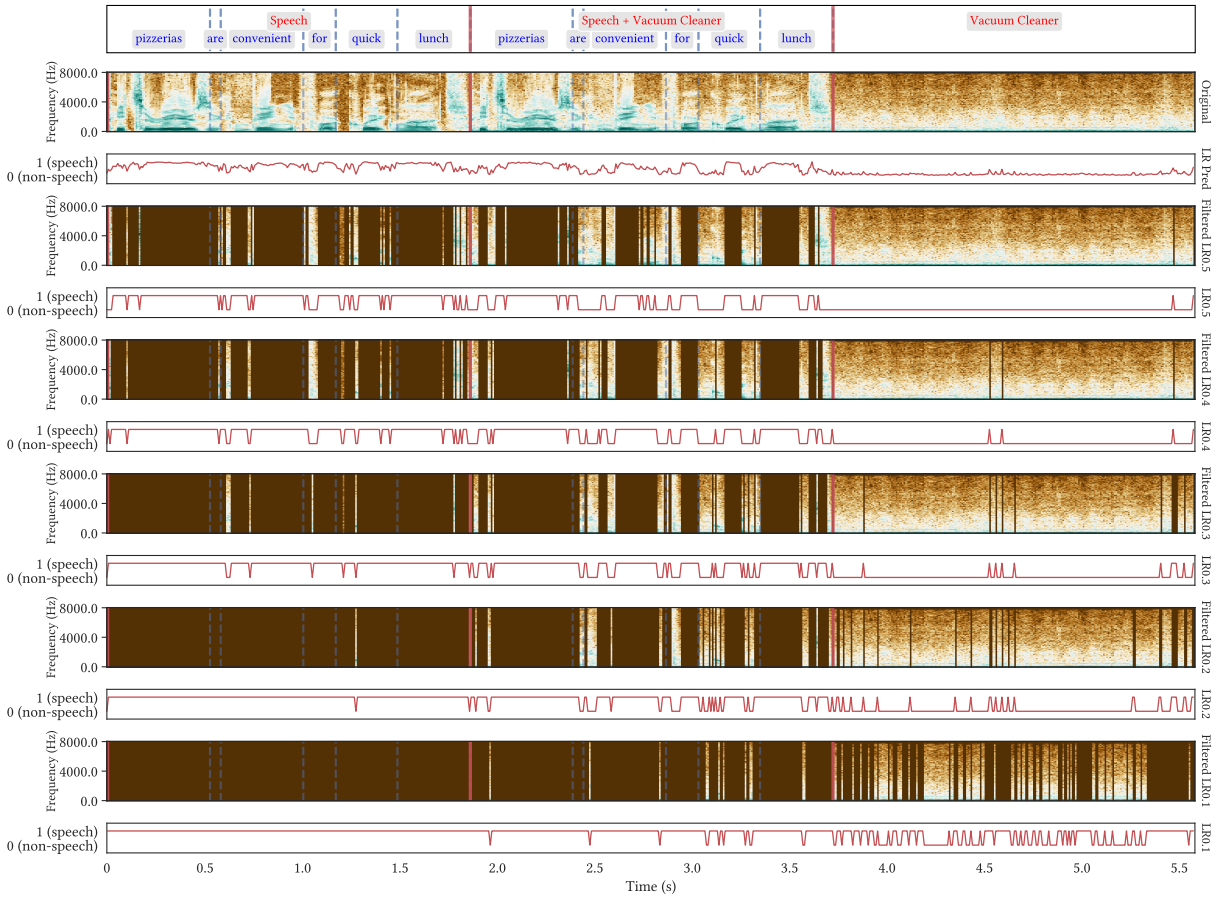


Figure 5.5: Illustration of Kirigami's Logistic Regression model for phoneme prediction. The LR Pred line graph shows the predictions from 0 to 1, while LR0.5 to LR0.1 shows predictions at threshold values indicating speech (1) or non-speech (0). The original and filtered spectrogram demonstrate a balance between privacy (speech filtering) and utility (activity recognition).

The value of the threshold τ, configurable to be between 0 and 1, plays a crucial role in determining the model's inclination towards either preserving privacy or maintaining utility. A value closer to 0.5 leans toward balancing both. Given that the Kirigami filter is an ML model, there are instances where it is incorrect, which leads to either some speech data being leaked or some audio event data being filtered. For example, for the LR0.5 configuration, some frames with the word "quick" are mistakenly classified as non-speech (a false-negative). As the threshold changes from 0.5 to 0.1, the model becomes more conservative and prioritizes privacy protection. For example, for the LR0.1 configuration, all the segments with the word "quick" are now detected correctly as speech, but towards the end, numerous segments with the vacuum cleaner sound alone are incorrectly filtered out as speech (false positives), which can affect the utility of activity detection. The optimal threshold depends on the specific use case and the application requirement. In situations where privacy is crucial, such as if the sensor is installed in a private office, a lower threshold would be more suitable. Conversely, in less sensitive contexts (e.g., shared spaces) or where the accuracy of activity detection is more important (HAR for fall detection scenario), a threshold closer to 0.5 may be more appropriate. We further quantify the

impact of different thresholds on privacy vs utility and discuss its implications as compared to various prior approaches in Section 5.5.3 and 5.5.4.

5.4.3 Kirigami Speech Filter on the Edge

A key goal in developing the Kirigami filter was to ensure its feasibility of deployment on edge, which typically implies operating in resource-constrained environments. For instance, popular ARM Cortex M class microcontrollers commonly found in IoT devices have around 128KB RAM 100-150MHz CPUs, and thus cannot run deep learning-based ASR models such as Whisper to filter speech. With all its configurable threshold parameters, this meant that the Kirigami filter needed to be implemented in environments with frugal memory resources and limited computing capabilities.

We quantized the Kirigami LR model to reduce its memory footprint, ensuring more efficient storage and processing on resource-constrained devices. This involved converting the model's floating-point parameters to integers, a process that conserves memory and contributes to improved computational efficiency. Overall, our Kirigami approach requires the storage of 129 weight values, including the intercept, and the computation of one normalization, one dot product, and one logit function. We implemented the Kirigami LR model on the popular edge microcontroller ARM Cortex-M4F with 256 KB RAM and 1 MB flash and measured the memory consumption and latency. Our measured memory footprint of the quantized Kirigami model coefficients was 518 bytes ($<$ 1 KB) and a total of 2.1 KB ($<$ 3 KB) for the entire Kirigami filter, including the model intermediate weight calculations. The end-to-end latency for prediction of an FFT sample is approximately 0.71 ms, demonstrating that the Kirigami filter is not only resource-efficient in terms of memory consumption but also exhibits low latency, making it well-suited for deployment in real-time applications on edge devices with limited computational resources. We implemented the Kirigami LR model in both C and Python to ensure comprehensive compatibility of the Kirigami filter across different device environments and run efficiently on devices with limited computational power and memory. We further evaluate the real-world accuracy performance of our edge Kirigami filter in Section 5.5.6.

5.4.4 Adapting the Kirigami Speech Filter for Real-World Environments

A key design goal of the Kirigami filter is to robustly filter out speech in real-world environments. Our initial hypothesis to achieve this was to train the Kirigami filter on a custom dataset where activities of interest are overlaid with speech events. We formed this dataset by augmenting environmental sounds from ESC50 dataset of activities and ambient sound with the TIMIT speech audio, using this dataset to train the Kirigami model. This approach allowed us to simulate real-world conditions where people speak with other activities and events happening in the background. The Kirigami speech filter, while effective in controlled environments, faced several challenges in these real-world situations, primarily due to the presence of background noise, diverse acoustic landscapes, and variability in ambient noise levels.

Thus, we aimed to identify background or ambient sounds in the environment and filter those out before identifying and filtering speech events. However, real-world environments exhibit a dynamic spectrum of ambient noise, with levels that fluctuate based on factors such as location,

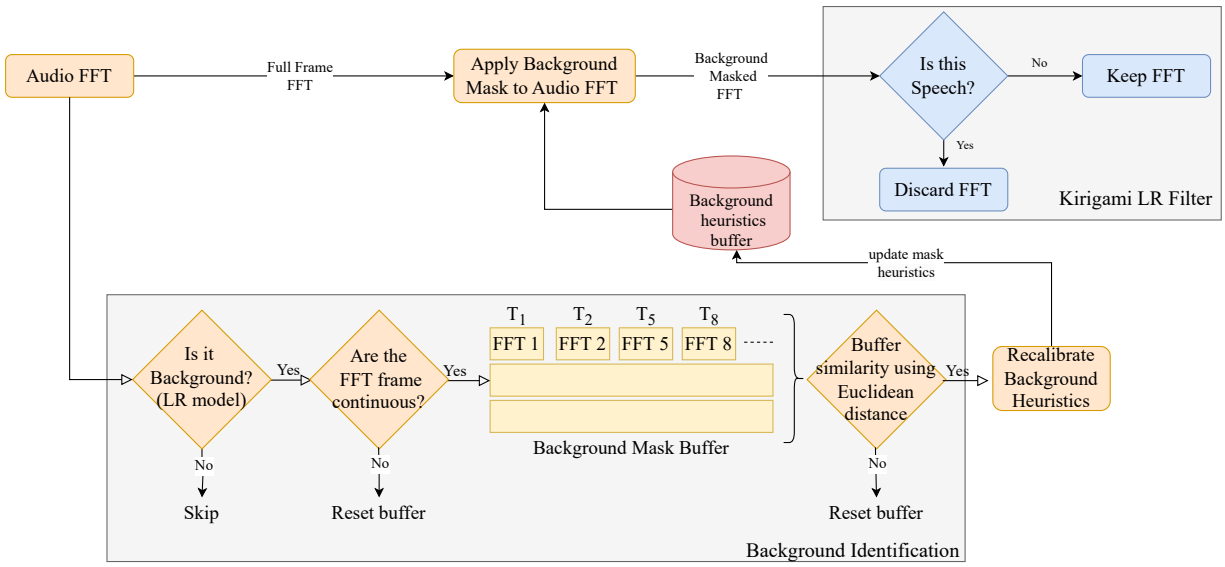


Figure 5.6: Flowchart depicts the adaptive background masking process in conjunction with Kirigami's speech filter. The process involves background detection, buffer comparison, heuristic calculation, and the generation of a background mask to filter out background frequencies. The resulting background-filtered FFT enhances Kirigami's speech filter for improved accuracy by eliminating background noise.

time of day, and environmental conditions. For example, workshop environments may feature machinery traffic sounds, while quieter office spaces may still have variable noise from air conditioners, HVAC, or occasional people chatting in the surroundings. Moreover, background noises, which may be constant or intermittent, span a wide spectrum of frequencies and intensities.

To overcome these challenges, we present an adaptive background masking process combined with Kirigami's speech filter. We continuously collect background noise profiles from the environment, estimate a mask for these background noises, and apply this mask to filter out the noise. As shown in figure 5.6, our approach consists of the following steps: (1) background identification, (2) creating a background mask buffer, and (3) background mask generation and filtering. Once the noise is filtered, the data is sent to our Kirigami LR model for speech filtering. The background identification step uses a Logistic Regression model to predict whether the input FFT represents background noise versus foreground speech or activity of interest. This model is trained on datasets containing a mixture of background noises of typical environments from Microsoft Scalable Noisy Speech Dataset (MS-SNSD) [189] and foreground activities and speech from TIMIT [80] and ESC-50 [177]. We attempted to further increase the real-world fidelity of noise mixtures by overlaying the foreground speech and activities with background noise at various signal-to-noise ratios and various pitches of background noises. Second, these predicted FFT data are added to the background noise mask buffer, which maintains multiple buffers of continuous background FFT data. This buffer imposes conditions on the temporal continuity of background FFT samples, ensuring that the FFT frames within the buffer are contiguous. Once multiple of these buffers are filled up, the similarity across different buffers is gauged using

the Euclidean distance metric. If the buffers are similar, the process generates a background mask. This process ensures the reliability and accuracy of the captured background profile, enabling adaptation to diverse environmental settings. The background mask generation process relies on a method called spectral gating [121]. This technique involves estimating a background threshold (or gate) for each frequency band within the collected background profile, calculated using the mean and standard deviation over frequency. This threshold is then used to compute a mask, which gates noise below the frequency-varying threshold. During the background masking phase, we initiate the process by establishing a gain control for each frequency band. If a frequency surpasses the previously determined threshold, the gain is set to 0 dB; otherwise, the gain is reduced (*e.g.*, to -18 dB) to mitigate background noise. Following this, we use frequency smoothing to ensure that individual frequencies are neither excessively suppressed nor boosted in isolation. We then direct the background-masked FFT to the Kirigami speech filter, which is now potentially less susceptible to the influence of background noise. By incorporating an adaptive algorithm that responds to fluctuations in ambient noise, we enhance the Kirigami filter's versatility in handling variable acoustic environments, ensuring reliable speech recognition performance across diverse real-world scenarios. We evaluate the robustness of our approach in the real world in Section 5.5.6.

5.5 Evaluation

This section evaluates the effectiveness of state-of-the-art speech recognition systems in recovering speech text from the prior privacy-focused featurization approaches. In addition, we evaluate Kirigami's ability to identify phonemes from audio data. Overall, our evaluation aims to answer the following questions:

- RQ1: How accurately do modern ASR-based systems identify speech contents from audio featurized using prior approaches?
- RQ2: How robust is Kirigami 's filter to ASR-based attacks, and how does Kirigami's filtering approach affect the utility?
- RQ3: How accurately does Kirigami's filter perform in real-world environments?

5.5.1 Evaluation Setup

Dataset: We utilize the TIMIT [80] dataset to evaluate the feasibility of inferring speech from featurized data, fine-tuning the ASR-based models, and building the Kirigami filter. The TIMIT dataset contains a total of 5 hours of English speech with 4,620 phonetically transcribed sentences, with approximately ten sentences per speaker. Each sentence is segmented into phonetic units, such as phonemes and words, allowing for detailed analysis and modeling of speech sounds. To evaluate the utility of Kirigami filter and the prior filtering approaches, we use the ESC-50 [177] dataset. The dataset contains 2000 environmental sound recordings from 50 classes involving various sound types, including animal sounds, natural sounds, human non-speech sounds, *etc*. To match the scope and difficulty of the application scenarios in prior audio privacy filtering approaches, we selected ten classes: toilet flush, sneezing, clapping, breathing,

coughing, footsteps, laughing, brushing teeth, snoring, drinking, door knock, washing machine, vacuum cleaner, clock alarm, and clock tick. Finally, we also created an overlay of the TIMIT dataset with the ESC-50 dataset to evaluate privacy and utility performance in a noisy environment. For speech inference evaluation, the overlaid data is produced by overlaying a random sound file from ESC-50 on top of each speech audio file from TIMIT. The resulting audio file contains the same speech content and length as the original. Similarly, for utility evaluation, we overlaid a random speech audio file from TIMIT on each sound file from ESC-50. We match the loudness of two different audio files based on the loudness level in decibels relative to full scale (dBFS).

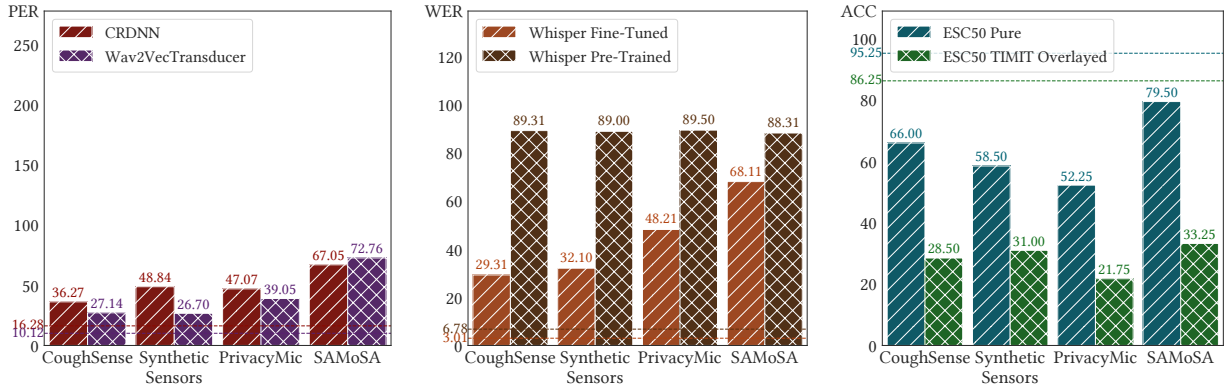
Speech Inference Evaluation: To examine the extent to which modern ASR models can infer speech content information from prior audio privacy filter approaches, we implemented each privacy filter approach and evaluated the speech inference performance. The approaches that we included in our evaluation are CoughSense [133], Synthetic Sensors [129], PrivacyMic [143], and SAMoSA [155] as all these works indicate privacy as a primary factor in their filter design process. We applied each of these privacy filter approaches to obtain a dataset of filtered audio samples. We used four different configurations of ASR models: CRDNN, Wav2VecTransducer, Whisper Pre-Trained, and Whisper Fine-Tuned. The training set of filtered audio samples is used to fine-tune the model weights, which facilitates the ASR model to adapt to filtered audio samples and learn suitable new feature extraction and prediction mechanisms. The models are trained to start from existing checkpoints for optimal speech inference performance.

Privacy Performance Measures: The performance of speech inference is measured in Phoneme Error Rate (PER) and Word Error Rate (WER). PER and WER are defined as the number of insertions, deletions, and substitutions normalized by the length of the target sentence. PER measures the number of incorrect phoneme predictions produced by the ASR model, while WER measures the rate at which words are predicted incorrectly. The evaluation of speech inference measured in PER and WER is conducted on both TIMIT as the pure speech dataset and overlaid dataset, although we adopt the PER and WER on pure speech data as the primary measure of privacy protection.

Utility Performance Measures: Utility, often emerging as an opposing goal to privacy, also needs to be assessed to understand the effectiveness of a privacy filter. An effective filter ideally shall achieve high PER and WER while having minimal loss in the utility performance compared to non-filtered audio. We adopted the Audio Spectrogram Transformer (AST) [85], a state-of-the-art audio classifier and one of the best performers on the ESC50 dataset, to evaluate the accuracy of inferences as the utility performance. We used a 5-fold cross-validation standard to the ESC50 dataset and calculated the classification accuracy of the 10 classes selected from the ESC50 dataset on, both, the pure environmental sound and the sound overlaid with speech. Unlike speech inference where the privacy leakage on pure speech is the primary concern, the classification accuracy on both pure and noisy data is crucial for consistent performance across different environmental conditions. Taking all these measures together allows us to assess the trade-off between privacy protection and utility preservation for each filtering approach.

5.5.2 RQ1: Feasibility of Speech Inference from Prior Filtering Approaches

We fine-tune the ASR models by applying each audio featurization approach to the training set of the TIMIT dataset. The ASR models learn to infer speech from these featurized audio samples through fine-tuning. Finally, the performance of speech inference from all ASR models is evaluated using the PER and WER metrics on the pure speech Timit dataset and Timit speech data overlaid ESC-50 activity data.

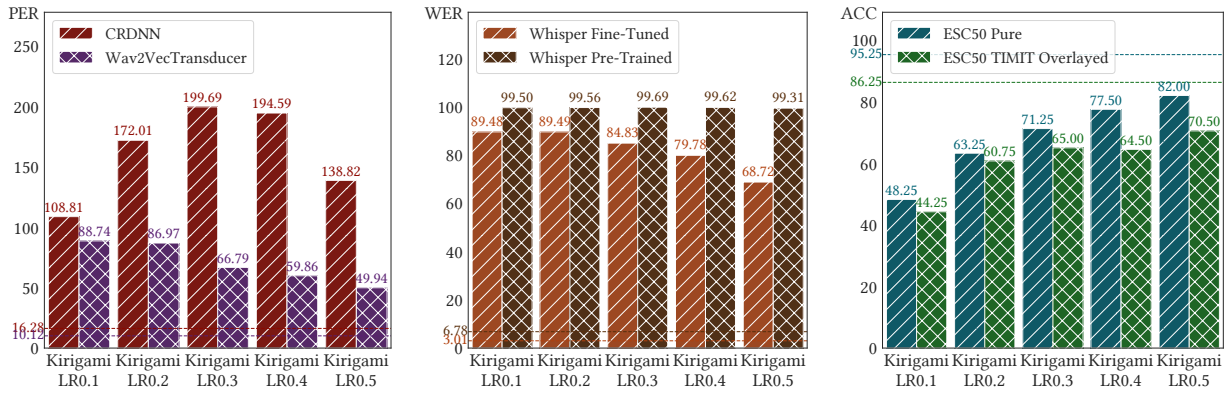


(a) Phoneme Error Rate (PER) (b) Word Error Rate (WER) (c) Classification Accuracy

Figure 5.7: Results of (a) phoneme-based speech inference, (b) word-based speech inference, and (c) activity classification accuracy on prior filtering approaches.

Phoneme-based Speech Inference: Fig. 5.7(a) summarized the experiment results of speech inference on featurized audio data using four prior approaches to audio speech filtering using the CRDNN and Wav2VecTransducer models. Overall, these results demonstrated a concerning level of privacy risks in audio privacy filtering techniques. CoughSense [133], Synthetic Sensors [129], and PrivacyMic [143] showed PER of 27.14%, 39.05%, and 26.70% respectively on pure speech sounds, proving the feasibility in inferring phonemes from the filtered speech data. Wav2VecTransducer outperforms CRDNN in inferring phonemes on these three approaches except on SAMoSA. SAMoSA [155], with simple downsampling and a large FFT window size approach, exhibits adequate protection on speech. We conjectured that this protection might be due to the length of FFT windows measured in time (600ms) far exceeding the time to speak a phoneme in most cases. To help assess the audio filtering effectiveness in comparison to complete audio data, we included our baseline approaches using FFT data from 256/128 windows and step sizes without any filtering, achieved PER of 16.28% and 10.12% using CRDNN and Wav2VecTransducer models, respectively. In Figure 5.7 (a) and (b), this baseline is shown as dashed lines.

Word-based Speech Inference: We also compare the Word Error Rates (WER) of the prior audio filtering techniques using fine-tuned and pre-trained Whisper models to assess the efficacy of the word-based speech inference models. Figure 5.7(b) shows the WER of fine-tuned whisper for prior filter approaches. CoughSense, Synthetic Sensors, PrivacyMic, and SAMoSA, showed



(a) Phoneme Error Rate (PER) (b) Word Error Rate (WER) (c) Classification Accuracy

Figure 5.8: Results of (a) phoneme-based speech inference, (b) word-based speech inference, and (c) activity classification accuracy on Kirigami filtering approaches.

WER of 29.31%, 32.10%, 48.21%, and 68.11%, respectively, on pure speech sounds showing that fine-tuned whisper models can recognize speech content even after the data is filtered by the prior approaches. In addition, we also see that using an off-the-shelf pre-trained Whisper model has higher WER, which means the prior filter approaches are still resilient to the pre-trained Whisper model. The weakest filter among the four is CoughSense [133] as it has the lower WER scores (29.31%), meaning the inference obtained from the fine-tuned model provides enough information about the original speech content. As a point of comparison, our baseline approaches using FFT data with 256/128 windows and step sizes without any filtering yielded WER of 3.01% and 6.78% with fine-tuned and pre-trained Whisper models, respectively. Overall, these results demonstrate the potential of fine-tuning the ASR model to effectively infer speech content from filtered audio, highlighting its capability to overcome the prior speech filter approaches and provide accurate word-based speech recognition.

Utility Impact: Figure 5.7(c) shows the audio classification accuracy on 10 activity classes on the ESC50 dataset [177] over the four prior approaches. The baseline configuration (no filter) achieved 95.25% and 86.25%. Out of the four prior approaches, the best performer is SAMoSA [155], which achieved 79.50% accuracy in classifying pure activity sounds. The classification performance for the other three approaches is significantly lower than the baseline configuration. Notably, all four approaches showed significant performance drops on overlaid sounds. We hypothesized that this performance drop might be caused by the always-on manner of these filtering approaches, which degrades the expressivity of the audio data and makes activity and speech sound less distinguishable when overlayed. The drop is especially pronounced for SAMOSA (a drop of 46.25%). We posit this drop to SAMOSA's very low default sampling rate (1 kHz). This approach works well when the signal is clean, but the performance plummets significantly when the speech and ambient sounds are overlaid.

5.5.3 RQ2 : Performance of Kirigami filters

Fig. 5.8 summarizes the performance of the Kirigami filter with different configurations of threshold values. Overall, Kirigami filters showed superior protection for speech privacy compared to the prior approaches, especially for configurations that lean towards privacy, such as LR0.1 and LR0.2.

Phoneme-based Speech Inference: As shown in Fig 5.8 (a), the CRDNN model produces almost complete noise, with PER values above 100%, for any of the 5 Kirigami filter configurations. For the Wav2VecTransducer model, as the threshold value moves from 0.5 to 0.1, the difficulty of inferring phonemes, as measured by the PER produced by the Wav2VecTransducer model, increased as expected.

Word-based Speech Inference: We also see similar trends in word-based models (Fig. 5.8 (b)). When the Kirigami filters are applied to the Whisper pre-trained models, we see that they have a higher WER score of more than 90%. For fine-tuned models, LR0.3, LR0.2, and LR0.1 all achieved above 80% WER. In addition, we see that as we change the threshold configuration of Kirigami from being privacy-preserving (threshold = 0.1) to providing higher utility (threshold = 0.5), the WER values decrease from 89.48% to 68.72% for fine-tuned whisper model. Even the lowest WER 68.72%, which is produced by LR0.5 that leans more towards utility, is already higher than the WER for SAMoSA [155], the best-performing privacy filtering technique out of the four prior filters.

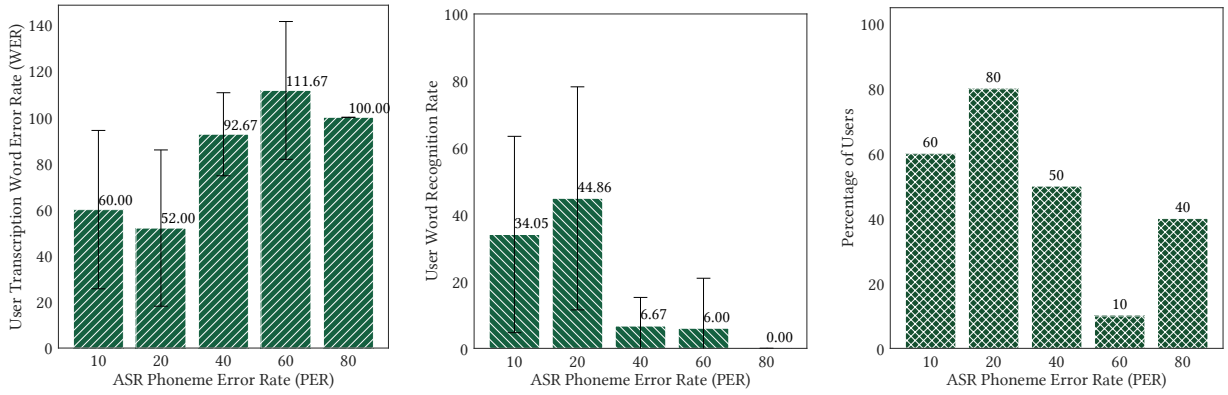
Utility Impact: Out of the five configurations, LR0.5 achieved the best classification accuracy at 82.00% for pure activity sounds and 70.50% for overlaid sounds, which also outperforms all 4 prior filtering approaches that we evaluated. As the Kirigami filter is configured to be more privacy-sensitive, the classification accuracy slightly drops. Even at LR0.2, the accuracy for both pure and overlaid sounds is still above 60%. Another notable difference from prior approaches is that for all Kirigami filters, the negative impacts from overlaid sound are very moderate, at most 11.50% for LR0.5. This advantage of Kirigami, as we hypothesized, is because Kirigami keeps the complete FFT values at the pauses of speech in the overlaid sound, which provides adequate information for the activity recognition. This highlights another advantage of Kirigami filters as to not only protect speech privacy but also maintain utility value even when activities are performed when speech is present.

5.5.4 PER v.s WER Contextualization

While Phoneme Error Rate (PER) and Word Error Rate (WER) are widely used in speech recognition literature, it is difficult to contextualize their privacy implications. For instance, one could ask at what level of PER or WER is an audio featurization technique safe or risky. In addition, it remains a question of what information can be inferred at different PER and WER values. To understand the practical implications of speech inference, we conducted a IRB-approved user study to contextualize how much information can users decipher from the inferred phonemes and words.

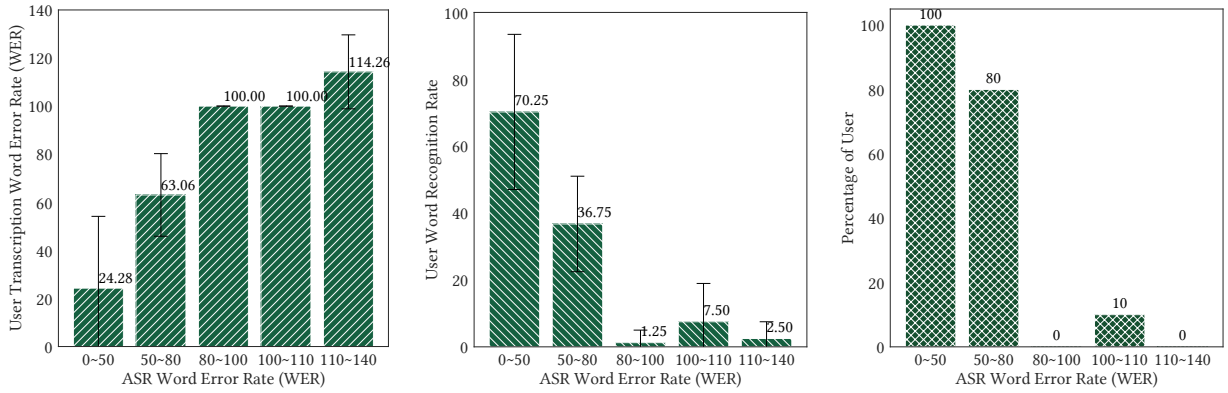
Questionnaire Design: We randomly selected ten sentences from the TIMIT dataset [80] that independently convey a complete meaning. For instance, the sentence *pizzerias are convenient for quick lunch* conveys a statement about pizzerias and lunch. Each sentence was subjected to speech inference predictions through various privacy filters, and the PER and WER were measured for each. For each one of the ten selected sentences, we randomly picked five different predictions that fall into five ranges of PER or WER values. Using these sentences, we created a pool of 50 scenarios, half of which are phoneme-based speech inferences, and the other half are from word-based models. In each scenario, we ask the participants five questions, including transcribing the sentence, identifying words from the original audio, choosing the most likely speech topic, choosing the most likely speech content, and rating the similarity of the prediction to the original sentence. In phoneme scenarios, we presented the segmented phonemes (e.g., *p-iy-ch-er-r-iy-z ih-k-n-v-iy-n y-ih f-aa-r-ah k-w-ih-l-aa ch*), spelling prediction (e.g., *peceruries enchant ye fara Quilla ch*), as well as a reference phoneme pronunciation table. For word model predictions, we only show the sentence prediction (e.g., *combine play them grams a large bowl*).

Study Procedure: In total, we recruited 10 participants (seven females and three males) from the university with an average age of 24.7, ranging between 22 and 28 years. Out of the ten participants, two participants self-identified themselves as having linguistic backgrounds. Before the study began, we introduced participants to phonemes and speech inferences. Then, we went over two example scenarios, one from a phoneme model and the other from a word model. We guided participants through the process of answering five questions for the phoneme scenario and five questions for the word scenario. We demonstrated how to transcribe the words and infer speech content based on the phonemes and word predictions that contain errors, as well as addressed any confusion that participants may have had regarding our study. Once the study began, each participant was randomly assigned ten scenarios, one for each sentence. The ten scenarios that a participant answers all have distinct PER and WER ranges.



(a) User Transcription WER (b) User Word Recognition Rate (c) Topic Inferred User Percentage

Figure 5.9: Results of user transcribing sentence (a), recognizing words (b), and inferring speech topic (c) at various PER from ASR models.



(a) User Transcription WER (b) User Word Recognition Rate (c) Topic Inferred User Percentage

Figure 5.10: Results of user transcribing sentence (a), recognizing words (b), and inferring speech topic (c) at various WER from ASR models.

Study Results: We summarized the results of the user study in Fig. 5.9 and 5.10. Figure 5.9 suggests a steady increase in the difficulty of recognizing words from phonemes as PER increases. Participants, on average, recognize around 40% of the words when phoneme prediction PER is around $10 \sim 20\%$. The majority of participants ($50 \sim 80\%$) are able to infer the topic of the sentence from its phoneme prediction until 60% PER, especially when the user has some prior knowledge of the context or has relevant options shown to them. Although on 80% PER, 40% of participants selected the correct topic, we believe the participants answered the topics correctly by chance after we manually examined the questions and phoneme predictions that these participants received. Therefore, we consider 60% as a suggested threshold of PER, above which minimal information can be obtained from the model inference. For WER, the difficulty in transcribing and recogizing words increases as WER increases. Figure 5.10 shows sharp turning point at $80\sim100\%$, after which it becomes very challenging for participants to recognize any words or topics. Therefore, we suggest an 80% WER threshold as the point at which the model inference can provide only limited information. However, it is recommended to consider both PER and WER together for better privacy assurance. This study provides useful insights into the performance evaluation of ASR systems and can guide future research in this field.

5.5.5 Comparison of Kirigami and Prior Speech Filtering Approaches

Figures 5.11 (a) and (b) show the comparison of the privacy and utility tradeoffs of prior and Kirigami speech-filtering approaches based on two benchmark datasets, pure Timit speech data for speech inference (to assess privacy risks), ESC50 activity recognition dataset (to assess utility benefits for activity recognition) and Timit speech overlaid on ESC50 dataset (to assess utility benefits for activity recognition in noisy data) for phoneme and word based ASR models. Using the scatter plot we can assess the effectiveness of the prior approaches and the Kirigami filters in preserving privacy while preserving the utility for activity recognition. In the scatter plot, the x-axis represents the level of privacy achieved by each approach, with greater distances from

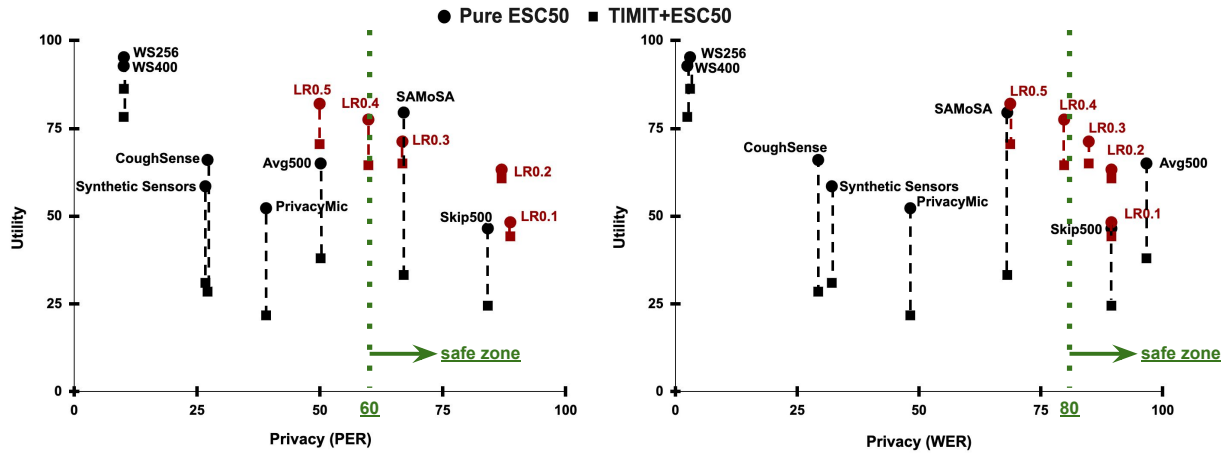


Figure 5.11: Scatter Plot of Privacy and Utility Trade-Off of Different Audio Featurization Techniques. The vertical dashed lines represent the drop in accuracy for the utility measure in the presence of simultaneous speech and ambient sounds (overlaid sounds). Informed by our user study, we consider regions more than 60% PER and 80%WER as safe zones as little information from speech can be inferred. The ideal region is the top-right corner as it maximizes error in reproducing the spoken content and accuracy for the end goal task. For both plots, several Kirigami configurations are near that corner. In (a), SAMoSA [155] is close to the corner too, but the drop off in utility due to the presence of overlaid sounds is substantial. In comparison, Kirigami filters are more immune to noisy environments.

the base indicating higher privacy protection. The y-axis represents the level of utility achieved, indicating the effectiveness or performance of the activity recognition tasks. The privacy metrics PER and WER values for the filtering approaches, picking the lowest PER values and lowest WER values (most privacy-invasive) among the ASR models. Based on user study results, a threshold of 60% for PER and 80% for WER is established as the point at which the filtering approach is deemed safe. The utility metric is picked based on this high accuracy achieved after the filtering technique using the ESC50 dataset. Ideally, the desired positioning of the filtering approaches on the scatter plot is in the top right quadrant. This indicates achieving the highest utility while simultaneously providing the highest privacy guarantees.

Figure 5.11(a) shows the most privacy-preserving filter for phoneme-based ASR models is the Kirigami's LR 0.2, while the least privacy-preserving filter apart from the baseline (WS256, WS400) is CoughSense [133] and Synthetic sensors [129]. While other approaches, such as SAMoSA [155], have high utility accuracy(76%) for pure ESC50 dataset, and their approach is in the PER safe zone, their utility accuracy (26%) drops for activity recognition when noise data is present (Timit speech overlay on ESC50). Based on this, the most ideal approach is Kirigami's LR 0.2 primarily due to higher PER numbers and better utility accuracy for both pure ESC 50 and Timit speech overlay on ESC50 datasets. Figure 5.11(b) shows the most privacy-preserving filter word-based ASR models have both Kirigami's LR 0.2 and Kirigami's LR 0.1, least privacy-preserving is CoughSense [133]. We also see that most of the prior speech-filtering approaches are in the unsafe zone, including SAMoSA and PrivacyMic, indicating that these approaches are

ineffective in preserving privacy. Considering both privacy and utility aspects, Kirigami's LR 0.2 filter emerges as the most suitable choice due to its higher PER numbers and better utility accuracy for both the pure ESC50 and Timit speech overlay on ESC50 datasets. It strikes a balance between privacy preservation and utility enhancement. Our Kirigami's LR 0.2 filter offers a compelling solution, providing a high level of privacy preservation while maintaining satisfactory utility accuracy. However, as we delve into the extensive real-world study, the story takes an unexpected turn. Contrary to our initial expectations, the effectiveness of the Kirigami's LR 0.2 filter, while effective in controlled environments, was significantly impacted by the presence of real dynamic background noise, diverse acoustic landscapes, and fluctuations in ambient noise levels. To overcome this, we present an adaptive background masking process combined with Kirigami's speech filter as mentioned in Section 5.4.4 and evaluate Kirigami's effectiveness in discarding speech.

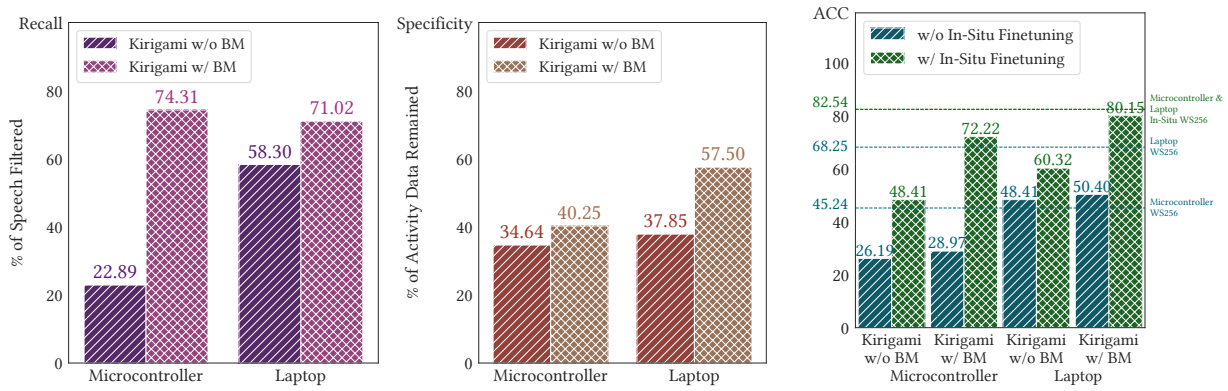
5.5.6 RQ3: Evaluation of Kirigami filter in the Real World

We conducted a user study to evaluate the robustness of our Kirigami filters for speech recognition in real-world environments beyond using audio datasets. In addition, we evaluate speech recognition accuracy in different locations with varying background noises and characterize the Kirigami filter's performance.

Scenarios Definition: We define three scenarios to characterize distinct speech and activity patterns in real-world settings. In scenario 1, to showcase the Kirigami's robustness to the duration of speech, participants are given a script containing randomly selected short and longer-duration sentences from the TIMIT dataset [80], each independently conveying complete meanings. They are asked to speak three short sentences, averaging 15 to 20 seconds each, and three longer sentences, taking approximately 1 minute, repeating each sentence three times. In scenario 2, participants are tasked with speaking short sentences at varying distances (1, 2, and 3 feet) from the source microphone. In Scenario 3, evaluating Kirigami's utility preservation, participants engage in diverse activities, including sporadic actions like clapping or typing, continuous actions like vacuum running, and human voice-based activities such as coughing or laughing. Each of these scenarios is repeated in three locations: a lab, a makerspace, and a conference room. We chose these locations to include a diverse range of background noise profiles.

Study Procedure: For this study, we recruited 7 participants (4 Females, 3 Males) ranging from 22 to 28 years old (Average = 24.7 years). We capture audio data from two input sources:(1) raw audio data from the Laptop (Macbook) microphone and (2) featurized FFT data from a microphone connected to a microcontroller (ARM Cortex-M4F with 256 KB RAM and 1 MB flash). Each participant is provided with a script and a set of sentences. To emulate a real-world setting, we only put constraints on the entire scenario's speech start and end time. However, participants can speak the sentences in any manner they see fit. For each scenario, we evaluate Kirigami's LR filter with and without a background mask, denoted as *Kirigami w/ BM* and *Kirigami w/o BM* and calculate speech and activity recognition accuracy.

Study Metrics: To measure the real-world performance of Kirigami filters to detect speech in the



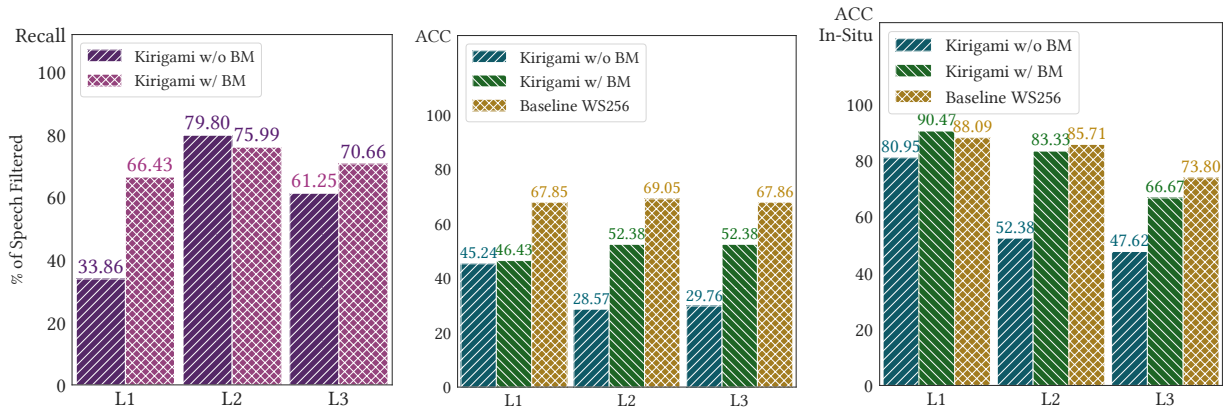
(a) Percentage of filtered speech (b) Percentage of retained activities (c) Activity recognition accuracy

Figure 5.12: Comparing the performance of Kirigami LR filter with and without background mask from Microcontroller and Laptop. The figure shows the percentage of (a) speech-filtered and (b) activity data retained and overall activity recognition accuracy.

real world accurately, we use recall – percentage of speech removed when speech happens and specificity – percentage data available for utility-based models. We selected these two metrics due to their ability to evaluate filter performance independent of the composition ratio between speech and non-speech durations in real-world scenarios. Depending on the application, the ratio of speech and non-speech data can be different from the ratio in our user study or dataset. For this reason, using accuracy as the metric in our case would be strongly affected by the composition ratio of speech in our user study and might not be an informative indicator of real-world performance. To measure the activity recognition performance, similar to before, we use an AST-based model and calculate the classification accuracy in the real world. We further fine-tuned the global model by taking partial activity data as training samples from the user study to show an increase in classification accuracy.

Overall User Study Results: Figure 5.12 shows the performance comparison between the Kirigami filter with and without the background mask when the filter runs on a Microcontroller or Laptop. This result was obtained after the study was conducted among diverse participants and conducted in different locations in the building (L1, L2, L3) with varying background noise.

In Figure 5.12 (a), we see that the Kirigami filter with the background mask (BM) consistently outperforms its counterpart without BM, both on micro-controller and laptop platforms. The filtered speech data percentage is notably higher with BM (Micro: 74.31%, Laptop: 71.02%) compared to without BM (Micro: 22.89%, Laptop: 58.30%). Similarly, the presence of BM results in a greater retention of activity data (Micro: 40.25%, Laptop: 57.50%) compared to without BM (Micro: 34.64%, Laptop: 37.85%), as depicted in Figure 5.12 (b). This observation suggests that the speech recognition performance of the Kirigami LR without BM is significantly impacted by the diverse array of background noises present in real-world scenarios. Furthermore, our analysis indicates that Kirigami with BM achieves higher activity recognition scores, particularly for the laptop (without fine-tuning: 60.32%, with in-situ fine-tuning: 80.15%), surpassing the scores obtained without BM (without fine-tuning: 48.41%, with in-situ fine-tuning: 60.32%). A similar



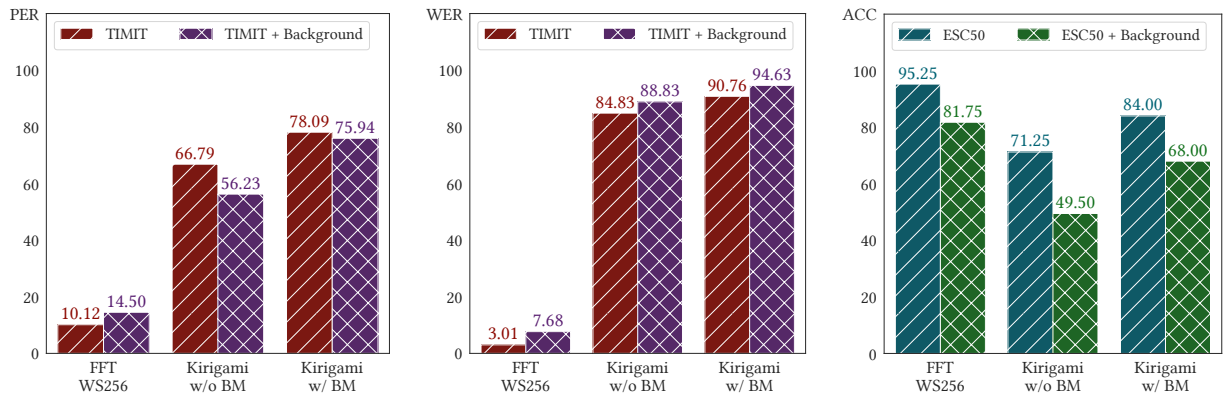
(a) Percentage of speech data filtered (b) Activity recognition accuracy (c) Activity recognition accuracy of model fine-tuned with in-situ data

Figure 5.13: Comparing the performance of Kirigami LR filter on Laptop with and without background mask at various locations: L1-Lab, L2-Makerspace, L3-Conference Room.

trend is observed for the microcontroller, where Kirigami with BM (without fine-tuning: 28.97%, with in-situ fine-tuning: 72.22%) consistently outperforms Kirigami without BM (without fine-tuning: 26.19%, with in-situ fine-tuning: 48.41%). In summary, our findings demonstrate the consistent and robust performance of the Kirigami filter with the background mask in speech filtering across diverse environments. In contrast, the Kirigami filter without the background mask experiences significant performance variations.

Evaluating Kirigami's Performance Across Different Environments: To examine the resiliency of Kirigami LR filter to various background profiles in real-world scenarios, we conducted the user study in three distinct locations characterized by diverse background settings in our campus building. Location L1 represented a laboratory setting, a shared space with multiple individuals, featuring a moderate-level background noise generated by continuous HVAC running and occasional conversations in the surroundings. Location L2, a maker space, exhibited a background profile dominated by low-frequency noise from machinery. In contrast, Location L3, a conference room with an open window, presented an external noise profile, including vehicle honks and birds chirping. Location L3 was the noisiest background environment, while L2 was deemed the least noisiest.

Figure 5.13 shows the comparison of Kirigami filter performance with and without BM in different locations. In general, the Kirigami filter without BM exhibits less accurate speech removal and demonstrates inconsistency across diverse locations. For instance, in location L1, a laboratory environment, the percentage of filtered speech decreases to 33.86% without BM, while the Kirigami filter with BM removes 66.43% of speech data. But in location L2, which is the quieter environment, we see that both Kirigami filter with and w/o filters filter have comparable speech percentage filtered showcasing Kirigami w/o BM performance changes in different locations. However, in comparison, Kirigami w/ BM speech filtering accuracy is consistent in different locations (L1: 66.43%, L2: 75.99%, and L3: 70.66%) while ensuring the utility of the



(a) Phoneme Error Rate (PER) (b) Word Error Rate (WER) (c) Classification Accuracy

Figure 5.14: Results of (a) phoneme-based speech inference, (b) word-based speech inference, and (c) classification accuracy on prior filtering approaches on the clean and noisy dataset.

data preserved is also high across different locations.

Evaluating Speech Inference and Activity Recognition using Kirigami with BM: Speech inference models often experience a decline in recognition accuracy when operating in noisy environments. Similarly, Kirigami filters, operating in a detect-and-remove manner, might also have to face challenges in filtering out speech in noisy environments. Therefore, we conducted empirical tests to confirm that Kirigami w/ BM maintains its robustness under ASR models when exposed to a noisy dataset. This investigation aims to assess the net impact of noisy data on speech inference and activity recognition.

In this evaluation, we use the TIMIT dataset and the same 10 classes ESC50 and various background environmental noises (*TIMIT + Background*) and (*ESC50 + Background*)/ The constructed dataset retains the same structure as the original TIMIT and ESC50 datasets, except various background noises are overlaid. As before, we fine-tuned the CRDNN model, Wav2VecTransducer, and Whisper AI models to infer speech after filtering using Kirigami w/o BM and Kirigami w/ BM. We fine-tuned AST models for activity recognition using the filtered audio. In addition, we include the baseline model using an FFT of 256/128 windows and step sizes, without Kirigami filters, to gauge the impact of background noise on the ASR model alone.

Fig. 5.14 summarizes the performance of the Kirigami filter with and without BM under clean and noisy audio data. We include the summary of evaluation results for space limitation from only the strongest performers: Wav2VecTransducer and Fine-Tuned Whisper AI models. Overall, Kirigami w/ BM showed superior protection for speech privacy and preserved more utility values for activity recognition models than Kirigami w/o BM. As seen in Fig. 5.14 (a) and (b), Kirigami w/ BM remain high PER and WER across clean and noisy datasets, demonstrating its reliable privacy protection under noisy environments, while Kirigami w/o BM suffers from a degradation in PER. As seen in Fig. 5.14 (c) showed that Kirigami w/ BM outperforms Kirigami w/o BM on both the original ESC50 and noisy datasets. The superior performance of Kirigami w/ BM, even for the clean case, is possibly due to its capability to suppress intrinsic background

noise in the original ESC50 dataset.

5.6 Discussion and Limitations

Evaluation or Privacy Filtering Models: Replicating and testing each proposed technique individually is time-consuming and expensive in terms of cloud computing credits needed, especially with model re-tuning or re-training. To address this, we carefully selected at least one prior work representing each type of privacy filtering that we identified. While this selection provides valuable insights into the performance of different filtering approaches, it may not encompass the entire spectrum of privacy filtering techniques available. Future research could explore a broader range of filtering techniques to gain an even more comprehensive understanding of their effectiveness and trade-offs. We believe that Kirigami's edge filtering approach to detect and filter speech-like segments will still remain superior to other approaches in terms of privacy.

Alternative ML Model for Filtering: We focused on using LR as the primary ML model used by our Kirigami filter rather than exploring other alternatives. While our results show that our LR-based Kirigami filter is quite effective, other ML models specifically designed for edge devices, such as TinyML or lightweight recurrent neural network (RNN) models, could offer additional benefits and trade-offs. Our goal was to use a resource-frugal shallow model that could run on a wide range of IoT devices, but we leave the investigation of these alternative ML models as a future exploration.

User Study Validity: We acknowledge the smaller size of our participant pool for the user study as a limitation.A larger and more diverse sample size would further enhance the validity and generalizability of the study results. A larger sample would also provide a broader representation of user preferences, behaviors, and perceptions, leading to more robust conclusions.

Additional Metrics for Speech Privacy: Our study highlights an important consideration regarding the use of Phoneme Error Rate (PER) and Word Error Rate (WER) as metrics for evaluating speech privacy. While PER and WER are commonly used metrics for assessing the performance of automatic speech recognition (ASR) systems, they are not specifically designed for privacy evaluation. Although we measured and reported the "safe zones" based on our user study, indicating areas where privacy is preserved, it is important to note that these safe zones are not guaranteed to be completely safe from privacy risks. Our findings suggest that while PER and WER are useful in determining the privacy characteristics of audio featurization, they should be complemented with additional privacy evaluation measures to provide a more comprehensive assessment of speech privacy. Further research into specialized metrics or evaluation methodologies for speech privacy would contribute to the development of more reliable and robust privacy evaluation frameworks.

5.7 Conclusion

Deep learning-based automatic speech recognition (ASR) has posed new challenges to privacy-focused audio featurization techniques. Such a risk exists primarily because modern ASR systems can be tuned to recognize speech content specifically to these audio featurization techniques. We aim to systematically characterize various featurization techniques on audio data, particularly those that extract statistical and spectral features using Fast Fourier Transforms (FFTs), and evaluate the privacy risks and utility tradeoffs. We first explore different FFT-based featurization approaches proposed in prior works that aim to remove sensitive information from raw audio while providing utility to activity recognition tasks. We then study the recent advancements in deep learning-based automatic speech recognition (ASR) and their potential impact on these edge audio featurization techniques. We also investigate the utility of different featurization approaches in generating discernible features for machine learning prediction. We then propose Kirigami, a general-purpose edge audio speech filter resilient to various speech recognition or audio reconstruction techniques while being feasible to implement on edge devices with limited computational power. We plan to open-source our Kirigami codebase for researchers and practitioners to use and build upon.

Chapter 6

Real-World Applications Enabled by our General-Purpose Sensing System

The Internet of Things (IoT) stands poised to revolutionize our living environments, yet its transformative potential remains largely unrealized due to privacy concerns and practical deployment challenges, as discussed in earlier chapters. To address this, in the preceding chapters, we introduced the concept of general-purpose sensing systems and presented novel approaches to address the challenges of privacy, extensibility, and practical deployability in IoT environments. We discussed the design and implementation of Mites, a scalable general-purpose sensing platform, MLIoT for seamless machine learning integration, TAO for context recognition, and Kirigami for privacy-preserving audio processing. While these technical solutions lay the groundwork for overcoming many of the limitations of traditional purpose-built IoT systems, the ultimate success of general-purpose sensing systems depends on their ability to enable researchers and developers to build a wide variety of compelling IoT applications.

By leveraging the versatile and comprehensive sensing capabilities of the system support features, I demonstrate how a variety of applications can be rapidly prototyped and deployed across different domains, including building management and maintenance, occupancy modeling, and activity recognition. These applications showcase the potential of a general-purpose sensing system to serve various stakeholders within a building environment, such as occupants, building managers, and facility operators. Each application highlights different design primitives of the system, illustrating how the modular and scalable architecture of a general-purpose sensing system can be adapted to meet the specific needs and challenges of diverse use cases.

In the following sections, we present five exemplary applications, each highlighting the power and flexibility of a general-purpose system in a unique context. By doing so, we aim to demonstrate the system's capability to not only address current challenges in smart building environments but also to pave the way for future innovations.

This section presents five exemplary applications in different domains: building management and maintenance (Section 6.1), occupancy modeling (Section 6.2) and activity modeling (Section 6.3). The goal is to demonstrate how the general-purpose system can be used to rapidly prototype new applications. In addition, these applications are geared towards different stakeholders of the building (occupants, building managers, etc.) and leverage different design primitives of our system.

6.1 Management and Maintenance

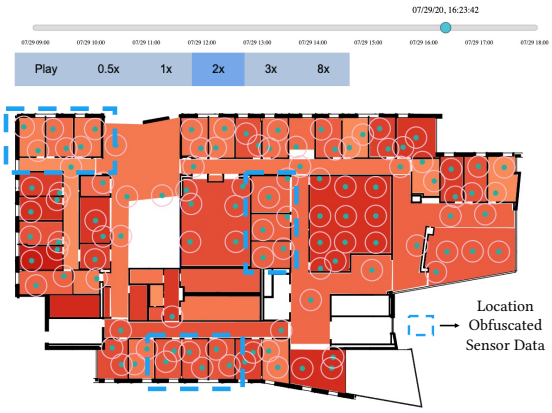
Prior efforts have prototyped applications for building managers such as building environmental modeling [14, 49], fault diagnosis and detection (FDD) [107, 158, 183], and management and maintenance [69, 166]. Such applications can help with obtaining building rating certifications around Well Building standards [227] that assess the sustainability of the building and the health and well-being of the building occupants. These applications benefit from the Mites system as it provides necessary primitives such as rich multi-modal sensing at scale while ensuring the essential privacy features needed for the building occupants (e.g., providing data only at the granularity the applications need). We present two applications we built on top of the Mites system for monitoring the building environment and the health of the sensor deployment.

6.1.1 Spatial Environment Monitoring Application

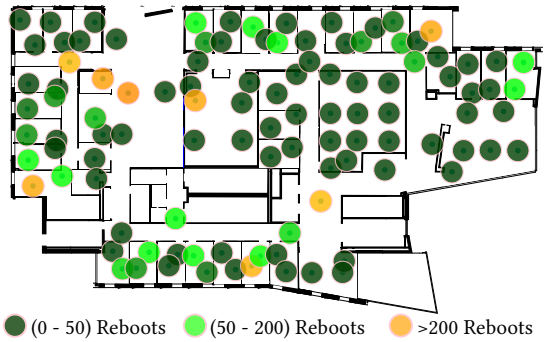
To provide a comprehensive view of the environment within a building (e.g., hot spots in the building), we implemented a spatio-temporal viewer application that stakeholders such as building managers and occupants can use to dynamically visualize current and historical sensor data captured from Mites devices. Figure 6.1(a) illustrates the temperature data from Mites devices in different physical spaces of the building as a heatmap (red=warm, yellow=less warm, blue=colder). We see that certain rooms and corridors in the building are warmer than others, indicating higher setpoints or potentially higher occupancy. Importantly, to ensure the privacy of the occupants of location-obfuscated rooms, the Mites system ensures that sensor data from these locations are random and grouped together into a set of rooms (Section 2.3.3). Building Managers can use this application to detect HVAC system faults and diagnose whether they are localized or systemic. In addition, when other environmental sensor data is overlayed on the same floorplan, it can be used for environmental modeling applications [49]. Building managers can also see how daylight varies across spaces (using the color and the illumination sensor) or isolate noise issues when occupants complain about adding additional insulation to walls.

6.1.2 Device Health Monitoring Application

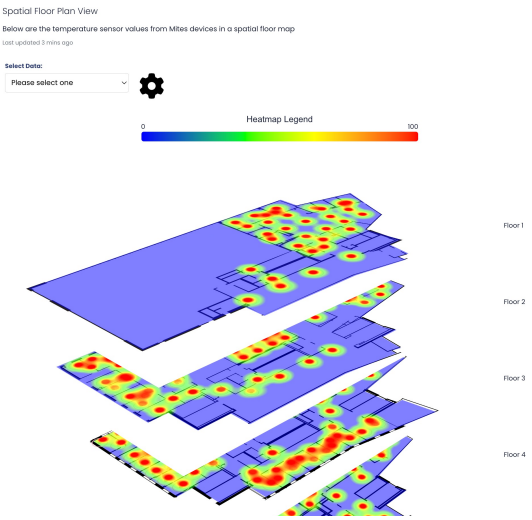
The Mites system gathers numerous health metrics from each Mites device, e.g., the status (online/offline), the number of reboots, and signal strength. While building managers can merely view these data as a time series plot, it is often challenging to diagnose faults and find the primary cause without contexts such as the physical location or how devices in close proximity are performing (i.e., "is WiFi poor in a particular area?"). Figure 6.1(b) shows our spatio-temporal interface showing the reboots experienced by the Mites devices on one floor of our building. Individual circles show the location of the Mites device, and the color gradients denote the number of reboots (dark green indicates low reboots per day, while light green and yellow denote higher reboots). Our interface allows the selection of the desired time range (e.g., January 1st to 31st, 2022) and the replay of the data during that period. This interface has been an invaluable debugging tool during our deployment of 314 Mites devices on our building testbed to find transient and/or persistent issues. Our campus network team worked closely with us throughout the process and has repeatedly mentioned that the Mites deployment in the building and this



(a) Spatio-Temporal Sensor Data Viewer



(b) Spatial Device Health Viewer



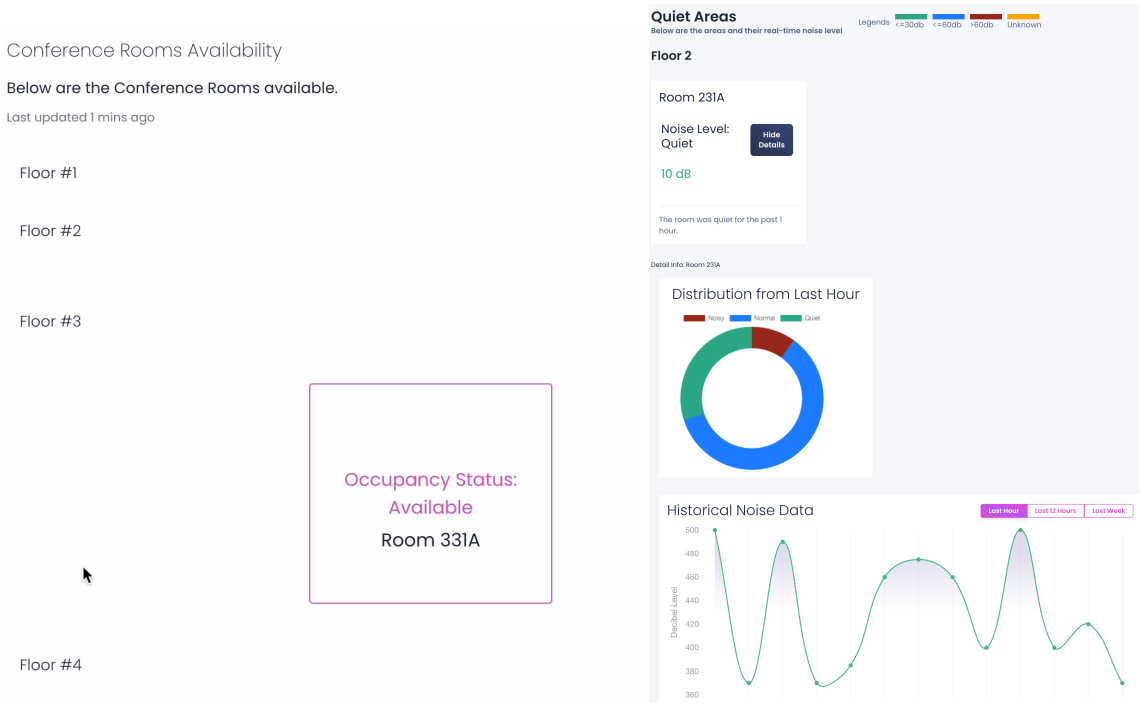
(c) Spatial-Temporal Viewer Across Floors

Figure 6.1: Management and Maintenance applications built on the Mites system. Figures (a) and (b) showcase the spatio-temporal view of the temperature data of each office on a specific floor, captured from the Mites. The color-scaled temperature values that indicate the variation of warmth in rooms are less accurate for location-obfuscated data. (b) The location of individual Mites devices anchored on the floor plan. The color for each circle is filled with a corresponding scaled RGB value based on the number of reboots of that device – an important metric to assess the devices' health spatially.

specific interface were incredibly useful to them in finding and fixing problems with the WiFi network. Specifically, the Mites devices that were having trouble connecting to WiFi, and their locations, formed a distributed WiFi client 'observatory' of sorts and helped identify WiFi dead spots, incorrect WiFi AP TX power configurations, and an intermittent issue with authentication credentials not being cached. In particular, this was the case, even though we have an enterprise-grade WiFi infrastructure.

6.2 Occupancy Modeling

Applications in the domain of occupancy modeling, such as occupancy detection [83, 203], occupant identification [74, 168], and occupancy-based control [9, 24, 30, 116] benefit from sensor deployment such as Mites. These applications primarily model the occupancy of a space or enable occupancy-based equipment control (e.g., HVAC). Such applications are valuable for building managers for energy-efficient building operations and for occupants who want comfortable workspaces. Supporting such applications requires several features of the Mites system such as rich sensing, privacy support, scale, and ML. We present two such applications that we have built for the availability of conference rooms and quiet spaces to work in the building.



(a) Conference Room Availability App (b) Find a Quiet Space App

Figure 6.2: Occupancy modeling applications built on top of Mites platform. Figure (a) showcases an occupancy application that detects the availability of conference rooms using PIR sensor data. Figure (b) shows an application that identifies quiet spaces in the building using acoustic features from multiple Mites devices (wall and ceiling) in the same location.

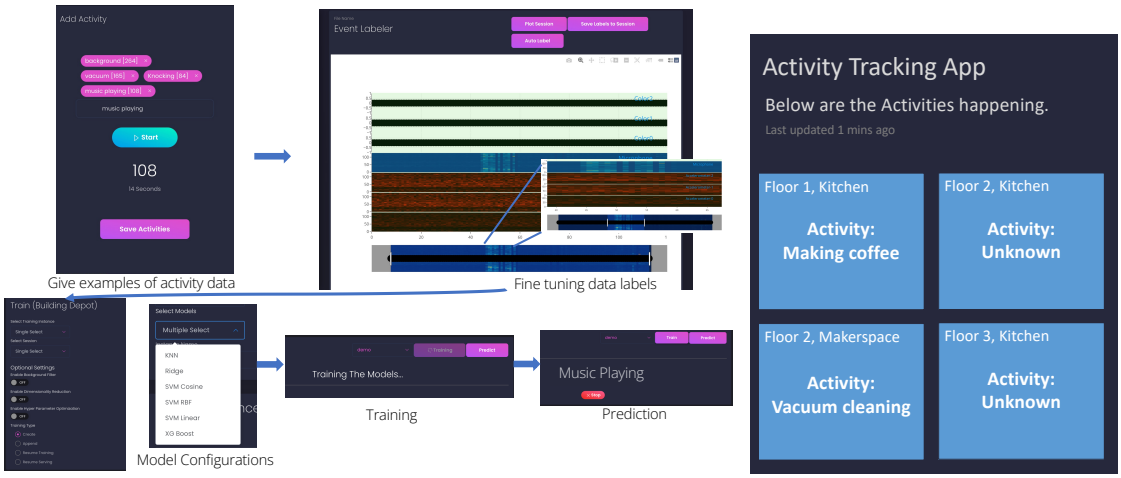
6.2.1 Availability of Conference Rooms

Building managers and occupants often struggle to find meeting rooms and other common spaces. For example, conference rooms that are previously booked for meetings remain unused if the meetings are canceled or finished earlier. To address this, we built an app on top of the Mites system to identify rooms in the building as shown in Figure 6.2(a). This application connects to the Mites platform, identifies public resources such as conference rooms, accesses the sensor

data (PIR) from the Mites devices located on the ceiling, and predicts the occupancy (green-available, red-occupied) of the space. The application also uses the ML features (Section 3) in our system to train an ML-based binary classifier model to distinguish between occupancy and non-occupancy. Building managers can use this coarse usage data and correlate it with environmental parameters like the temperature, humidity, daylight, glare, and noise from surrounding spaces to determine and ultimately fix issues that lead to some conference rooms being used less over others.

6.2.2 Find a Quiet Space

Similar to the previous application, identifying quiet or noisy locations in the building is crucial. Sound within an enclosed space from HVAC equipment, appliances, and other people, have shown to hinder productivity, focus, and memory retention in students and office employees [227]. We built a *Find Quiet space in the building* app that utilizes sensor data (microphone) from multiple Mites devices in an office/shared space (ceiling and wall) to accurately obtain the noise level in the space. Specifically, for larger rooms or halls, combining audio data from different locations is important to (a) accurately predict noise levels and (b) localize the area with higher noise levels in the room. The application leverages the Mites system's scalable stream processing capability (Section 2) to obtain real-time sensor data from multiple devices in the room. The application then converts the obtained FFT features from the audio sensors to individual decibel values [68]. We then use the WELL building standards [228] to calculate the Sound Pressure Level (SPL) and compare it with the threshold provided to identify quiet spaces over noisy ones.



(a) Tools for Machine Learning (b) Activity Tracking App

Figure 6.3: Activity modeling applications built on top of Mites and MLIoT platforms. Figure (b) shows the end-to-end ML toolchain illustrating data collection, annotation, model selection, training and inference. Figure (b) shows an activity recognition application that can detect activities in a kitchenette such as *making coffee* and *heating food* utilizing several sensor channels.

6.3 Human Activity Modeling Applications

Applications in the domain of human activity modeling will also benefit from the Mites system [55, 127, 128]. The occupants of the building can use this information to understand their activities in their personal spaces and assess their wellness (e.g., productivity or stress). These applications are geared towards detecting activities and their patterns and require several capabilities offered by Mites, namely, rich sensing modalities, data annotation tools for in-situ training of activity labels, scalable ML, and support for privacy controls. We built an activity recognition application to identify common activities performed in public locations, such as kitchenettes and our maker space. Figure 6.3(a) illustrates the scalable ML tools to annotate activities of interest, train an ensemble of models, and deploy them for prediction. It begins with the data collection phase, where raw multimodal sensor data is gathered from various sources, such as environmental sensors and wearable devices. This is followed by the annotation process, where the raw data is labeled with meaningful tags, providing context for supervised learning tasks. Model selection comes next, where appropriate ML models are chosen based on the application requirements, sensor data, and computational constraints. Finally, the training phase fine-tunes the models using the annotated data to generate accurate inferences, transforming the raw data into actionable insights while supporting scalable and adaptive IoT deployments. This holistic process ensures seamless integration of machine learning with general-purpose sensing systems.

Figure 6.3(b) shows the activity recognition application for kitchenettes predicting activities such as *making coffee, heating food, using the sink, or toasting bread* using the sensor data from the Mites devices located in the space (particularly the accelerometer and microphone). Notably, these applications demonstrate a complex activity recognition running simultaneously over multiple public spaces, predicting activities it was trained to recognize, which would not be possible without the features outlined in this thesis.

6.4 Extensible Design for Rapid Prototyping Applications

The Mites system exposes a set of REST APIs and PubSub interfaces to its backend to access the sensor data from the Mites devices or the metrics data that allow the developers to prototype IoT applications rapidly. Our REST APIs support individual and batch queries of time-series data for configurable time ranges. To support real-time low latency streaming of both sensor data and inferred events, we incorporated a real-time PubSub broker service (RabbitMQ) that uses the Advanced Message Queuing Protocol (AMQP) protocol. This service enables event-based asynchronous communication to deliver real-time data to different subscribers (applications) at scale, allowing many-to-many communication patterns. Notably, both our REST API and the PubSub interface use the same underlying permission mechanisms for enforcing access control to specific sensors from each Mites device for security and privacy. Using these interfaces, new applications and services can be easily integrated with the underlying Mites system, with less than 40 lines of code.

Chapter 7

Conclusions and Future Directions

This thesis has presented a comprehensive approach to addressing the challenges of privacy, extensibility, and practical deployability in IoT environments through the development of general-purpose sensing systems. My work spans five key areas, each contributing to the overarching goal of creating safe, secure, and user-friendly IoT infrastructures.

First, I introduced Mites, a high-fidelity, general-purpose sensing platform that incorporates onboard featurization to balance data privacy with utility. Through architectural optimizations and rigorous evaluation, I demonstrated Mites' ability to support diverse applications while maintaining high data delivery rates in real-world deployments.

Second, I developed MLIoT, an end-to-end machine learning system that supports the entire lifecycle of IoT applications, from training and efficient service to re-training based on user feedback. MLIoT integrates multiple distributed components and optimization techniques, making our system adaptive, dynamic, and well-suited to handle the diversity of IoT use cases. In addition, this system addresses the critical need for privacy controls throughout the ML model lifecycle, providing users with fine-grained control over model training, serving, and deployment. MLIoT's flexibility and scalability make it well-suited for the diverse requirements of IoT applications.

Third, I presented TAO, which introduces a context-sensing framework designed to elevate the abstraction of ML inferences, enhancing their utility in diverse applications. Using OWL-based ontologies, the TAO system models the different activity patterns as sequential, parallel, or interleaved activities as context information. The temporal pipeline uses an unsupervised clustering algorithm to detect context from new activity patterns and automatically extends our ontology based on new activity patterns.

Fourth, I proposed Kirigami, a general-purpose edge audio speech filter that balances privacy protection with utility for activity recognition tasks. Through systematic characterization of various featurization techniques, I developed a framework for evaluating the trade-offs between privacy and utility in audio data processing.

Finally, I conducted a longitudinal user study to examine the real-world impact and user adoption of our general-purpose sensing systems. This study provided crucial insights into user behavior, privacy preferences, and data-sharing patterns over time. By investigating the relevance of existing privacy primitives, the impact of perceived application utility on privacy choices, and the nature of data sharing between user groups, I validated the effectiveness of our technical

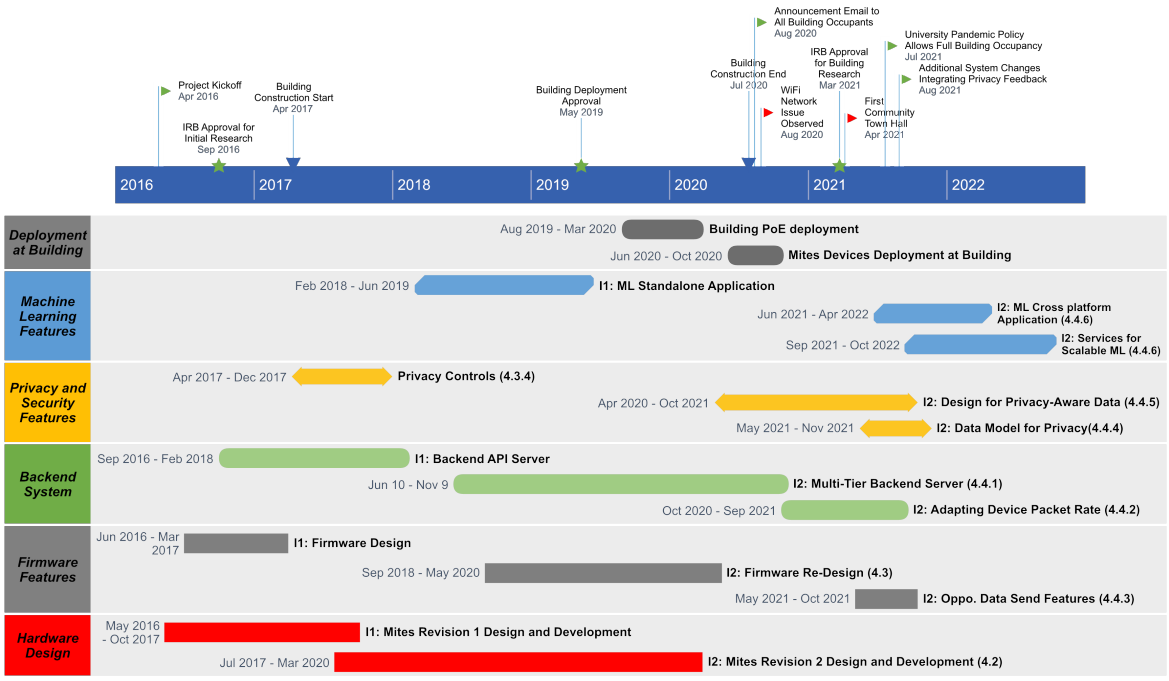


Figure 7.1: Overall timeline of the different projects outlined in this thesis. I highlight key events at the top and the features we worked on for different parts of our stack. The timeline shows the iterative design process, including multiple revisions to the Mites device, backend, and integration with services such as ML.

solutions and identified areas for further improvement.

Collectively, these contributions represent a significant step forward in realizing the full potential of IoT in transforming our living environments. By addressing key challenges in privacy, extensibility, and practical deployment, my work lays the foundation for the widespread adoption of general-purpose sensing systems. The longitudinal study, in particular, bridges the gap between theoretical capabilities and practical acceptance, providing a holistic view of how our proposed solutions perform in real-world settings. My work has been deployed to the real world at TCS Hall, a five-floor mixed-use office building on the Carnegie Mellon University campus used today by more than 200+ occupants. I sincerely hope my work will impact future researchers and practitioners attempting to design and deploy a similar general-purpose sensing system for buildings.

7.1 Lessons Learnt

We have been working towards our vision of a general-purpose ubiquitous sensing infrastructure to enable smart building applications for more than five years. Figure 7.1 illustrates a timeline view of various key events in our iterative process of design, implementation, deployment, and refinement with real-world stakeholders, including building architects, the facility design group, contractors, and building occupants in the loop. While IoT devices are being deployed world-

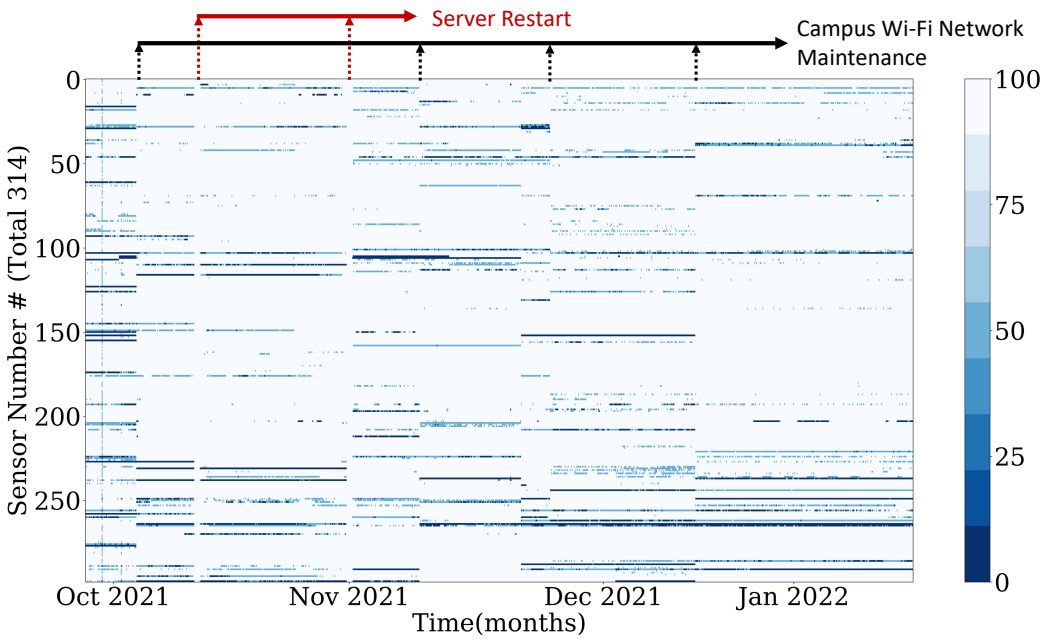


Figure 7.2: Overall packet rate performance (0 - 100%) of the each Mites device over a four-month period. From Oct 2021 - Jan 2022, we see the trend that the packet rate of devices changes from worse (dark blue) to better (light blue), while some of them continue to remain worse due to WiFi receptivity issues after several upgrades to the campus WiFi network maintenance (e.g., software updates to APs) over time.

wide, including buildings, requirements such as security, user privacy, maintenance, and ML integration are often not first-class design constraints (Section 4.2). We believe that our insights into the design, development, and deployment of our building-scale sensing infrastructure based on our stated set of goals will be useful to researchers and practitioners doing similar deployments in the future, helping them achieve systems that are likely to succeed in the real world. We have organized our takeaways into five categories:

7.1.1 Iterative Hardware Design: Don't Reinvent the Wheel, but You May Have to Build Your Own

Before embarking on building our own multi-modal sensor package, we attempted to leverage existing designs such as the TI Sensor tag or Rasberry PIs with sensor "HATs". Ultimately, they were all inadequate for several reasons, such as lack of sensing capabilities, inextensible design, cost, poor programming support, lack of flexibility, etc. Ultimately, we decided to build our own prototype and leverage the sensors chosen by other researchers. Therefore, our first prototype was a 'Lo-Fi' breadboard version with ten different sensors (Figure 2.2(a)). After multiple iterations, we finally came up with a custom PCB design optimized for size, cost, and layout (Figure 2.2(b-c)). Notably, testing these prototypes in the real world was crucial. However, our first design worked (as shown in figure 7.3 and figure 7.1 as I1: Mites Rev1 Design), we decided to eliminate some sensors (e.g., non-contact infrared temperature sensor) since they were superseded by others (e.g., PIR - Passive Infrared) or were capturing similar information at higher

costs (e.g., a geophone is expensive compared to a MEMS IMU) [128].

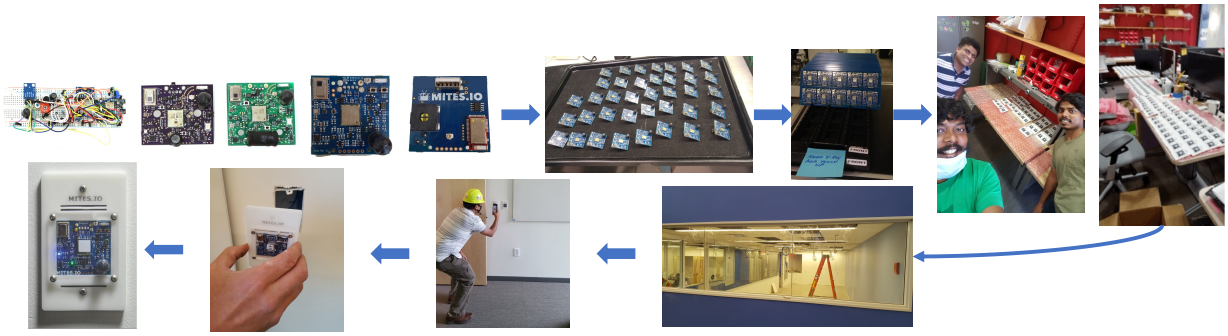


Figure 7.3: The set of images showcases the development journey of the Mites hardware prototype, from initial design to full-scale deployment. The first five images highlight early-stage prototype designs. The next two images transition to the factory setting, where batches of Mites devices are assembled on production lines, showing the systematic process of building the hardware at scale. Next, a hands-on moment is captured where we manually flash firmware onto over 350 devices, ensuring each unit is correctly programmed. The final four images depict the deployment phase, showing the Mites devices installed on the walls and ceilings of a building at Carnegie Mellon University, ready to collect data and power real-world applications.

Moreover, during the initial design of the Mites hardware, after much exploration, we decided to build our Mites device around the Particle's P0 module [171]. It provided us with the base prototyping functionalities in their system firmware (open source DeviceOS) and provided complete flexibility for building and customizing our application firmware. This saved us significant time in the development process but also created a dependency on Particle's P0 hardware module, their DeviceOS, and any limitations (limited memory, older microcontroller, limited support, etc.).

7.1.2 Known Unknowns and Unknown Unknowns for Sensor Deployments

During our initial deployment of devices in the real world, we encountered some unexpected hardware and software problems. For example, we implemented hardware and software watchdog timers (Section 2.3.2) to force devices to reboot when they go into different error states, believing that it would clear all expected faults (e.g., known unknowns). Indeed, these recovery mechanisms worked for a small scale of sensors (10 - 50 sensors) for a few months, but when we deployed all 314 sensors in our building, multiple devices would periodically stop sending data, and the watchdog timers did not help recovery (e.g., unknown unknowns). After several weeks of debugging our hardware (August 2021, as shown in Figure 7.2), our backend, and numerous discussions with the campus networking team, we found that the enterprise WiFi controllers installed in the building randomly moved some of the Mites devices to an unregistered mode, allowing WiFi association but dismissing connection to our backend and hence not triggering the watchdog. This issue was resolved by manually clearing the state of the Aruba WiFi controllers, and a bug was filed with the vendor.

7.1.3 Real-world Conditions are Chaotic and Unpredictable

Each Mites device sends 2 KB packets at a maximum rate of 10 Hz (that is, 20 KB/s) over traditional enterprise 2.4 GHz WiFi networks. During our initial deployment at different locations on campus, we did not identify any problems with packet rate drop or connectivity issues of the devices. Additionally, we assumed that sending data at 20 KB/s from 300+ devices would be easily handled by enterprise WiFi networks. However, as mentioned in Section 2.3.3, achieving reliable packet delivery at 10 Hz was nontrivial even for enterprise WiFi networks (Figure 2.6(a)). The potential reasons for this include limited non-overlapping channels in 2.4 GHz, co-channel interference, and contention between Mites devices sending packets. To overcome this, we developed the adaptive packet rate mechanism based on real-world network conditions (Section 2.3.3). A key lesson based on our experience building-scale, high-fidelity general-purpose sensing requires comprehensive mechanisms to dynamically adapt to existing network conditions.

7.1.4 Deployability is the Key to Reducing Costs

We optimized the design of the Mites system for deployability, which helped us reduce the overall cost of device deployment and labor costs to install them. For example, our Mites devices, running 110 V AC drop, are prohibitively expensive in buildings due to code requirements. Hence, to overcome this, we chose low-voltage Power-over-Ethernet (PoE) and plenum-rated Ethernet cables, which are substantially cheaper to run. In addition, to simplify this deployment process, especially for contractors who deployed the Mites devices, we programmed the Mites devices to have a "SoftAP" mode, which we extended to the application firmware. Therefore, during the deployment, to connect the Mites to the WiFi, the contractor would pull a Mites device from a box, connect it to the POE cable inside the 1-gang work box, and use SoftAP mode to connect it to the designated WiFi network. After the device successfully connected to our backend, denoted by its LEDs, using a simple interface, the contractor marked the device (indicated by its ID) with its physical location. End-to-end, our contractors reported that they took between 15 - 20 minutes per device installation at a labor cost of $100 / hour. This led to an estimated cost of $25 - 35 per device. Note that this did not include the cost of running the Ethernet cables to all locations, adding 1-gang wall boxes, and connecting the cables to network switches in the closets. Overall, to run the additional POE cables to the network equipment closet and single-gang workboxes, for around 350 total PoE Ethernet drops, our contractors charged us around $\$120K$ USD, including the installation of the actual sensors (resulting in approximately $\$340$ per Mites device installation). Without these features, retrofitting the devices in an existing building would lead to higher costs. Our key insight here is that POE drops should be opportunistically added to convenient locations when a new building is being built to future-proof it for making it smart by simply adding Mites-like sensors.

We designed two versions, both of which are powered by a standard USB-A male plug. Furthermore, to support ad-hoc deployments of Mites sensor nodes, especially when 110 V AC wall power is available, we design our device to be powered by a standard USB power adapter.

7.1.5 Community Sense of Privacy is the Key to Building Trust

We designed and implemented numerous privacy mechanisms on the Mites device itself (e.g., edge featurization and denaturing of sensitive sensor data), as well as expressive transparency primitives and controls enabled by our Mites system and application. We built these privacy controls in anticipation of building occupant privacy requests. All TCS Hall occupants were informed of the Mites deployment (August 2020), explaining the sensor boards, our plan to submit a study design to the IRB and begin collecting sensor data, and ways to contact the research team. Notably, this was before occupants moved in since COVID restrictions were still in place. Then, after our IRB protocol was approved (March 2021) and occupants had started to occupy the new building, we held various town halls (the first of which was in April 2021, with the last held in July 2021). In these meetings, we discussed the Mites deployment and answered any questions or concerns. Throughout this community feedback period, we also provided mechanisms for community members to provide anonymous feedback. However, as with any new technology, there was skepticism and pushback from some members of our TCS Hall community. This feedback from our community members and occupants of the building was extremely helpful and led us to redesign some aspects of our data model (Sections 2.3.3), such as reducing the indirect association risk. Importantly, this improvement period happened before any sensor data was collected. We also allowed building occupants to request unplugging the Mites device in addition to the POE-based power-off mechanisms we had designed from the start of the project. We also point readers to our 20-page Mites FAQ document [153], describing our security primitives as well as the various notice and choice mechanisms we implemented for occupant privacy. Although this feedback cycle took time, it was critical for us to incorporate feedback, improve the system, and gain community trust. As of the time of writing, after more than four years of deployment, nine offices (out of approximately 110) have opted out.

7.2 Future Directions

Building on the foundational work of designing high-fidelity, general-purpose sensing platforms, developing end-to-end machine learning systems for IoT, establishing frameworks for generating rich contextual insights, and implementing privacy-preserving audio filters, several promising future research directions can be pursued. These directions aim to address emerging challenges and leverage new opportunities in the IoT landscape.

7.2.1 Improving Support for Privacy

Dynamic and Adaptive Privacy Controls: My thesis demonstrates the importance of providing privacy controls in sensing systems. However, a critical next step is to develop dynamic privacy controls that adjust based on the context of data collection and usage. These controls should recognize different scenarios and apply appropriate privacy measures. For instance, privacy settings in a private home environment should differ significantly from those in a public space like a office building or a gym.

Future research should focus on adaptive privacy mechanisms that learn from user interactions and preferences to provide a tailored privacy experience. These mechanisms could use machine learning to predict and enforce privacy settings dynamically, ensuring that user privacy is maintained without compromising the functionality and utility of the sensing systems.

Privacy in Shared Spaces: Privacy management is a significant concern with sensing systems, especially in shared user spaces. While my thesis addresses some of these concerns by using privacy-by-design techniques to featurize raw data and provide users with control over who gets access to their information, these measures do not fully mitigate all privacy issues a user may encounter. The limited control users have over sensors that capture sensitive information in a shared environment leads to privacy concerns, as these systems often apply the same privacy policy to everyone in the space. Future research should explore methods to enable individualized privacy control for each user. For example, if Alice does not want to be tracked through the building, she should be able to opt out, whereas if Bob wants to be tracked to receive personalized recommendations, he should be able to opt in within the same shared space. This approach would allow for more nuanced and user-specific privacy management, enhancing user trust and acceptance of IoT systems.

Privacy, Utility, and Data Sharing in General-Purpose Sensing Systems: The Internet of Things (IoT) has the potential to transform our living environments, but its widespread adoption remains hindered by privacy concerns and deployment challenges, as outlined in previous chapters. While this thesis introduced novel approaches—such as the Mites platform for scalable, general-purpose sensing, MLIoT for machine learning integration, TAO for context recognition, and Kirigami for privacy-preserving audio processing—these technical solutions alone are not sufficient for driving adoption.

Future work should focus on addressing the real-world user acceptance and long-term sustainability of these systems. This includes conducting longitudinal studies to evaluate how users interact with general-purpose sensing systems in their daily lives, how privacy concerns evolve over time, and how well these systems adapt to changing stakeholder needs. Specifically, a month-long user study could be conducted to understand the interaction between users and general-purpose sensing platforms like Mites. This would provide valuable insights into the practical utility of the system and the user perceptions of privacy controls and system transparency. Additionally, future research should explore how these systems can continue to evolve, improving upon key areas like extensibility, privacy management, and deployment scalability, while fostering a more robust ecosystem of real-world applications. Long-term user feedback and practical deployment studies will be essential in bridging the gap between the theoretical capabilities of general-purpose sensing and their successful integration into everyday environments.

7.2.2 Real-world Activity Recogntion

Improving Data Labeling Techniques For Building Deployable Systems: Real-world activity recognition presents unique challenges, primarily due to the variability in background noise, sensor placement, user behavior, and the difficulty in capturing accurate label data. My the-

sis addresses some aspects of this by using clustering techniques and labeling tools to capture accurate label data; there is more work to do here. Future research should build upon this by exploring advanced clustering algorithms and semi-supervised learning techniques to minimize user involvement further while maintaining high accuracy.

Foundation models for Activity Recognition: The building of foundation models for sensor data holds promise for improving activity recognition. Foundation models, pre-trained on large and diverse datasets collected from real-world TCS deployment, can provide robust feature representations that generalize well to various real-world smart-building scenarios. Future research should focus on adapting these models to sensor data, optimizing them for the specific characteristics and constraints of IoT environments. This approach can significantly enhance the performance and adaptability of activity recognition systems, making them more practical.

Multimodal Machine Learning Approaches to Improve Accuracy: My thesis highlights the integration of various sensing modalities and the need for a dedicated ML platform to handle the resulting data. Future work should focus on developing multimodal machine learning algorithms that can integrate and analyze diverse data sources such as audio, environmental, and motion sensors. These algorithms can provide richer and more accurate insights while ensuring that the privacy of each data type is adequately protected. Future research should also aim to create robust and scalable multimodal models that can adapt to varying data quality and availability. These models should be capable of handling real-time data streams and making inferences that respect user privacy across different contexts and applications.

7.2.3 Synthetic Data Generation

My thesis shows the potential of using data from IoT systems to train machine learning models. As IoT systems generate vast amounts of data, synthetic data generation can play a critical role in augmenting training datasets, especially when real-world data is scarce, sensitive, or expensive to obtain. Future research can focus on developing sophisticated generative models that create high-quality synthetic data while preserving the properties and privacy of the original datasets.

Selective Data Obfuscation for Privacy: A critical area for future research is selective obfuscation techniques within synthetic data generation. These techniques aim to preserve the utility of data for model training while selectively obfuscating sensitive information to protect user privacy. Selective obfuscation involves masking or altering sensitive attributes in synthetic datasets while retaining the statistical characteristics and utility required for effective model training. This approach ensures that synthetic data maintains privacy guarantees, preventing unauthorized disclosure of sensitive information. Research can explore advanced methods, such as differential privacy mechanisms tailored for synthetic data generation, which provide rigorous privacy guarantees without compromising the utility of the data. Furthermore, investigating the application of selective obfuscation across different sensing modalities—such as audio, environmental, and motion data—can expand its utility and effectiveness in diverse IoT applications. Developing scalable algorithms and frameworks for selective obfuscation will be crucial to supporting large-scale deployments of IoT systems while complying with stringent privacy regulations and user

expectations.

7.2.4 Robust Edge Computing

My thesis illustrates the importance of edge computing in reducing latency and improving privacy by processing data locally. To further enhance edge computing capabilities, future research should focus on developing lightweight and efficient algorithms that can run on edge devices with limited computational resources. Another important direction is the development of robust frameworks for edge-cloud collaboration. Such frameworks should allow seamless data processing and analytics across edge and cloud environments, optimizing performance while maintaining privacy and security. This approach would enable more efficient and secure IoT systems that can adapt to changing conditions and user requirements.

7.2.5 App Store for General-Purpose Sensing Systems

An innovative avenue for future research involves the development of an app store tailored specifically for general-purpose sensing systems in IoT environments. This app store would serve as a centralized platform where developers can publish, distribute, and manage applications that leverage sensor data for diverse use cases.

Access Control Mechanisms: One crucial aspect to explore is the implementation of robust access control mechanisms within the app store framework. This includes defining policies and protocols that govern how applications access and utilize sensor data. Privacy-enhancing technologies, such as differential privacy and encryption, can be integrated to protect sensitive information while enabling meaningful data utilization.

App-to-App and App-to-User Privacy Controls: Future work could delve into enhancing privacy controls within the app store ecosystem. This involves enabling users to define granular permissions for each application, specifying what data can be accessed and how it can be used. Additionally, mechanisms for managing interactions between different applications (app-to-app) and between applications and users (app-to-user) should be explored to maintain privacy and security across the ecosystem.

Sensor Views and Data Governance: To empower users with greater control over their data, research can focus on developing intuitive sensor views within the app store interface. These views would provide transparent insights into which sensors are active, what data is being collected, and how it is being utilized by installed applications. Advanced data governance frameworks can also be explored to enable users to modify data-sharing settings dynamically based on changing preferences and contexts.

Community and Developer Engagement: Lastly, fostering a vibrant community of developers and stakeholders is essential for the success of the general-purpose sensing ecosystem. Initiatives could include providing comprehensive developer tools, APIs, and documentation to streamline app development and deployment.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA, 2016. USENIX Association. ISBN 9781931971331. 2.2.2, 3, 3.1, 3.1.2, 3.2.1, 3.2.3

[2] Martín Abadi et al. Tensorflow: A system for large-scale machine learning. In *Proc. of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA, 2016. USENIX Association. ISBN 9781931971331. 3.4.1

[3] Alireza Abdoli. Time series data mining algorithms towards scalable and real-time behavior monitoring, 2021. URL `https://arxiv.org/abs/2112.14630`. 4.2

[4] Adafruit. Adafruit feather platform. `https://learn.adafruit.com/adafruit-feather`, 2023. 2, 2.2.1

[5] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. Fit iot-lab: A large scale open experimental iot testbed. In *Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, WF-IOT '15, page 459–464, USA, 2015. IEEE Computer Society. ISBN 9781509003662. doi: 10.1109/WF-IoT.2015.7389098. URL `https://doi.org/10.1109/WF-IoT.2015.7389098`. 2

[6] Yuvraj Agarwal, Thomas Weng, and Rajesh K. Gupta. The energy dashboard: Improving the visibility of energy consumption at a campus-wide scale. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, page 55–60, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605588247. doi: 10.1145/1810279.1810292. URL `https://doi.org/10.1145/1810279.1810292`. 1, 2.1.1

[7] Yuvraj Agarwal, Thomas Weng, and Rajesh K. Gupta. The energy dashboard: Improving the visibility of energy consumption at a campus-wide scale. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, page 55–60, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605588247. doi: 10.1145/1810279.1810292. URL

https://doi.org/10.1145/1810279.1810292. 2.3.1

[8] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, page 1–6, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304580. doi: 10.1145/1878431.1878433. URL https://doi.org/10.1145/1878431.1878433. 1, 2, 3

[9] Yuvraj Agarwal, Bharathan Balaji, Seemanta Dutta, Rajesh K. Gupta, and Thomas Weng. Duty-cycling buildings aggressively: The next frontier in hvac control. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 246–257, New York, NY, USA, April 2011. IEEE. 1, 2, 2.1.1, 2.2.3, 2.3.1, 6.2

[10] Alessandra Agostini, Claudio Bettini, and Daniele Riboni. Hybrid reasoning in the care middleware for context awareness. *International journal of Web engineering and technology*, 5(1):3–23, 2009. 4.2

[11] Jose Aguilar, Marxjhony Jerez, and Taniana Rodríguez. Cameonto: Context awareness meta ontology modeling. *Applied computing and informatics*, 14(2):202–213, 2018. 4.2

[12] Zeeshan Ahmed et al. Machine learning at microsoft with ml. net. In *Proc. of the 25th ACM SIGKDD Internat. Conference on Knowledge Discovery & Data Mining*, pages 2448–2458, New York, NY, USA, 2019. ACM. 3.2.3

[13] Ane Alberdi, Asier Aztiria, and Adrian Basarab. Towards an automatic early stress recognition system for office environments based on multimodal measurements: A review. *Journal of biomedical informatics*, 59:49–75, 2016. 4.2

[14] Akram Syed Ali, Christopher Coté, Mohammad Heidarinejad, and Brent Stephens. Elemental: An open-source wireless hardware and software platform for building energy and indoor environmental monitoring and control. *Sensors*, 19(18):4017, 2019. 2, 2.1.1, 2.3.2, 6.1

[15] Amazon AWS. Amazon Rekognition – Video and Image - AWS. https://aws.amazon.com/rekognition, 2020. 3.2.1, 3.3.1

[16] Uchenna P Daniel Ani, Jeremy M Watson, Benjamin Green, Barnaby Craggs, and Jason RC Nurse. Design considerations for building credible security testbeds: Perspectives from industrial control system use cases. *Journal of Cyber Security Technology*, 5(2): 71–119, 2021. 1, 2

[17] Arduino. Arduino Cloud. https://cloud.arduino.cc/, 2023. 2.2.2

[18] Louis Atallah and Guang-Zhong Yang. The use of pervasive sensing for behaviour profiling—a survey. *Pervasive and mobile computing*, 5(5):447–464, 2009. 4.2

[19] Amazon AWS. IoT Greengrass. https://aws.amazon.com/greengrass/, 2020. 3, 3.1.2, 3.2.1, 3.2.3

[20] Amazon AWS. AWS IoT— Industrial, Consumer, Commercial, Automotive — Amazon Web Services. https://aws.amazon.com/iot/, 2023. 2.2.2

[21] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020. 5, 5.1.3, 5.3.1

[22] Yang Bai, Li Lu, Jerry Cheng, Jian Liu, Yingying Chen, and Jiadi Yu. Acoustic-based sensing and applications: A survey. *Computer Networks*, 181:107447, 2020. 5

[23] Bharathan Balaji, Hidetoshi Teraoka, Rajesh Gupta, and Yuvraj Agarwal. Zonepac: Zonal power estimation and control via hvac metering and occupant feedback. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, page 1–8, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324311. doi: 10.1145/2528282.2528304. URL https://doi.org/10.1145/2528282.2528304. 2.3.1

[24] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. Sentinel: Occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320276. doi: 10.1145/2517351.2517370. URL https://doi.org/10.1145/2517351.2517370. 1, 2.1.1, 2.3.1, 3, 6.2

[25] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, BuildSys '15, page 13–22, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450339810. doi: 10.1145/2821650.2821674. URL https://doi.org/10.1145/2821650.2821674. 2

[26] Bharathan Balaji, Jason Koh, Nadir Weibel, and Yuvraj Agarwal. Genie: A longitudinal study comparing physical and software thermostats in office buildings. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, page 1200–1211, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450344616. doi: 10.1145/2971648.2971719. URL https://doi.org/10.1145/2971648.2971719. 1, 2, 2.3.1

[27] Gustavo EAPA Batista, Xiaoyue Wang, and Eamonn J Keogh. A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 699–710. SIAM, 2011. 4.3.3

[28] Denis Baylor et al. Tfx: A tensorflow-based production-scale machine learning platform. In *Proc. of the 23rd ACM SIGKDD Internat. Conference on KDD*, KDD '17, page 1387–1395, New York, NY, USA, 2017. ACM. ISBN 9781450348874. doi: 10.1145/3097983.3098021. URL https://doi.org/10.1145/3097983.3098021. 3.2.3

[29] Beecham. Why IoT Projects Fail, a Beecham Research report. https://www.idglat.com/afiliacion/whitepapers/2020-5-ar-why-iot-projects-fail-en.pdf, 2023. 1, 2, 2.1.3

[30] Alex Beltran, Varick L. Erickson, and Alberto E. Cerpa. Thermosense: Occupancy ther-

mal based sensing for hvac control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, page 1–8, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324311. doi: 10.1145/2528282.2528301. URL https://doi.org/10.1145/2528282.2528301. 1, 2, 2.2.1, 2.3.1, 6.2

[31] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive and mobile computing*, 6(2):161–180, 2010. 4.2

[32] Sejal Bhalla, Mayank Goel, and Rushil Khurana. Imu2doppler: Cross-modal domain adaptation for doppler-based activity recognition using imu data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(4), dec 2022. doi: 10.1145/3494994. URL https://doi.org/10.1145/3494994. 2.3.2

[33] Arka Bhattacharya. *Enabling scalable smart-building analytics*. University of California, Berkeley, University of California, Berkeley, 2016. 1, 2

[34] Aldo Luiz Bizzocchi. How many phonemes does the english language have? *International Journal on Studies in English Language and Literature (IJSELL)*, 5(10):36–46, 2017. 5.1.2

[35] Jeremy A Blumenthal, Meera Adya, and Jacqueline Mogle. The multiple dimensions of privacy: Testing lay expectations of privacy. *U. Pa. J. Const. L.*, 11:331, 2008. 5.1.1

[36] Sudershan Boovaraghavan, Anurag Maravi, Prahaladha Mallela, and Yuvraj Agarwal. Mliot: An end-to-end machine learning system for the internet-of-things. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, IoTDI '21, page 169–181, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383547. doi: 10.1145/3450268.3453522. URL https://doi.org/10.1145/3450268.3453522. 3, 3.4

[37] Sudershan Boovaraghavan, Chen Chen, Anurag Maravi, Mike Czapik, Yang Zhang, Chris Harrison, and Yuvraj Agarwal. Mites: Design and deployment of a general-purpose sensing infrastructure for buildings. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 7(1), mar 2023. doi: 10.1145/3580865. URL https://doi.org/10.1145/3580865. 2, 2.3.3, 2.4, 3.4.1, 5, 5.1.4

[38] Sudershan Boovaraghavan, Prasoon Patidar, and Yuvraj Agarwal. Tao: Context detection from daily activity patterns using temporal analysis and ontology. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 7(3), sep 2023. doi: 10.1145/3610896. URL https://doi.org/10.1145/3610896. 4.4

[39] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 1994. 5.3.1

[40] Andreas Bulling, Jamie A Ward, Hans Gellersen, and Gerhard Troster. Eye movement analysis for activity recognition using electrooculography. *IEEE transactions on pattern analysis and machine intelligence*, 33(4):741–753, 2010. 3.1.1

[41] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan,

Patrick McDaniel, and A. Selcuk Uluagac. Sensitive information tracking in commodity IoT. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1687–1704, Baltimore, MD, August 2018. USENIX Association. ISBN 978-1-939133-04-5. URL https://www.usenix.org/conference/usenixsecurity18/presentation/celik. 2.1.2

[42] Cgroups. Control Groups — The Linux Kernel. https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v1/cgroups.html, 2020. 3.3.2

[43] Carl Chalmers, Paul Fergus, et al. Detecting activities of daily living and routine behaviours in dementia patients living alone using smart meter load disaggregation. *IEEE Transactions on Emerging Topics in Computing*, 2020. 3

[44] Chen Chen, Ke Sun, and Xinyu Zhang. Captag: Toward printable ubiquitous internet of things: Poster abstract. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, SenSys '20, page 669–670, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375900. doi: 10.1145/3384419.3430410. URL https://doi.org/10.1145/3384419.3430410. 2

[45] Dong Chen, Phuthipong Bovornkeeratiroj, David Irwin, and Prashant Shenoy. Private memoirs of iot devices: Safeguarding user privacy in the iot era. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1327–1336, 2018. doi: 10.1109/ICDCS.2018.00133. 3

[46] Francine Chen, John Adcock, and Shruti Krishnagiri. Audio privacy: reducing speech intelligibility while preserving environmental sounds. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 733–736, 2008. 5.2.1

[47] Harry Chen, Tim Finin, and Anupam Joshi. The soupa ontology for pervasive computing. In *Ontologies for agents: Theory and experiences*, pages 233–258. Springer, 2005. 4.2

[48] Liming Chen and Chris Nugent. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 2009. 4.2

[49] Bhawana Chhaglani, Camellia Zakaria, Adam Lechowicz, Jeremy Gummeson, and Prashant Shenoy. Flowsense: Monitoring airflow in building ventilation systems using audio sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(1), mar 2022. doi: 10.1145/3517258. URL https://doi.org/10.1145/3517258. 1, 2, 2.1.1, 2.3.2, 5, 5.1.1, 6.1, 6.1.1

[50] Prerna Chikersal, Afsaneh Doryab, Michael Tumminia, Daniella K Villalba, Janine M Dutcher, Xinwen Liu, Sheldon Cohen, Kasey G Creswell, Jennifer Mankoff, J David Creswell, et al. Detecting depression and predicting its onset using longitudinal symptoms captured by passive sensing: a machine learning approach with robust feature selection. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 28(1):1–41, 2021. 4.1.1

[51] Trishul Chilimbi et al. Project adam: Building an efficient and scalable deep learning training system. In *Proc. of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, page 571–582, USA, 2014. USENIX Association. ISBN 9781931971164. 3.2.2

[52] Cisco. Cisco Survey Reveals Close to Three-Fourths of IoT Projects Are Failing. https://newsroom.cisco.com/press-release-content?articleId=1847422, 2023. 1, 2, 2.1.3

[53] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020. 4.2

[54] Comfy. Home - Comfy. https://comfyapp.com/, 2023. 2.3.1

[55] Marios Constantinides, Sanja Šćepanović, Daniele Quercia, Hongwei Li, Ugo Sassi, and Michael Eggleston. Comfeel: Productivity is a matter of the senses too. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(4), dec 2020. doi: 10.1145/3432234. URL https://doi.org/10.1145/3432234. 2.3.2, 6.3

[56] Giorgio Conte et al. Bluesentinel: A first approach using ibeacon for an energy efficient occupancy detection system. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, page 11–19, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331449. doi: 10.1145/2676061.2674078. URL https://doi.org/10.1145/2676061.2674078. 3.1.1

[57] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, 2012. 3, 4.4.1, 4.1

[58] Daniel Crankshaw et al. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. *CoRR*, abs/1409.3809, 2014. URL http://arxiv.org/abs/1409.3809. 3.2.3

[59] Daniel Crankshaw et al. Clipper: A low-latency online prediction serving system. In *Proc. of the 14th USENIX Conf. on Networked Systems Design and Implementation*, NSDI'17, page 613–627, USA, 2017. USENIX Association. ISBN 9781931971379. 3, 3.1, 3.1.2, 3.2.1

[60] Daniel Crankshaw et al. Inferline: ML inference pipeline composition framework. *CoRR*, abs, 2018. URL http://arxiv.org/abs/1812.01776. 3.2.1, 3.2.3

[61] Antonia Creswell, Kai Arulkumaran, and Anil A Bharath. On denoising autoencoders trained to minimise binary cross-entropy. *arXiv preprint arXiv:1708.08487*, 2017. 4.3.3

[62] B De Ruyter, E Aarts, P Markopoulos, and W Ijsselsteijn. Ambient intelligence research in homelab: Engineering the user experience. In *Ambient Intelligence*, pages 49–61. Springer, New York, NY, USA, 2005. 2.2.3

[63] Jeffrey Dean et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012. 3.1, 3.2.2

[64] Christian Debes et al. Monitoring activities of daily living in smart homes: Understanding human behavior. *IEEE Signal Process. Mag.*, 33(2):81–94, 2016. 3

[65] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 5.1.3

[66] The CMU Pronouncing Dictionary, 2008. (document), 5.2, 5.3.1

[67] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. doi: 10.1109/TIT.1983.1056650. 5.1.4

[68] DSP Related. Decibels — Mathematics of the DFT. https://www.dsprelated.com/freebooks/mdft/Decibels.html, 2023. 6.2.2

[69] Rizanne Elbakly, Moustafa Elhamshary, and Moustafa Youssef. Hyrise: A robust and ubiquitous multi-sensor fusion-based floor localization system. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(3), sep 2018. doi: 10.1145/3264914. URL https://doi.org/10.1145/3264914. 2.1.1, 2.3.2, 6.1

[70] Ehsan Elhamifar and Dat Huynh. Self-supervised multi-task procedure learning from instructional videos. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 557–573, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58520-4. 4.2

[71] Mohamed Faisal Elrawy, Ali Ismail Awad, and Hesham F. A. Hamed. Intrusion detection systems for iot-based smart environments: A survey. *Journal of Cloud Computing*, 7(1), December 2018. ISSN 2192-113X. doi: 10.1186/s13677-018-0123-6. URL https://doi.org/10.1186/s13677-018-0123-6. 3.3.1

[72] Pardis Emami Naeini, Janarth Dheenadhayalan, Yuvraj Agarwal, and Lorrie Faith Cranor. Which privacy and security attributes most impact consumers' risk perception and willingness to purchase iot devices ? In *2021 IEEE Symposium on Security and Privacy(SP)*, pages 519 – 536, New York, NY, USA, 2021. IEEE. doi: 10.1109/SP40001.2021.00112. 2.1.2

[73] Antti J Eronen et al. Audio-based context recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):321–329, 2005. 3

[74] Jonathon Fagert, Mostafa Mirshekari, Pei Zhang, and Hae Young Noh. Recursive sparse representation for identifying multiple concurrent occupants using floor vibration sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(1), mar 2022. doi: 10.1145/3517229. URL https://doi.org/10.1145/3517229. 6.2

[75] Zhiyun Fan, Meng Li, Shiyu Zhou, and Bo Xu. Exploring wav2vec 2.0 on speaker verification and language identification. *arXiv preprint arXiv:2012.06185*, 2020. 5.3.1

[76] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 636–654, San Jose, California, 2016. IEEE. doi: 10.1109/SP.2016.44. 2.1.2

[77] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020. 5.1.3

[78] Francesco Fraternali, Bharathan Balaji, Yuvraj Agarwal, Luca Benini, and Rajesh Gupta. Pible: Battery-free mote for perpetual indoor ble applications: Demo abstract. In *Proceedings of the 5th Conference on Systems for Built Environments*, BuildSys '18, page 184–185, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359511. doi: 10.1145/3276774.3282823. URL https://doi.org/10.

1145/3276774.3282823. 2.2.1, 2.2.3

[79] Francesco Fraternali, Bharathan Balaji, Dezhi Hong, Yuvraj Agarwal, and Rajesh K. Gupta. Marble: Collaborative scheduling of batteryless sensors with meta reinforcement learning. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '21, page 140–149, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450391146. doi: 10.1145/3486611.3486670. URL https://doi.org/10.1145/3486611.3486670. 2.2.4

[80] John S Garofolo. Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*, 1993. 5.1.3, 5.1.4, 5.3, 5.3.1, 5.4.1, 5.4.4, 5.5.1, 5.5.4, 5.5.6

[81] Alexander Gluhak, Srdjan Krco, Michele Nati, Dennis Pfisterer, Nathalie Mitton, and Tahiry Razafindralambo. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49(11):58–67, 2011. doi: 10.1109/MCOM.2011.6069710. 2

[82] Karan Goel and Emma Brunskill. Learning procedural abstractions and evaluating discrete latent temporal structure. In *International Conference on Learning Representations*, 2018. 4.2

[83] Shadan Golestan, Sepehr Kazemian, and Omid Ardakanian. Data-driven models for building occupancy estimation. In *Proceedings of the Ninth International Conference on Future Energy Systems*, e-Energy '18, page 277–281, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357678. doi: 10.1145/3208903.3208940. URL https://doi.org/10.1145/3208903.3208940. 1, 2, 6.2

[84] Daniel Golovin et al. Google vizier: A service for black-box optimization. In *Proc. of the 23rd ACM SIGKDD Internat. conference on KDD*, pages 1487–1495, 2017. 3.2.2

[85] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021. 5.5.1

[86] Google. Tensorflow core — machine learning for beginners and experts. https://www.tensorflow.org/overview, 2020. 3.2.1, 3.2.3

[87] Google Cloud. Cloud automl. https://cloud.google.com/automl, 2020. 3.2.2

[88] Google Cloud. Derive Insights via ML. https://cloud.google.com/vision, 2020. 3.2.1, 3.2.3, 3.3.1

[89] Google Research. Coral: toolkit to build ai products. https://coral.ai/, 2020. 3.3.1, 3.4.1

[90] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. Owl 2: The next step for owl. *Journal of Web Semantics*, 6(4):309–322, 2008. 4, 4.2, 4.3.2

[91] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012. 5.3.1

[92] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural

networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006. 5, 5.3.1

[93] Andrew Greasley and Chris Owen. *Behavior in Models: A Framework for Representing Human Behavior*, pages 47–63. Springer, 01 2016. ISBN 978-1-137-53549-8. doi: 10.1057/978-1-137-53551-1_3. 4.2

[94] Sidhant Gupta et al. Electrisense: Single-point sensing using emi for electrical event detection and classification in the home. In *Proc. of the 12th ACM Internat. Conference on Ubiquitous Computing*, UbiComp '10, page 139–148, New York, NY, USA, 2010. ACM. ISBN 9781605588438. doi: 10.1145/1864349.1864375. URL https://doi.org/10.1145/1864349.1864375. 3.1.1

[95] Sasirekha GVK, Adhisaya T, Aswini P, Jyotsna Bapat, and Debabrata Das. Challenges in the design of an iot testbed. In *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 14–19, Jaipur, India, 2019. IEEE. doi: 10.1109/ICCT46177.2019.8969009. 1, 2

[96] Rim Helaoui, Daniele Riboni, and Heiner Stuckenschmidt. A probabilistic ontological framework for the recognition of multilevel human activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 345–354, 2013. 4.2

[97] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, et al. Cnn architectures for large-scale audio classification. In *2017 IEEE Internat. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135, 2017. doi: 10.1109/ICASSP.2017.7952132. 3.4.1

[98] Peter Hevesi et al. Monitoring household activities and user location with a cheap, unobtrusive thermal sensor array. In *Proc. of the 2014 ACM Internat. joint conference on pervasive and ubiquitous computing*, pages 141–145, 2014. 3.1.1

[99] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 4.3.3

[100] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. 4.3.3

[101] Wen-Chin Huang, Chia-Hua Wu, Shang-Bao Luo, Kuan-Yu Chen, Hsin-Min Wang, and Tomoki Toda. Speech recognition by simply fine-tuning bert. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7343–7347. IEEE, 2021. 5.1.3

[102] Intel. Intel® neural compute stick: A plug and play development kit for ai inferencing. https://software.intel.com/en-us/neural-compute-stick, 2020. 3.3.1

[103] Stephen S Intille, Kent Larson, Emmanuel Munguia Tapia, Jennifer S Beaudin, Pallavi Kaushik, Jason Nawyn, and Randy Rockinson. Using a live-in laboratory for ubiquitous computing research. In *International Conference on Pervasive Computing*, pages 349–365, New York, NY, USA, 2006. Springer, Springer. 2.2.3

[104] Yasha Iravantchi, Karan Ahuja, Mayank Goel, Chris Harrison, and Alanson Sample. Privacymic: Utilizing inaudible frequencies for privacy-preserving daily activity recognition. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2021. 5, 5.1.4, 5.1, 5.2.1, 5.3, 5.3, 5.4

[105] Walaa N. Ismail, Mohammad Mehedi Hassan, and Hessah A. Alsalamah. Context-enriched regular human behavioral pattern detection from body sensors data. *IEEE Access*, 7:33834–33850, 2019. doi: 10.1109/ACCESS.2019.2904122. 4.2

[106] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA, 1988. ISBN 013022278X. 4.4.1

[107] Milan Jain, Mridula Gupta, Amarjeet Singh, and Vikas Chandan. Beyond control: Enabling smart thermostats for leakage detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(1), mar 2019. doi: 10.1145/3314401. URL https://doi.org/10.1145/3314401. 6.1

[108] Cisco Jasper. The hidden costs of delivering iiot services: Industrial monitoring & heavy equipment, 2023. 2.2.4

[109] Jeffrey Dunn. Fblearner flow: Facebook's ai backbone. https://engineering.fb.com/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/, 2020. 3.2.3

[110] Jeremy Hermann and others. Meet michelangelo: Uber's machine learning platform. https://eng.uber.com/michelangelo-machine-learning-platform/, 2020. 3.2.3

[111] Jie Jiang, Riccardo Pozza, Nigel Gilbert, and Klaus Moessner. Makesense: An iot testbed for social research of indoor activities. *ACM Trans. Internet Things*, 1(3), June 2020. ISSN 2691-1914. doi: 10.1145/3381914. URL https://doi.org/10.1145/3381914. 2, 2.2.3, 2.3.1

[112] Ming Jin, Ruoxi Jia, Zhaoyi Kang, Ioannis C. Konstantakopoulos, and Costas J. Spanos. Presencesense: Zero-training algorithm for individual presence detection based on power monitoring. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, page 1–10, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331449. doi: 10.1145/2674061.2674073. URL https://doi.org/10.1145/2674061.2674073. 1, 2

[113] Johnson Controls Developers. Johnson Controls. https://www.johnsoncontrols.com/, 2023. 2, 2.2.4

[114] Rohan Kabra, Divya Saxena, Dhaval Patel, and Jiannong Cao. Time series clustering for human behavior pattern mining, 2021. URL https://arxiv.org/abs/2110.07549. 4.2

[115] Sabin Kafle and Dejing Dou. A heterogeneous clustering approach for human activity recognition. In Sanjay Madria and Takahiro Hara, editors, *Big Data Analytics and Knowledge Discovery*, pages 68–81, Cham, 2016. Springer International Publishing. ISBN 978-

3-319-43946-4. 4.2

[116] Areg Karapetyan, Sid Chi-Kin Chau, Khaled Elbassioni, Majid Khonji, and Emad Dababseh. Smart lighting control using oblivious mobile sensors. In *Proceedings of the 5th Conference on Systems for Built Environments*, BuildSys '18, page 158–167, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359511. doi: 10.1145/3276774.3276788. URL https://doi.org/10.1145/3276774.3276788. 6.2

[117] Harmanpreet Kaur, Alex C. Williams, Daniel McDuff, Mary Czerwinski, Jaime Teevan, and Shamsi T. Iqbal. Optimizing for happiness and productivity: Modeling opportune moments for transitions and breaks at work. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–15, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376817. URL https://doi.org/10.1145/3313831.3376817. 4.2

[118] Nicky Kern, Bernt Schiele, and Albrecht Schmidt. Multi-sensor activity context detection for wearable computing. In Emile Aarts, René W. Collier, Evert van Loenen, and Boris de Ruyter, editors, *Ambient Intelligence*, pages 220–232, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39863-9. 4.2

[119] Sunder Ali Khowaja, Aria Ghora Prabono, Feri Setiawan, Bernardo Nugroho Yahya, and Seok-Lyong Lee. Contextual activity based healthcare internet of things, services, and people (hiotsp): An architectural framework for healthcare monitoring using wearable sensors. *Computer Networks*, 145:190–206, 2018. 1, 2, 2.1.1, 4.2, 5

[120] Rushil Khurana et al. Gymcam: Detecting, recognizing and tracking simultaneous exercises in unconstrained scenes. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(4):1–17, 2018. 3.1.1

[121] Davi Miara Kiapuchinski, Carlos Raimundo Erig Lima, and Celso Antônio Alves Kaestner. Spectral noise gate technique applied to birdsong preprocessing on embedded unit. In *2012 IEEE International Symposium on Multimedia*, pages 24–27, 2012. doi: 10.1109/ISM.2012.12. 5.4.4

[122] Julie A. Kientz, Shwetak N. Patel, Brian Jones, Ed Price, Elizabeth D. Mynatt, and Gregory D. Abowd. The georgia tech aware home. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, page 3675–3680, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580128. doi: 10.1145/1358628.1358911. URL https://doi.org/10.1145/1358628.1358911. 2.2.3

[123] Simon Klakegg, Kennedy Opoku Asare, Niels van Berkel, Aku Visuri, Eija Ferreira, Simo Hosio, Jorge Goncalves, Hanna-Leena Huttunen, and Denzil Ferreira. Care: Context-awareness for elderly care. *Health and Technology*, 11(1):211–226, 2022. 2.2.3

[124] Jacob Kröger. Unexpected inferences from sensor data: a hidden privacy threat in the internet of things. In *Internet of Things. Information Processing in an Increasingly Connected World: First IFIP International Cross-Domain Conference, IFIPIoT 2018, Held at the 24th IFIP World Computer Congress, WCC 2018, Poznan, Poland, September 18-19, 2018, Revised Selected Papers 1*, pages 147–159. Springer, 2019. 3

[125] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the

syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 4.2

[126] Sumeet Kumar, Le T Nguyen, Ming Zeng, Kate Liu, and Joy Zhang. Sound shredding: Privacy preserved audio sensing. In *Proceedings of the 16th international workshop on mobile computing systems and applications*, pages 135–140, 2015. 5.2.1

[127] Gierad Laput and Chris Harrison. Exploring the efficacy of sparse, general-purpose sensor constellations for wide-area activity sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(2), jun 2019. doi: 10.1145/3328926. URL https://doi.org/10.1145/3328926. 2.1.1, 2.3.2, 6.3

[128] Gierad Laput, Yang Zhang, and Chris Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3986–3999, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025773. URL https://doi.org/10.1145/3025453.3025773. 2, 2.3.1, 6.3, 7.1.1

[129] Gierad Laput, Yang Zhang, and Chris Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3986–3999, 2017. 5, 5.1, 5.3, 5.4, 5.5.1, 5.5.2, 5.5.5

[130] Gierad Laput, Yang Zhang, and Chris Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proc. of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 3986–3999, New York, NY, USA, 2017. ACM. ISBN 9781450346559. doi: 10.1145/3025453.3025773. URL https://doi.org/10.1145/3025453.3025773. 3, 3.1.1, 3.3.1, 3.3.2

[131] Gierad Laput, Karan Ahuja, Mayank Goel, and Chris Harrison. Ubicoustics: Plug-and-play acoustic activity recognition. In *Proc. of the 31st Annual ACM Symposium on UIST*, UIST '18, page 213–224, New York, NY, USA, 2018. ACM. ISBN 9781450359481. doi: 10.1145/3242587.3242609. URL https://doi.org/10.1145/3242587.3242609. 3, 3.1.1, 3.2.1, 3.3.1, 3.4.1

[132] Gierad Laput, Karan Ahuja, Mayank Goel, and Chris Harrison. Ubicoustics: Plug-and-play acoustic activity recognition. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 213–224, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359481. doi: 10.1145/3242587.3242609. URL https://doi.org/10.1145/3242587.3242609. 5.1.1

[133] Eric C Larson, TienJui Lee, Sean Liu, Margaret Rosenfeld, and Shwetak N Patel. Accurate and privacy preserving cough sensing using a low-cost microphone. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 375–384, 2011. 5, 5.1.1, 5.1, 5.2.1, 5.3, 5.3, 5.3.1, 5.4, 5.3.2, 5.5.1, 5.5.2, 5.5.2, 5.5.5

[134] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *mhwombat*, 2010. URL http://yann.lecun.com/exdb/mnist/. 3.1.1, 3.4.1

[135] Xin Lei, Andrew W Senior, Alexander Gruenstein, and Jeffrey Sorensen. Accurate and compact large vocabulary speech recognition on mobile devices. In *Interspeech*, volume 1,

2013. 3.2.1

[136] Du Li, Bharathan Balaji, Yifei Jiang, and Kshitiz Singh. A wi-fi based occupancy sensing approach to smart energy in commercial office buildings. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '12, page 197–198, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311700. doi: 10.1145/2422531.2422568. URL https://doi.org/10.1145/2422531.2422568. 2, 2.2.3

[137] Kang Li and Yun Fu. Prediction of human activity by discovering temporal sequence patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1644–1657, 2014. doi: 10.1109/TPAMI.2013.2297321. 4.2

[138] Mu Li et al. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on operating systems design and implementation (OSDI 14)*, OSDI'14, page 583–598, USA, 2014. USENIX Association. ISBN 9781931971164. 3.2.3

[139] Li Liu, Yuxin Peng, Ming Liu, and Zigang Huang. Sensor-based human activity recognition system with a multilayered model using time series shapelets. *Knowledge-Based Systems*, 90:138–152, 2015. 4.2

[140] Dimitrios Lymberopoulos, Athanasios Bamis, and Andreas Savvides. Extracting spatiotemporal human activity patterns in assisted living using a home sensor network. In *Proceedings of the 1st International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '08, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580678. doi: 10.1145/1389586.1389621. URL https://doi.org/10.1145/1389586.1389621. 4.2

[141] Donald M MacKay. Towards an information-flow model of human behaviour. *British Journal of Psychology*, 47(1):30–43, 1956. 4.2

[142] Naveen Sai Madiraju, Seid M. Sadat, Dimitry Fisher, and Homa Karimabadi. Deep temporal clustering : Fully unsupervised learning of time-domain features, 2018. URL https://arxiv.org/abs/1802.01059. 4.3.3, 4.3.3

[143] Nathan Malkin, Joe Deatrick, Allen Tong, Primal Wijesekera, Serge Egelman, and David Wagner. Privacy attitudes of smart speaker users. *Proceedings on Privacy Enhancing Technologies*, 2019(4):250–271, 2019. 5, 5.3.1, 5.5.1, 5.5.2

[144] Cecilia Mascolo. Listen to your health: Reflections on mobile health diagnostics through audio signals. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, pages 1–1, 2020. 5.1.1

[145] Matrix Labs. MATRIX is the World's Most Popular App Ecosystem for the Convergence of AI and Internet of Things. https://matrix.one/, 2023. 2, 2.2.1

[146] P. McCullagh et al. *Generalized Linear Models*. Chapman & Hall, London, 1989. 3.3.2

[147] Georgios Meditskos and Ioannis Kompatsiaris. iknow: Ontology-driven situational awareness for the recognition of activities of daily living. *Pervasive and Mobile Computing*, 40: 17–41, 2017. 1, 2, 2.1.1, 4.2

[148] Georgios Meditskos, Stamatia Dasiopoulou, and Ioannis Kompatsiaris. Metaq: A

knowledge-driven framework for context-aware activity recognition combining sparql and owl 2 activity patterns. *Pervasive and Mobile Computing*, 25:104–124, 2016. 4, 4.2, 4.3.2, 4.3.2

[149] Microsoft. Azure iot. https://azure.microsoft.com/en-us/overview/iot/, 2020. 3.2.1

[150] Microsoft. Azure IoT – Internet of Things Platform — Microsoft Azure. https://azure.microsoft.com/en-us/overview/iot/, 2023. 2.2.2

[151] Mostafa Mirshekari, Shijia Pan, Adeola Bannis, Yan Pui Mike Lam, Pei Zhang, and Hae Young Noh. Step-level person localization through sparse sensing of structural vibration. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, page 376–377, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334754. doi: 10.1145/2737095.2742924. URL https://doi.org/10.1145/2737095.2742924. 2.3.1

[152] Mites.io. Mites.io: a full-stack ubiquitous sensing platform. https://mites.io/, 2023. 3, 3.1.1, 3.3.2

[153] Mites.io developers. Mites.io - TCS Hall Deployment FAQs. https://mites.io/TCSDeployment/Mites-FAQs.pdf, 2023. 7.1.5

[154] MLFlow. MLflow - A platform for the machine learning lifecycle — MLflow. https://mlflow.org/, 2023. 2.2.2

[155] Vimal Mollyn, Karan Ahuja, Dhruv Verma, Chris Harrison, and Mayank Goel. Samosa: Sensing activities with motion and subsampled audio. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–19, 2022. (document), 5, 5.1.4, 5.1, 5.2.1, 5.3, 5.4, 5.5.1, 5.5.2, 5.5.2, 5.5.3, 5.11, 5.5.5

[156] Mehrab Bin Morshed, Koustuv Saha, Richard Li, Sidney K. D'Mello, Munmun De Choudhury, Gregory D. Abowd, and Thomas Plötz. Prediction of mood instability with passive sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(3), sep 2019. doi: 10.1145/3351233. URL https://doi.org/10.1145/3351233. 4.2

[157] Srinarayana Nagarathinam, Arunchandar Vasan, Venkatesh Sarangan, Rajesh Jayaprakash, and Anand Sivasubramaniam. Good set-points make good neighbors: User seating and temperature control in uberized workspaces. In *Proceedings of the 5th Conference on Systems for Built Environments*, BuildSys '18, page 144–147, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359511. doi: 10.1145/3276774.3276781. URL https://doi.org/10.1145/3276774.3276781. 1, 2

[158] Balakrishnan Narayanaswamy, Bharathan Balaji, Rajesh Gupta, and Yuvraj Agarwal. Data driven investigation of faults in hvac systems with model, cluster and compare (mcc). In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, page 50–59, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331449. doi: 10.1145/2674061.2674067. URL https://doi.org/10.1145/2674061.2674067. 2.1.1, 2.3.1, 6.1

[159] Michele Nati, Alexander Gluhak, Hamidreza Abangar, and William Headley. Smartcampus: A user-centric testbed for internet of things experimentation. In *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 1–6, New York, NY, USA, June 2013. WPMC. 2.2.3

[160] Carman Neustaedter and Phoebe Sengers. Autobiographical design in hci research: Designing and learning through use-it-yourself. In *Proceedings of the Designing Interactive Systems Conference*, DIS '12, page 514–523, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312103. doi: 10.1145/2317956.2318034. URL https://doi.org/10.1145/2317956.2318034. 2.3.1

[161] Notion. Notion DIY Smart Monitoring System. https://getnotion.com/, 2023. 2.2.1

[162] Nvidia. Jetson nano developer kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit, 2020. 3.3.1

[163] Joseph D O'Connor. *Better English Pronunciation*. Cambridge University Press, 1980. 5.1.2

[164] George Okeyo, Liming Chen, Hui Wang, and Roy Sterritt. Ontology-based learning framework for activity assistance in an adaptive smart home. In *Activity recognition in pervasive intelligent environments*, pages 237–263. Springer, 2011. 4.2

[165] Temitope Oluwafemi, Tadayoshi Kohno, Sidhant Gupta, and Shwetak Patel. Experimental security analyses of Non-Networked compact fluorescent lamps: A case study of home automation security. In *LASER 2013 (LASER 2013)*, pages 13–24, Arlington, VA, October 2013. USENIX Association. ISBN 978-1-931971-06-5. URL https://www.usenix.org/laser2013/program/oluwafemi. 2.1.2

[166] Kumar Padmanabh, Adi Malikarjuna, Sougata Sen, Siva Prasad Katru, Amrit Kumar, Sai Pawankumar C, Sunil Kumar Vuppala, and Sanjoy Paul. Isense: A wireless sensor network based conference room management system. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, page 37–42, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605588247. doi: 10.1145/1810279.1810288. URL https://doi.org/10.1145/1810279.1810288. 2, 2.1.1, 6.1

[167] Madhurananda Pahar, Igor Miranda, Andreas Diacon, and Thomas Niesler. Automatic non-invasive cough detection based on accelerometer and audio signals. *Journal of Signal Processing Systems*, 94(8):821–835, 2022. 5

[168] Shijia Pan, Tong Yu, Mostafa Mirshekari, Jonathon Fagert, Amelie Bonde, Ole J. Mengshoel, Hae Young Noh, and Pei Zhang. Footprintid: Indoor pedestrian identification through ambient structural vibration sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3), sep 2017. doi: 10.1145/3130954. URL https://doi.org/10.1145/3130954. 6.2

[169] Panasonic. AMG8853 - Infrared Array Sensor GridEYE - Built-in Sensors - Industrial Devices & Solutions - Panasonic. https://industrial.panasonic.com/ww/products/pt/grid-eye/models/AMG8853, 2023. 2.3.2

[170] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015. 5.1.3, 5.1.4, 5.3.2

[171] Particle. Particle - Welcome to real IoT. `https://particle.io/`, 2023. 7.1.1

[172] Adam Paszke et al. Automatic differentiation in pytorch, 2017. 3.4.1

[173] F. Pedregosa, G. Varoquaux, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 3.4.1

[174] Yi-Hao Peng, Ming-Wei Hsi, Paul Taele, Ting-Yu Lin, Po-En Lai, Leon Hsu, Tzu-chuan Chen, Te-Yen Wu, Yu-An Chen, Hsien-Hui Tang, and Mike Y. Chen. Speechbubbles: Enhancing captioning experiences for deaf and hard-of-hearing people in group conversations. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–10, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173867. URL `https://doi.org/10.1145/3173574.3173867`. 5.1.1

[175] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1):414–454, 2013. 4.2

[176] Phidgets Inc. Phidgets Inc. - Products for USB Sensing and Control. `https://www.phidgets.com`, 2023. 2.2.1

[177] Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. ACM Press, 2015. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2806390. URL `http://dl.acm.org/citation.cfm?doid=2733373.2806390`. 5.4.1, 5.4.4, 5.5.1, 5.5.2

[178] Pincelate, 2008. 5.3.1

[179] Preeja Pradeep and Shivsubramani Krishnamoorthy. The mom of context-aware systems: A survey. *Computer Communications*, 137:44–69, 2019. 4.2

[180] PTC. ThingWorx: Industrial IoT Software — IIoT Platform — PTC. `https://www.ptc.com/en/products/thingworx`, 2023. 2.2.2

[181] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022. 5, 5.1.3, 5.3.2, 5.4

[182] Haroon Rashid, Nipun Batra, and Pushpendra Singh. Rimor: Towards identifying anomalous appliances in buildings. In *Proceedings of the 5th Conference on Systems for Built Environments*, BuildSys '18, page 33–42, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359511. doi: 10.1145/3276774.3276797. URL `https://doi.org/10.1145/3276774.3276797`. 1, 2

[183] Haroon Rashid, Nipun Batra, and Pushpendra Singh. Rimor: Towards identifying anomalous appliances in buildings. In *Proceedings of the 5th Conference on Systems for Built Environments*, BuildSys '18, page 33–42, New York, NY, USA, 2018. Association for

Computing Machinery. ISBN 9781450359511. doi: 10.1145/3276774.3276797. URL https://doi.org/10.1145/3276774.3276797. 6.1

[184] Raspberry Pi. Buy a Sense HAT – Raspberry Pi. https://www.raspberrypi.com/products/sense-hat/, 2023. 2

[185] Raspberry Pi Foundation. Raspberry pi 4 model b. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/, 2020. 3.1.2, 3.4.1

[186] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, et al. Speechbrain: A general-purpose speech toolkit. *arXiv preprint arXiv:2106.04624*, 2021. 5.1.4, 5.3.1, 5.3.1

[187] Reza Rawassizadeh, Elaheh Momeni, Chelsea Dobbins, Joobin Gharibshah, and Michael Pazzani. Scalable daily human behavioral pattern mining from multivariate temporal data. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):3098–3112, 2016. doi: 10.1109/TKDE.2016.2592527. 4.2

[188] Raytac. Raytac MDBT40-P256V3 Nordic nRF51822 Bluetooth Module. https://www.raytac.com/product/ins.php?index_id=66, 2023. 2.3.1

[189] Chandan KA Reddy, Ebrahim Beyrami, Jamie Pool, Ross Cutler, Sriram Srinivasan, and Johannes Gehrke. A scalable noisy speech dataset and online subjective test framework. *Proc. Interspeech 2019*, pages 1816–1820, 2019. 5.4.4

[190] Joseph Redmon et al. You only look once: Unified, real-time object detection. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 779–788, Las Vegas, NV, USA, 2016. IEEE. 3.1.1, 3.1.2

[191] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, Montreal, Quebec, Canada, 2015. IEEE. 3.1.1

[192] Daniele Riboni and Claudio Bettini. Cosar: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289, 2011. 4.2

[193] Daniele Riboni and Claudio Bettini. Owl 2 modeling and reasoning with complex human activities. *Pervasive and Mobile Computing*, 7(3):379–395, 2011. 4.2

[194] Daniele Riboni, Claudio Bettini, Gabriele Civitarese, Zaffar Haider Janjua, and Rim Helaoui. Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. In *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 149–154. IEEE, 2015. 4.2

[195] Daniele Riboni, Claudio Bettini, Gabriele Civitarese, Zaffar Haider Janjua, and Rim Helaoui. Smartfaber: Recognizing fine-grained abnormal behaviors for early detection of mild cognitive impairment. *Artificial intelligence in medicine*, 67:57–74, 2016. 4.2

[196] Daniele Riboni, Timo Sztyler, Gabriele Civitarese, and Heiner Stuckenschmidt. Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, page 1–12, New York, NY, USA, 2016.

Association for Computing Machinery. ISBN 9781450344616. doi: 10.1145/2971648.2971691. URL https://doi.org/10.1145/2971648.2971691. 4.2

[197] Vladimir Risojević, Robert Rozman, Ratko Pilipović, Rok Češnovar, and Patricio Bulić. Accurate indoor sound level measurement on a low-power and low-cost wireless sensor node. *Sensors*, 18(7):123–132, 2018. ISSN 1424-8220. doi: 10.3390/s18072351. URL https://www.mdpi.com/1424-8220/18/7/2351. 2.2.1

[198] Alex Rogers, Siddhartha Ghosh, Reuben Wilcock, and Nicholas R. Jennings. A scalable low-cost solution to provide personalised home heating advice to households. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, page 1–8, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324311. doi: 10.1145/2528282.2528284. URL https://doi.org/10.1145/2528282.2528284. 1, 2

[199] A. Rowe, M. E. Berges, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett, J. M. F. Moura, and L. Soibelman. Sensor andrew: Large-scale campus-wide sensing and actuation. *IBM Journal of Research and Development*, 55(1.2):6:1–6:14, 2011. doi: 10.1147/JRD.2010.2089662. 2, 2.2.3

[200] S. Harris, A. Seaborne. SPARQL 1.1 query language, W3C recommendation. http://www.w3.org/TR/sparql11-query/, 2022. 4, 4.3.2

[201] Manaswi Saha et al. Energylens: Combining smartphones with electricity meter for accurate activity detection and user annotation. In *Proceedings of the 5th International Conference on Future Energy Systems*, e-Energy '14, page 289–300, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328197. doi: 10.1145/2602044.2602058. URL https://doi.org/10.1145/2602044.2602058. 3.1.1

[202] Samsung. Smartthings. https://www.smartthings.com, 2020. 3.1.2

[203] Oliver Shih, Patrick Lazik, and Anthony Rowe. Aures: A wide-band ultrasonic occupancy sensing platform. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, BuildSys '16, page 157–166, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342643. doi: 10.1145/2993422.2993580. URL https://doi.org/10.1145/2993422.2993580. 6.2

[204] Siemens Developers. Siemens. https://www.siemens.com/global/en.html, 2023. 2

[205] M. Song, H. Li, and H. Wu. A decentralized load balancing architecture for cache system. In *2015 Internat. Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 114–119, Xi'an, China, 2015. IEEE. doi: 10.1109/CyberC.2015.44. 3.3.1

[206] Qun Song, Chaojie Gu, and Rui Tan. Deep room recognition using inaudible echos. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–28, 2018. 5.1.1

[207] Evan R Sparks, Shivaram Venkataraman, Tomer Kaftan, Michael J Franklin, and Benjamin Recht. Keystoneml: Optimizing pipelines for large-scale advanced analytics. In

2017 IEEE 33rd Internat. conference on data engineering (ICDE), pages 535–546, New York, NY, USA, 2017. IEEE, IEEE. 3.1, 3.2.2

[208] TDK InvenSense. MPU-6500. https://www.digikey.com/en/products/detail/tdk-invensense/MPU-6500/4385413, 2023. 2.3.1

[209] Texas Instruments. TIDC- SENSORTAG. https://www.ti.com/tool/TIDC-CC2650STK-SENSORTAG, 2023. 2.2.1, 2.3.1

[210] The Huntington News. NU administration removes occupancy sensors in ISEC in response to privacy, ethical concerns. https://rb.gy/i1op2o, 2023. 1, 2

[211] The Verge. A new hack can turn an Echo into a live microphone. https://www.theverge.com/2017/8/1/16079044/amazon-echo-hack-microphone-listen-in-mark-barnes, 2023. 5

[212] Tom Torfs, Tom Sterken, Steven Brebels, Juan Santana, Richard van den Hoven, Vincent Spiering, Nicolas Bertsch, Davide Trapani, and Daniele Zonta. Low power wireless sensor network for building monitoring. *IEEE Sensors Journal*, 13(3):909–915, 2012. 2

[213] Ikram Ud Din, Mohsen Guizani, Suhaidi Hassan, Byung-Seo Kim, Muhammad Khurram Khan, Mohammed Atiquzzaman, and Syed Hassan Ahmed. The internet of things: A review of enabled technologies and future challenges. *IEEE Access*, 7:7606–7640, 2019. doi: 10.1109/ACCESS.2018.2886601. 2

[214] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, 16(4): 62–74, 2017. doi: 10.1109/MPRV.2017.3971131. 4.2

[215] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE pervasive computing*, 16(4): 62–74, 2017. 3, 4.4.1, 4.1

[216] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 4.3.3

[217] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. *Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software*, pages 165–186. Atlantis Press, Paris, 2011. ISBN 978-94-91216-05-3. doi: 10.2991/978-94-91216-05-3_8. URL https://doi.org/10.2991/978-94-91216-05-3_8. 4.2

[218] VergeSense Developers. Home — VergeSense Great decisions require great data. https://vergesense.com/, 2023. 2.3.1

[219] Patrick Verkaik, Yuvraj Agarwal, Rajesh Gupta, and Alex C. Snoeren. Softspeak: Making voip play well in existing 802.11 deployments. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, page 409–422, USA, 2009. USENIX Association. 2.3.3

[220] Claudia Villalonga, Muhammad Asif Razzaq, Wajahat Ali Khan, Hector Pomares, Ignacio Rojas, Sungyoung Lee, and Oresti Banos. Ontology-based high-level context inference for human behavior identification. *Sensors*, 16(10):1617, 2016. 4, 4.2, 4.3.2, 4.3.2, 4.4.2, 4.1

[221] Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. . net gadgeteer: A platform for custom devices. In *International Conference on Pervasive Computing*, pages 216–233, Berlin, Heidelberg, 2012. Springer, Springer. 2.2.1

[222] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294. 4.3.3

[223] Nazar Waheed, Xiangjian He, Muhammad Ikram, Muhammad Usman, Saad Sajid Hashmi, and Muhammad Usman. Security and privacy in iot using machine learning and blockchain: Threats and countermeasures. *ACM Computing Surveys (CSUR)*, 53(6): 1–37, 2020. 3

[224] Edward J. Wang et al. Magnifisense: Inferring device interaction using wrist-worn passive magneto-inductive sensors. In *Proc. of the 2015 ACM Internat. Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, page 15–26, New York, NY, USA, 2015. ACM. ISBN 9781450335744. doi: 10.1145/2750858.2804271. URL https://doi.org/10.1145/2750858.2804271. 3.1.1

[225] Wei Wang, Jinyang Gao, Meihui Zhang, et al. Rafiki: Machine learning as an analytics service system. *Proc. VLDB Endow.*, 12(2):128–140, October 2018. ISSN 2150-8097. doi: 10.14778/3282495.3282499. URL https://doi.org/10.14778/3282495.3282499. 3.2.1

[226] Yingzhi Wang, Abdelmoumene Boumadane, and Abdelwahab Heba. A fine-tuned wav2vec 2.0/hubert benchmark for speech emotion recognition, speaker verification and spoken language understanding. *arXiv preprint arXiv:2111.02735*, 2021. 5.3.1

[227] WELL. WELL — IWBI. https://www.wellcertified.com/, 2023. 1, 2, 6.1, 6.2.2

[228] WELL. WELL PERFORMANCE RATING — Acoustic Performance. https://v2.wellcertified.com/en/performance-rating/acoustic%20performance/feature/1, 2023. 6.2.2

[229] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016. 4.3.3

[230] Doris Xin, Stephen Macke, Litian Ma, Jialin Liu, Shuchen Song, and Aditya Parameswaran. Helix: Holistic optimization for accelerating iterative machine learning. *Proc. VLDB Endow.*, 12(4):446–460, December 2018. ISSN 2150-8097. doi: 10.14778/3297753.3297763. URL https://doi.org/10.14778/3297753.3297763. 3, 3.1, 3.1.2, 3.2.2

[231] Frank F. Xu, Lei Ji, Botian Shi, Junyi Du, Graham Neubig, Yonatan Bisk, and Nan Duan. A benchmark for structured procedural knowledge extraction from cooking videos, 2020. URL https://arxiv.org/abs/2005.00706. 4.2

[232] M. Xu, S. Alamro, T. Lan, and S. Subramaniam. Laser: A deep learning approach for speculative execution and replication of deadline-critical jobs in cloud. In *2017 26th Internat. Conference on Comp. Comm. and Networks (ICCCN)*, pages 1–8, Vancouver, BC, 2017. IEEE. doi: 10.1109/ICCCN.2017.8038373. 3, 3.1.2, 3.2.1

[233] Teng Xu, James B. Wendt, and Miodrag Potkonjak. Security of iot systems: Design challenges and opportunities. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 417–423, San Jose, California, 2014. IEEE. doi: 10.1109/ICCAD.2014.7001385. 2.2.4

[234] Xuhai Xu, Ebrahim Nemati, Korosh Vatanparvar, Viswam Nathan, Tousif Ahmed, Md Mahbubur Rahman, Daniel McCaffrey, Jilong Kuang, and Jun Alex Gao. Listen2cough: Leveraging end-to-end deep learning cough detection model to enhance lung health assessment using passively sensed audio. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1), 2021. doi: 10.1145/3448124. URL `https://doi.org/10.1145/3448124`. 5.1.1, 5.1.4

[235] Liang Yang, Zimu Zheng, Jingting Sun, Dan Wang, and Xin Li. A domain-assisted data driven model for thermal comfort prediction in buildings. In *Proceedings of the Ninth International Conference on Future Energy Systems*, e-Energy '18, page 271–276, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357678. doi: 10.1145/3208903.3208914. URL `https://doi.org/10.1145/3208903.3208914`. 1, 2

[236] Juan Ye, Graeme Stevenson, and Simon Dobson. Usmart: An unsupervised semantic mining activity recognition technique. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 4(4):1–27, 2014. 4.2

[237] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340472. doi: 10.1145/2834050.2834095. URL `https://doi.org/10.1145/2834050.2834095`. 2.1.2

[238] Chunhui Yuan and Haitao Yang. Research on k-value selection method of k-means clustering algorithm. *J*, 2(2):226–235, 2019. ISSN 2571-8800. doi: 10.3390/j2020016. URL `https://www.mdpi.com/2571-8800/2/2/16`. 4.3.3

[239] Matei Zaharia et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proc. of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, page 2, USA, 2012. USENIX Association. 3.2.3

[240] Matei Zaharia et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45, 2018. 3.2.3

[241] Shaila Zaman, Amanveer Wesley, Dennis Rodrigo Da Cunha Silva, Pradeep Buddharaju, Fatema Akbar, Ge Gao, Gloria Mark, Ricardo Gutierrez-Osuna, and Ioannis Pavlidis. Stress and productivity patterns of interrupted, synergistic, and antagonistic office activities. *Scientific data*, 6(1):1–18, 2019. 1, 2, 2.1.1

[242] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419. IEEE, 2018. 4.3.3

[243] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online deep clustering for unsupervised representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4.3.3

[244] Yunke Zhang, Fengli Xu, Tong Li, Vassilis Kostakos, Pan Hui, and Yong Li. Passive health monitoring using large scale mobility data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1), 2021. doi: 10.1145/3448078. URL `https://doi.org/10.1145/3448078`. 5.1.1

[245] Jiangchuan Zheng and Lionel M. Ni. An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, page 153–162, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312240. doi: 10.1145/2370216.2370241. URL `https://doi.org/10.1145/2370216.2370241`. 4.2

[246] Xianrui Zheng, Chao Zhang, and Philip C Woodland. Adapting gpt, gpt-2 and bert language models for speech recognition. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 162–168. IEEE, 2021. 5.1.3

[247] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 4.2

[248] Jan Henrik Ziegeldorf, Oscar Garcia Morchon, and Klaus Wehrle. Privacy in the internet of things: threats and challenges. *Security and Communication Networks*, 7(12):2728–2742, 2014. 2.1.2