

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Repurposing the Ubiquitous Acoustic Devices for Cross-Modality Sensing

Permalink

<https://escholarship.org/uc/item/1s80d5r1>

Author

Sun, Ke

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Repurposing the Ubiquitous Acoustic Devices for Cross-Modality Sensing

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Ke Sun

Committee in charge:

Professor Xinyu Zhang, Chair
Professor Rajesh K. Gupta
Professor Patrick Pannuto
Professor Tauhidur Rahman
Professor Edward Wang

2024

Copyright

Ke Sun, 2024

All rights reserved.

The Dissertation of Ke Sun is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

TABLE OF CONTENTS

Dissertation Approval Page	iii
Table of Contents	iv
List of Figures	viii
List of Tables	xii
Acknowledgements	xiv
Vita	xvii
Abstract of the Dissertation	xviii
Chapter 1 Introduction	1
1.1 Opportunities for Repurposing Acoustic Sensors and Actuators in Ubiquitous Devices	2
1.1.1 Repurposing Loudspeakers and Microphones for Ultrasonic Sonar	2
1.1.2 Repurposing Actuators for Inaudible Sound Generation	3
1.1.3 Transforming Various Sensors into Side-channel Microphones	3
1.1.4 Repurposing Microphones for Daily-Life Sound Logging	4
1.2 Dissertation Contributions	5
Chapter 2 Automatic Speech Privacy Protection Against Voice Assistants	9
2.1 Introduction	9
2.2 Related Works	12
2.2.1 Hidden Command Attacks and Defenses	12
2.2.2 Audio Privacy Leakage and Protection	14
2.3 Threat Analysis	15
2.4 Automatic Jamming Control	17
2.4.1 A Primer of Selective Jamming	17
2.4.2 Jamming Control Pipeline	19
2.4.3 Minimizing Wake Word Misdetection	21
2.4.4 Maximizing Private Speech Mute Rate	22
2.4.5 When to Resume Jamming?	23
2.5 Practical Jamming Design	24
2.5.1 Inaudible Jamming Sound	24
2.5.2 Jamming a Single Microphone	25
2.5.3 Jamming a Microphone Array	26
2.6 Implementation	34
2.6.1 Hardware	34
2.6.2 Software	34
2.7 Experimental Evaluations	35

2.7.1	Micro Benchmarks	35
2.7.2	Latency Analysis	37
2.7.3	Energy Consumption	38
2.7.4	Generalization	41
2.8	Limitations and Future Work	43
2.9	Conclusion	44
2.10	Acknowledgments	44
Chapter 3	Stealing Permission-protected Private Information From Smartphone Voice Assistant Using Zero-Permission Sensors	45
3.1	Introduction	45
3.2	Related Works	49
3.2.1	Motion Leakage via Sensors on Smartphone	49
3.2.2	Speech Recognition via Motion Sensors	50
3.2.3	Spoken Language Understanding	51
3.3	Threat Analysis	52
3.4	Motion Sensor Signal (MSS) Preprocessing	55
3.4.1	Choosing the Sensor Type and Channel	55
3.4.2	Real-time Detection and Segmentation of Voice in MSS	56
3.4.3	Feature Extraction	57
3.4.4	VUI Response Identification	58
3.5	Inferring Privacy from a Single VUI Response	59
3.5.1	Problem Statement	59
3.5.2	Model Design	61
3.5.3	Training Strategy	63
3.6	Extracting Permission-protected Privacy	65
3.6.1	One-Time Stealing	65
3.6.2	Short-Term Contextual Inference	66
3.6.3	Long-Term Monitoring	68
3.7	Defense Against StealthyIMU	69
3.7.1	Predistortion of Speech Signals	69
3.7.2	Redesigning the Permissions	70
3.8	Dataset and Implementation	70
3.8.1	StealthyIMU Dataset	70
3.8.2	Implementation	73
3.9	Evaluation	74
3.9.1	DNN Model Ablation Study	74
3.9.2	One-time Stealing	78
3.9.3	Short-term Contextual Inference	79
3.9.4	Long-term Monitoring	80
3.9.5	Generalization	82
3.9.6	System Overhead Evaluation	85
3.9.7	Defense Evaluation	86
3.10	Conclusion	88

3.11	Acknowledgments	89
Chapter 4	Single-Channel Speech Enhancement Using Ultrasound on a Smartphone .	90
4.1	Introduction	90
4.2	Related Work	93
4.2.1	Audio-only Speech Enhancement	93
4.2.2	Multi-modal Speech Enhancement	95
4.2.3	Device-free Ultrasonic Sensing	96
4.3	Sensing the Articulatory Gestures	96
4.3.1	Transmitted Ultrasound Signals Design	98
4.3.2	Mitigating Sensing Interference	99
4.4	An Overview of UltraSE DNN Model	100
4.5	DNN Input Feature Design	101
4.6	Multi-modal Fusion Design	104
4.6.1	Two-stream Feature Embedding	104
4.6.2	Speech and Ultrasound Fusion Network	106
4.7	cGAN-based Cross-modal Training	106
4.7.1	Cross-modal Similarity Measurement	109
4.7.2	cGAN-based Model Training	112
4.8	Multi-domain Speech Enhancement	114
4.8.1	Understanding the Pros and Cons of T-F Domains Speech Enhancement	114
4.8.2	Two-stage Multi-domain Network Design	116
4.9	UltraSE Implementation	118
4.9.1	UltraSpeech Dataset	118
4.9.2	UltraSE DNN Implementation	119
4.10	Experimental Evaluation	119
4.10.1	Micro Benchmark Comparison	120
4.10.2	Ablation Study	122
4.10.3	System Efficiency	123
4.10.4	Generalization	124
4.11	Conclusion	128
4.12	Acknowledgments	129
Chapter 5	Multimodal Daily-life Logging in Free-living Environments Using Non- Visual Egocentric Sensors on a Smartphone	130
5.1	Introduction	130
5.2	Related work	136
5.3	EgoADL Setup and Data Collection	138
5.4	Preliminary Study	144
5.4.1	Advantages of Egocentric Sensing	144
5.4.2	Sensing Modality Selection	147
5.5	EgoADL Supervised Learning	148
5.5.1	Problem Formulation	148
5.5.2	Input and Output Design	149

5.5.3	MMFWSF Transformer	150
5.5.4	Training Strategy	153
5.6	EgoADL Self-supervised Learning	154
5.6.1	Single-modal Self-Supervised Deep Clustering	154
5.6.2	Cross-Modal Self-Supervised Deep Clustering	156
5.7	Knowledge Distillation from Natural Language Labels	157
5.7.1	Label Refinement	157
5.7.2	Distilling Contextual Information from Text	159
5.8	Implementation and Experimental Evaluation	160
5.8.1	EgoADL Implementation and Evaluation Metrics	160
5.8.2	Micro Benchmark Analysis of EgoADL Supervised Learning Model ...	161
5.8.3	Accuracy, Generalization and Extensibility of EgoADL SSL Model	162
5.8.4	Evaluating the Limits of Non-Visual Sensors	167
5.8.5	Evaluation on Knowledge Distillation from Natural Language	170
5.8.6	Energy Consumption	172
5.9	Discussion and Limitations	173
5.10	Conclusion	175
5.11	Acknowledgments	176
Chapter 6	Conclusion and Future Work	177
6.1	Dissertation Conclusion	177
6.2	Future Work	180
6.2.1	Multi-device Multi-modal Sensing on Ubiquitous Acoustic Devices	180
6.2.2	Resource-efficient Cross-Modality Sensing on Ubiquitous Acoustic De- vices	180
6.2.3	Enabling Human-AI-Sensor Interaction for Acoustic Sensing and Privacy Protection	181
6.2.4	Decoding Semantic Boundaries for Cross-Modality Sensing	182
Bibliography	183

LIST OF FIGURES

Figure 2.1.	MicShield acts as a companion device with the VAs.	10
Figure 2.2.	Analysis of “Alexa” speech signals.	17
Figure 2.3.	Proof-of-the-concept experiments for jamming control pipeline design. ...	19
Figure 2.4.	MicShield automatic jamming control pipeline.	20
Figure 2.5.	Analysis of private speech: “Pizzerias are convenient for a quick lunch.” ..	25
Figure 2.6.	Jamming effectiveness v.s. SNR. Frequency distortion jamming can effectively protect the speech privacy for the single-microphone VA when the speech SNR is less than -15 dB.	27
Figure 2.7.	Jamming effectiveness for different jamming methods.	28
Figure 2.8.	SPL heat map for ultrasonic transducer.	29
Figure 2.9.	Acoustic overloading of received microphone. (a) zoomed-in saturated jammed signal in time domain; (b) spectrogram of jammed signal.	30
Figure 2.10.	Geometric model for MicShield design.	30
Figure 2.11.	Hardware implementations and setups of MicShield.	31
Figure 2.12.	Software stack design and implementations.	31
Figure 2.13.	Micro benchmark using the wake word of “Alexa”.	32
Figure 2.14.	Analysis of system time consumption.	37
Figure 2.15.	An example instantaneous power.	38
Figure 2.16.	Micro benchmark using the wake words of “OK Google” & “Hey Google”.	40
Figure 3.1.	StealthyIMU threat model.	46
Figure 3.2.	Motion sensor channels SNR	56
Figure 3.3.	Ultra Lightweight Voice Detection Pipeline	57
Figure 3.4.	Feature Extraction and Speaker Identification	58
Figure 3.5.	Example to convert the navigation voice text to private intents	59

Figure 3.6.	Model Architecture	61
Figure 3.7.	Knowledge Distillation Training Strategy.	63
Figure 3.8.	Trace and commands in navigation	65
Figure 3.9.	Pipeline of the speech predistortion defense.	69
Figure 3.10.	Google Assistant Voice Identification.	77
Figure 3.11.	GPS trace recovery examples	78
Figure 3.12.	Short-term GPS distance error	81
Figure 3.13.	Long-term attack results	81
Figure 3.14.	Long-term home address inference.	81
Figure 3.15.	Generalization across different smartphones, volume levels, and motion artifacts.	84
Figure 3.16.	Predistortion Speech Defense	87
Figure 4.1.	UltraSE targets the scenario where the user holds the smartphone to record the speech in a noisy environment. UltraSE uses ultrasound sensing as a complementary modality to separate the desired speaker’s voice from interferences.	91
Figure 4.2.	T-F domain features of an example speech segment: “Don’t ask me to carry an oily rage like that.”	97
Figure 4.3.	DNN input feature design	103
Figure 4.4.	Overview of UltraSE’s multi-modal multi-domain DNN design. Convolution layer notation: Channels@Kernel size	104
Figure 4.5.	Two-stream feature embedding. Channels@Kernel size in convolution layer.	105
Figure 4.6.	Architecture of the T-F domain cross-modal similarity measurement network (<i>i.e.</i> , the Discriminator).	109
Figure 4.7.	Overview of UltraSE’s cGAN-based cross-modal training.	110
Figure 4.8.	PDF of outputs from the cross-modal similarity measurement network.	111
Figure 4.9.	Benchmark of the T-F domain methods.	115

Figure 4.10.	T domain phase network. Channels@Kernel size in convolution layer. . . .	117
Figure 4.11.	Noisy SiSNR v.s. Enhanced SiSNR.	122
Figure 4.12.	SNR of articulatory gestures.	126
Figure 4.13.	SNR_g under hand gesture interference	127
Figure 4.14.	Real-world Usage WER.	128
Figure 5.1.	EgoADL is an egocentric ADL sensing system, leveraging an on-body smartphone as a sensor hub to capture the audio, Wi-Fi CSI, and motion sensor signals simultaneously.	131
Figure 5.2.	EgoADL data in the time domain, including Wi-Fi CSI, audio, accelerometer signals, ground truth labels and egocentric video from a head-mounted GoPro for ground truth labeling.	138
Figure 5.3.	EgoADL demographics of participants.	139
Figure 5.4.	Implementation pipeline of the EgoADL preprocessing and labeling.	140
Figure 5.5.	EgoADL labeling tool	141
Figure 5.6.	State-based human behaviors in EgoADL dataset.	142
Figure 5.7.	Event-based human behaviors in EgoADL dataset.	143
Figure 5.8.	Egocentric v.s. Device-free Sensing SNR for audio and Wi-Fi.	144
Figure 5.9.	Egocentric SINR for the same benchmark sound event and human activity as in Fig. 5.8.	145
Figure 5.10.	EgoADL dataset labels word cloud.	146
Figure 5.11.	Venn diagram visualizes the advantages of each modality.	147
Figure 5.12.	EgoADL Multi-Modal Frame-Wise Slow-Fast (MMFWSF) transformer design with modality-specific encoders and a transformer-based seq-2-seq model for translating sensory feature into natural language.	149
Figure 5.13.	EgoADL SSL methods.	155
Figure 5.14.	Representative label refinement.	158
Figure 5.15.	BERT-based EgoLM design, which learn the contextual information.	159

Figure 5.16. EgoADL classwise mAP 164

Figure 5.17. EgoADL classwise mAP with label refinement 165

Figure 5.18. Venn diagram visualizes the advantages and limitations of EgoADL multi-modal fusion. Actions/ objects with > 80% top-1 mAP are in the circle for different modality fusions. 166

Figure 5.19. Overlapped ADL labels between EgoADL and egocentric vision 170

Figure 6.1. Ambient Intelligence Vision 179

LIST OF TABLES

Table 2.1.	Power consumption (mW) in controlled settings.	39
Table 2.2.	Percentage of words in CMU Pronouncing Dictionary [17] that have similar number of consecutive phoneme sequence as different wake words.....	42
Table 3.1.	StealthyIMU potential attacking permissions and VUI response examples. .	54
Table 3.2.	Types of Voice Command and Dataset Scale.....	71
Table 3.3.	Impacts of Training Datasets.....	75
Table 3.4.	VUI response identification ablation study.	75
Table 3.5.	VUI Response Private Entity Recognition Ablation Study.	76
Table 3.6.	SLU model training approach.	77
Table 3.7.	One-time stealing results.	79
Table 3.8.	Navigation voice for different cities.	82
Table 3.9.	Generalization across different smartphones and sampling rate	82
Table 3.10.	Voice detection and segmentation overhead.	85
Table 3.11.	On-device DNN overhead. ID: VUI response identification model; SLU: VUI response private entity recognition model.	85
Table 3.12.	Defense Speech Quality Subjective Assessment	87
Table 4.1.	Layers comprising ultrasound subnetwork.	107
Table 4.2.	Layers comprising speech subnetwork (BLSTM and FC layers parameters are the same as the ultrasound subnetwork.).	108
Table 4.3.	Layers comprising T domain phase network. Kernel size = 32, Stride = 2, Padding = 15.....	113
Table 4.4.	UltraSE micro benchmark.	120
Table 4.5.	UltraSE ablation study.	124
Table 4.6.	Inference time for processing 5 s speech.	124
Table 5.1.	Representative ADL systems using Wi-Fi CSI, IMU and Audio.	135

Table 5.2.	EgoADL CNN-based encoders parameters (C: Channel).	151
Table 5.3.	EgoADL micro benchmark.	163
Table 5.4.	Accuracy of EgoADL SSL.	167
Table 5.5.	Generalization of EgoADL SSL.	168
Table 5.6.	Extensibility of EgoADL SSL.	168
Table 5.7.	Comparison between EgoADL and egocentric vision.	169
Table 5.8.	Performance gain due to EgoLM. “ADL”, “A”, “O”, “S”, “E” represent to “Overall ADL”, “Action”, “Object”, “State” and “Event”, respectively. . . .	171

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to everyone who has supported me throughout my Ph.D. journey. Without the encouragement, guidance, and understanding of so many, this dissertation would not have been possible.

I am profoundly grateful to my advisor, Professor Xinyu Zhang, for his unwavering support, insightful guidance, and encouragement. His expertise, patience, and constructive feedback have not only shaped this dissertation but also significantly contributed to my growth as a researcher. Professor Zhang always encouraged me to pursue the most impactful research problems that aligned with my interests, and our intensive discussions helped me refine ideas and tackle complex challenges. He also provided invaluable assistance in shaping my career path, supporting me wholeheartedly in achieving my dream of becoming a professor. Without his unwavering support, I would not have been able to secure my tenure-track assistant professor position at the EECS department of the University of Michigan, Ann Arbor. I could not have asked for a better advisor.

I would also like to extend my heartfelt thanks to my Ph.D. committee members: Professor Rajesh K. Gupta, Professor Patrick Pannuto, Professor Tauhidur Rahman, and Professor Edward Wang. Their invaluable time, feedback, and insights have played a pivotal role in refining my research and broadening my understanding of key concepts in my field. Their collective expertise and thoughtful advice have been instrumental in improving this dissertation.

To my academic collaborators, I extend my deepest gratitude for their partnership and shared efforts in advancing our research together. This includes Professor Farinaz Koushanfar, Professor Kun Qian, Professor Lu Su, Professor Wei Wang, and Professor Chris Xiaoxuan Lu, as well as graduate students Chunyu Xia, Jung-woo Chang, Chen Chen, Baicheng Chen, Xingyu Chen, Songlin Xu, and Zihao Feng. Our collaborative discussions sparked insightful ideas that greatly influenced the direction of our research. Their contributions have made our joint work rewarding and enriching, and I have learned a great deal from each of them.

I would also like to acknowledge my mentors and managers from the industry, whose

real-world perspectives, mentorship, and encouragement have shaped my approach to practical problem-solving and enriched my professional development. Special thanks go to Doctor Berkant Tacer, Doctor Ravi Pulugurtha, Doctor Nikhil Shankar, Krishna Kamath, Doctor Renato Nakagawa, Doctor Rong Hu, and Doctor Karthik Kumar from Amazon Lab 126; Doctor Hao Chen and Doctor Charlie Zhang from Samsung Research America; and Doctor Lindsey Sunden from Google Research. Their mentorship has had a profound impact on both my research and my career development.

I am incredibly thankful to my friends, both near and far, for their unwavering support, encouragement, and understanding throughout this journey. I would like to specifically mention Renjie Zhao, Jiayou Guo, Yi Xu, Song Wang, Jingqi Huang, Wenyu Peng, and many others who have stood by me through both the highs and lows. Their friendship has been a source of strength, and their camaraderie has enriched my experience during my Ph.D.

Lastly, I owe my deepest gratitude to my family. To my parents, Lingzhu Huang and Wenjie Sun, for their unconditional love and sacrifices. Because of the pandemic, I haven't seen them in the last five years, yet their constant presence and reassuring words have given me the strength and determination to persevere. To my wife, Haiwei Yong, for her constant support, patience, and belief in me—her love has been a source of energy and encouragement during even the most challenging times. She has been my pillar of strength, helping me when I felt exhausted and bringing joy and balance to my life.

To everyone mentioned above, and to all those whose names I may have inadvertently omitted, please accept my sincerest thanks for your contributions and support. This dissertation would not have been possible without your assistance and encouragement.

Chapter 2 contains material from “Alexa, Stop Spying on Me: Speech Privacy Protection Against Voice Assistants”, by Ke Sun, Chen Chen, and Xinyu Zhang, which appears in the 18th ACM Conference on Embedded Networked Sensor Systems (SenSys), 2020 [211]. The dissertation author was the primary investigator and author of this paper.

Chapter 3 contains material from “StealthyIMU: Extracting Permission-protected Private

Information from Smartphone Voice Assistant using Zero-Permission Sensors”, by Ke Sun, Chunyu Xia, Songlin Xu, and Xinyu Zhang, which appears in the 30th edition of the Network and Distributed System Security Symposium, 2023 [213]. The dissertation author was the primary investigator and author of this paper.

Chapter 4 contains material from “UltraSE: Single-Channel Speech Enhancement Using Ultrasound”, by Ke Sun, and Xinyu Zhang, which appears in the 30th annual International Conference on Mobile Computing and Networking (MobiCom), 2021 [215]. The dissertation author was the primary investigator and author of this paper.

Chapter 5 contains material from “Multimodal Daily-life Logging in Free-living Environments Using Non-Visual Egocentric Sensors on a Smartphone”, by Ke Sun, Chuyu Xia, Xinyu Zhang, Hao Chen, and Charlie Zhang, which appears in the Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), Volume 8, Issue 1, 2024 [214]. The dissertation author was the primary investigator and author of this paper.

VITA

- 2012–2016 Bachelor of Computer Science and Technology, Department of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
- 2016–2019 Master of Computer Science, Department of Computer Science and Technology
Nanjing University
- 2019–2024 Doctor of Philosophy, Department of Computer Science and Engineering
University of California San Diego

ABSTRACT OF THE DISSERTATION

Repurposing the Ubiquitous Acoustic Devices for Cross-Modality Sensing

by

Ke Sun

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California San Diego, 2024

Professor Xinyu Zhang, Chair

Ubiquitous acoustic sensors and actuators, *i.e.*, microphones and loudspeakers, are among the most common components in consumer electronic devices. Traditionally, these components have been primarily used for sound-related tasks, including voice-user interfaces, sound playback, and sound event detection. However, with the growing demand for consumer electronics to deliver more intelligent, cost-effective, human-centric, and trustworthy ambient intelligence while reducing costs, energy consumption, and computational overhead, there is an increasing need to unlock the full potential of these components for cross-modality sensing applications.

The dissertation argues that, by designing advanced signal processing techniques and deep learning models, I am able to repurpose acoustic components for cross-modality sensing

applications, achieving resolutions comparable to dedicated sensors. To support this, I develop end-to-end systems that encompass hardware design, sensor placement optimization, advanced signal processing, deep neural network architectures, and system-level optimizations. By leveraging a thorough understanding of the strengths and limitations of acoustic sensors and actuators, this dissertation reveals novel functionalities in ubiquitous acoustic devices, including speech privacy protection, speech enhancement, and monitoring of daily life and health.

Chapter 1

Introduction

Acoustic signals are fundamental to human communication and interaction, serving as the backbone of our daily lives. From spoken language to environmental sounds, these signals provide a continuous stream of information that helps us interpret and navigate the world around us. On average, humans are exposed to between 20,000 and 30,000 words each day, primarily through speech, underscoring the omnipresence of acoustic communication. Beyond interpersonal interactions, acoustic signals hold immense potential in various domains, including environmental monitoring, healthcare, and human-computer interaction. The wealth of information embedded in sound waves can be harnessed to assess the environment, track physiological states, and enhance user experiences through advanced technologies.

Acoustic devices, which process these signals, have become ubiquitous in modern life. As of 2023, over 6.8 billion people worldwide use smartphones, most of which are equipped with high-performance microphones and speakers. Additionally, smart speakers, wireless earbuds, and other acoustic-enabled devices are experiencing rapid growth, with the global smart speaker market projected to exceed 500 million units by 2026. The pervasiveness of these devices is evident as individuals frequently engage with multiple acoustic platforms daily, whether for communication, virtual assistant interactions, or media consumption. This widespread availability of acoustic devices presents significant opportunities for repurposing them beyond their conventional functions. This dissertation explores how these ubiquitous acoustic devices,

when paired with the right algorithms and models, can be transformed into powerful sensors for tasks beyond traditional audio processing, particularly for cross-modality sensing.

1.1 Opportunities for Repurposing Acoustic Sensors and Actuators in Ubiquitous Devices

Acoustic sensors and actuators (*i.e.*, microphones and loudspeakers) are widely integrated into a broad array of mobile, wearable, and IoT devices. Traditionally, their primary roles include supporting voice-user interfaces, sound playback, and sound event detection. However, this dissertation posits that these acoustic components can be repurposed for cross-modality sensing applications—such as speech privacy leakage protection, speech processing, and mobile health—by employing cross-modality signal processing and deep learning models. This approach harnesses a comprehensive understanding of the strengths and limitations of various sensors to reveal novel and unexpected functionalities of acoustic sensors and actuators. In this section, I explore several key opportunities that illustrate the potential of these repurposed technologies.

1.1.1 Repurposing Loudspeakers and Microphones for Ultrasonic Sonar

Loudspeakers and microphones in ubiquitous devices are primarily designed for audible sound. However, since these components typically operate at a sampling rate of 48 kHz, they are capable of transmitting and receiving inaudible acoustic signals with frequencies above the audible range, specifically higher than 20 kHz. This characteristic enables their repurposing for ultrasonic sonar applications.

In my previous research, this principle has been utilized to transform the loudspeakers and microphones in consumer devices into ultrasonic sonar systems, enabling the detection of distance and movement between two acoustic devices, or between objects and the acoustic devices. The core concept involves using loudspeakers to transmit ultrasonic signals (above 20 kHz), while the microphones capture the reflected signals from various propagation paths. By applying software-based solutions, these devices can extract key information from the

propagation paths, such as time-of-flight (ToF), time difference of arrival (TDoA), and phase changes, *etc.* This enables precise measurement of distance, object positioning, and real-time tracking of distance changes. This principle has been applied in both device-based and device-free tracking applications. For instance, it has been used for device-based tracking and localization, such as tracking smartphones [180, 148] and drones [150]. Additionally, it has enabled device-free tracking, where objects without embedded acoustic sensors can be monitored, as seen in hand gesture tracking [248, 250, 245] and vital sign monitoring [183, 243]. In this dissertation, I extend this principle in a novel way by utilizing a single pair of loudspeakers and microphones for multi-modal sensing. Specifically, I design a single-channel speech enhancement and separation solution on smartphones, as detailed in Sec. 4, leveraging ultrasonic sonar techniques to enhance speech processing capabilities.

1.1.2 Repurposing Actuators for Inaudible Sound Generation

Recent research has demonstrated that various actuators can generate imperceptible acoustic vibrations, which are detectable by microphones but remain inaudible to humans. This principle has raised significant security concerns, as it enables the injection of “*inaudible voice commands*” into voice assistants, potentially leading to unauthorized device control. For example, loudspeakers can transmit ultrasonic signals that, while inaudible to humans, can be interpreted as audible commands by microphones [281, 191]. Additionally, “inaudible voice commands” can be generated through the photoacoustic effect, as shown in [210].

In contrast, this dissertation leverages such inaudible sound generation to seamlessly jam the microphone to achieve automatic speech privacy protection, which is discussed in Sec. 2.

1.1.3 Transforming Various Sensors into Side-channel Microphones

The pervasive use of speech communication devices amplifies the potential threats of *speech eavesdropping*, *i.e.*, secretly listening to private speech without the speaker’s consent or awareness. Due to the emergence of smart devices, speech eavesdropping can occur in broader

scenarios than overhearing through walls. Recent research indicates that non-acoustic hardware, such as motion sensors [153, 32, 34, 39, 209, 80], actuators [151, 192, 138], communication devices [255, 240, 271, 133, 254], storage devices [125], radars [52, 75, 268, 238, 288, 279, 54], cameras [63, 141], and instruments [169, 168, 286, 195], can create a side channel to eavesdrop the speech generated by humans or loudspeakers.

While these devices are not intended for sound recording, they can capture the unintended acoustic byproducts of speech, potentially enabling the recovery of private conversations. In this dissertation, I introduce a new attack model that leverages side-channel eavesdropping to intercept responses from voice user interfaces of voice assistants. Since voice assistants often serve as “Oracle” apps for mobile and IoT devices, my research shows that side-channel sensors can bypass permission protection mechanisms and steal sensitive information from the voice assistant’s responses, as discussed in Sec. 3.

1.1.4 Repurposing Microphones for Daily-Life Sound Logging

Microphones are traditionally used not only for speech capture but also to detect ambient sound events in the environment. Typically, external microphone arrays, such as those integrated into voice assistants, are employed to capture these sounds, which are then classified into a few pre-defined categories such as household noises, outdoor activities, or social interactions [126, 122, 82]. This form of sound event detection has proven useful for specific applications by identifying distinct sound patterns can trigger specific actions.

However, current sound event detection systems rarely integrate with wearable smart devices, such as smartphones, smartwatches, or earbuds, and, importantly, are not yet deeply connected with the human daily life context. Wearable devices, which are increasingly embedded into the user’s everyday environment, offer an untapped potential for continuous sound logging at specific on-body location. By leveraging the microphones on these devices, it becomes possible to capture and analyze a wide range of sounds associated with daily human activities, such as footsteps, door movements, or interactions with objects.

In this dissertation, I explore the potential of repurposing microphones in wearable devices to fuse with other existing sensors for continuous, unobtrusive sound logging in daily life (see Sec. 5). This extends the role of microphones beyond simple speech and ambient sound detection to a richer, more nuanced understanding of a person’s daily auditory landscape, contributing to applications such as personal activity tracking, health monitoring, and context-aware computing. This integration of daily-life sound logging into wearable devices not only enhances user experience but also opens new possibilities for ambient intelligence systems that are deeply intertwined with human life.

1.2 Dissertation Contributions

This dissertation investigates the potential of repurposing ubiquitous acoustic devices for innovative IoT applications, including speech privacy leakage and protection, speech processing, and healthcare monitoring. The core innovation lies in leveraging the existing sensors and actuators embedded in these devices, without requiring significant hardware modifications, to enable cross-modality sensing. This approach carefully balances enhanced sensing capabilities with the need for privacy protection. To achieve this, the dissertation explores the unique advantages of various sensors within these ubiquitous acoustic devices, identifies novel opportunities for cross-modal sensing, designs both hardware and software solutions to realize these functionalities, proposes advanced signal processing techniques and neural network models to address the challenges of multi-modal and cross-modal data fusion.

The specific contributions of this dissertation are:

- In Chapter 2, I design MicShield, the first system designed as a companion device to automatically protect speech privacy from always-on microphones. MicShield introduces a novel selective jamming mechanism that obfuscates the user’s private speech while allowing legitimate voice commands to pass through to voice assistants. The system proposes an innovative speech processing pipeline that utilizes framewise likelihood to

detect the onset of wake words, enabling precise selective jamming. To minimize user disruption, MicShield generates inaudible jamming signals that can still be captured by conventional microphones due to a well-known non-linear aliasing effect. To counteract potential defense mechanisms, such as microphone arrays, MicShield employs acoustic waveguides to direct jamming signals toward each microphone while preventing self-interference. I prototype a fully offline version of MicShield using low-cost off-the-shelf components and validate its effectiveness in protecting speech privacy without impairing the functionality of voice assistants. With MicShield, both automatic speech recognition (ASR) systems and human listeners can recognize less than 0.1% of words in private speech, while achieving nearly the same wake word response rate and speech signal-to-noise ratio (SNR) during normal voice assistant use.

- In Chapter 3, I propose StealthyIMU, a new threat that uses motion sensors to steal permission-protected private information from the Voice User Interfaces (VUIs). By exploiting the deterministic patterns inherent in VUI responses, I formulate the StealthyIMU attack as an spoken language understanding problem. I develop a sequence-to-sequence deep neural network (DNN) model with a cross-modality knowledge distillation strategy, enabling the direct extraction of private entities from motion sensor data. Our experiments demonstrate that StealthyIMU can accurately extract private information from 23 commonly used voice commands, including contacts, search history, calendar details, home address, and even GPS traces. To mitigate this threat, I propose a speech pre-distortion mechanism that defends against StealthyIMU while maintaining the natural quality of VUI speech output.
- In Chapter 4, I propose UltraSE, which aims to tackle the holy grail of audio processing, *i.e.*, single-channel speech separation and enhancement(SSE). UltraSE uses ultrasound sensing as a complementary modality to separate the desired speaker’s voice from interferences and noise. UltraSE uses a commodity mobile device (*e.g.*, smartphone) to

emit ultrasound and capture the reflections from the speaker's articulatory gestures. It introduces a multi-modal, multi-domain deep learning framework to fuse the ultrasonic Doppler features and the audible speech spectrogram. Furthermore, it employs an adversarially trained discriminator, based on a cross-modal similarity measurement network, to learn the correlation between the two heterogeneous feature modalities. Our evaluation results show that UltraSE can separate the targeted speech in a sophisticated environment with multiple speakers and ambient noise, improving SNR by 10.65 to 17.25 dB. UltraSE achieves an SNR gain of 6.04 dB on average over state-of-the-art single-channel speech enhancement methods, across various interference/noise settings. Its performance gain is even comparable to multi-channel (audio-visual) solutions.

- In Chapter 5, I introduce EgoADL, the first egocentric human activities of daily living (ADL) sensing system that uses an in-pocket smartphone as a multi-modal sensor hub to capture body motion, interactions with the physical environment and daily objects using non-visual sensors (audio, wireless sensing, and motion sensors). I developed a platform for EgoADL sensor data collection and labeling, establishing a comprehensive dataset for multi-modal egocentric ADL sensing through non-visual sensors. The platform has enabled the collection of 20 hours of labeled data and over 100 hours of unlabeled data, covering 221 distinct ADLs involving 70 actions and 91 objects. This data was gathered from 30 users across 20 different home environments, encompassing unrestricted activities, including ambulatory motion and interactions with daily objects. To process this data, I designed multi-modal frame-wise slow-fast encoders to learn feature representations that capture the complementary strengths of various sensing modalities. Additionally, I adapted a transformer-based sequence-to-sequence model to decode time-series sensor signals into sequences of words representing ADLs. To address the challenge of limited labeled data and enhance generalization, I introduced a self-supervised learning framework that extracts intrinsic supervisory signals from multi-modal sensor data. Our experiments in free-living

environments demonstrate that EgoADL achieves performance comparable to video-based approaches, advancing the vision of ambient intelligence in real-world settings.

Chapter 2

Automatic Speech Privacy Protection Against Voice Assistants

2.1 Introduction

Voice assistants (VAs), *e.g.*, Amazon Echo [31] and Google Home [85], can enable hands-free interactions between human and smart computing devices. They have become a mainstream user interface in the smart home ecosystem, and are widely adopted by emerging mobile devices, *e.g.*, wearable earbuds and virtual reality headsets. Market analysis reveals an installation base of more than 76 million [208], and 21% of US adults have a VA device in their homes [260].

Despite the surging popularity and the alluring ability to automate human life, VAs are sparking an outcry of privacy concerns. Officially, vendors advocate that these devices are programmed to record and send information to the cloud for processing only when they are activated by a wake word/ phrase, *e.g.*, “Alexa!” or “OK Google!” [29, 87]. However, there exists no easy way for users to trust and enforce such behavior. In fact, there are numerous cases when certain VA devices record private speech without user’s knowledge or consent. For instance, due to firmware bugs, an early version of Google Home Mini [86] kept recording users for 24/7 even without the wake word, and uploaded all the records to cloud servers [36]. Amazon reportedly hires human workers to transcribe recordings from their Echo devices, in the name of improving device performance and consumer experience. However, it has been found that

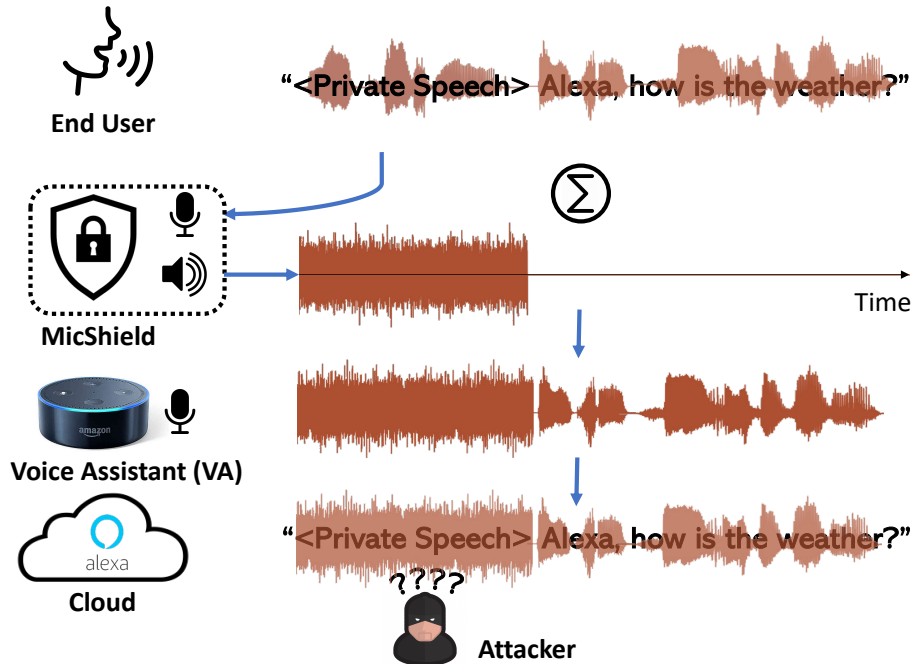



Figure 2.1. MicShield acts as a companion device with the VAs. The transducer of MicShield would start/stop emitting inaudible jamming signal based on the wake word. The unintended private speech before wake word is thus obfuscated before reaching the VA, which prevents attackers from eavesdropping the private conversations.

majority of the transcribed clips were uneventful or not preceded by wake words. Users have no control on their unintended audio records once being sent to remote cloud [275]. Besides, it has long been a concern that certain government agencies may take advantage of the VAs as a means of pervasive surveillance [263].

In this paper, we seek to answer the question: *can we force the VAs to record the legitimate commands only, rather than private speech?* A straightforward way to protect speech privacy is to disable the VA through a physical mute button  [118], and unmute only when the user needs to issue a command. However, this compromises the very first advantage of VAs, *i.e.*, convenient hands-free interactions. Besides, once the VA is compromised, the mute button is not trustable any more. Alternatively, one can adopt anti-eavesdropping approaches, which generate an interfering signal to jam the VA's microphone [193, 55]. But this makes the VA deaf and irresponsive to any activation commands.

We propose *MicShield*, a companion device which, for the first time, prevents VAs from recording private speech without affecting the VAs' normal functionalities. Figure 2.1 illustrates the basic threat model and defending mechanism of MicShield. Assuming the VA is untrustable, MicShield continuously emits a jamming signal to deafen the VA. Meanwhile, it keeps listening and stops jamming immediately when it senses the onset of a wake word. MicShield works offline, keeping all processing local and shredding voice data immediately afterwards. So it is not subject to the privacy risks from the always-listening, cloud-operated VAs.

To protect speech privacy without disturbing the VAs' daily usage, we must address two challenges. First, a straightforward way of continuous jamming will suppress not only private speech, but the wake word. This will cause VAs to become unresponsive to the subsequent voice commands. To overcome the dilemma, we leverage the fact that the wake word follows a fixed phoneme pattern, and even with the first few milliseconds jammed, the wake word can still be identified by Automatic Speech Recognition (ASR) algorithms. MicShield thus dynamically switches on/off jamming based on the likelihood of a wake word onset (*e.g.*, the initial few milliseconds of "Alexa!"). Once the likelihood exceeds a predefined threshold, MicShield suspends jamming to ensure the VA can hear majority of the wake word and the following voice commands. To avoid disturbing users, MicShield generates inaudible jamming signals, which however can be captured by regular microphones due to a well known non-linear aliasing effect [191].

The second challenge is to defeat the potential countermeasures based on microphone arrays, which exist in majority of the Off-The-Shelf (OTS) VAs. Whereas the microphone arrays have been used for sound localization [93], they can enhance the user's voice through acoustic beamforming, thus mitigating the effectiveness of MicShield's jamming. MicShield thwarts such potential countermeasures through a gain suppression method, which saturates the microphones and fully obfuscates the private speech. Our design employs acoustic waveguides to redirect the jamming signal towards each microphone. Meanwhile, these waveguides avoid the self-interference setbacks and ensure MicShield itself can still identify the wake words amid the

jamming signal.

We built a prototype of MicShield with OTS components and a 3D printed shield. In accordance with the privacy protection assumption, we optimize both the computation and energy consumption to make the whole MicShield system work offline. Our experimental results demonstrate the effectiveness of MicShield in protecting private speech and countering potential threats. With MicShield, both ASR algorithms and human perception can only recognize less than 0.1% of the words in private speech. Besides, MicShield does not break the functionalities of VAs. It achieves nearly the same wake word response rate and speech Signal-to-Noise-Ratio (SNR). Furthermore, we show the MicShield’s generalization across various wake words and VA devices provided by different vendors.

The main contributions of MicShield are as follows.

- We introduce a new concept to automatically protect speech privacy against always-on microphones by selectively jamming unintended private speech while passing intended voice command.
- We propose a novel speech processing pipeline which leverages the framewise likelihood to detect the onset of a wake words, thus realizing selective jamming.
- We propose a method to jam an entire microphone array using a single speaker, while avoiding self-interference.
- We prototype purely offline MicShield through low-cost OTS components, and validate its effectiveness in speech privacy protection without affecting the VAs’ functionalities.

2.2 Related Works

2.2.1 Hidden Command Attacks and Defenses

Inaudible Voice Attacks and Defenses: Inaudible voice attacks [191, 281, 193, 210] use ultrasound or laser to generate imperceptible acoustic vibration signals that can be captured by

microphones. BackDoor [191] shows that ultrasound can be recorded by traditional microphones in spite of the built-in low-pass filters. This is enabled by the hardware non-linearity of microphones, which creates aliased version of signals in the low frequency band. DolphinAttack [281] leverages a similar effect to attack the VAs through inaudible voice commands. [193] extended the attacking range from 5 ft to 25 ft, by using multiple ultrasonic speakers. A defending method was further proposed [193] to discriminate such attacks from speech signal, by identifying the spectrogram traces left by the non-linear effects. He *et al.* [102] further introduced a way to suppress the inaudible voice commands by using an ultrasonic transducer and interference cancellation methods. Inaudible voice command [210] can also be generated through the photoacoustic effect. By modulating the laser beam targeting a MEMS microphone, acoustic commands can be fabricated even at a distance of 110 m. To avoid disturbing users, MicShield uses inaudible voice as the jamming signal. Whereas previous work [191, 281, 193, 210, 102] investigated the attacks and defending approaches against *inaudible voice commands*, we focus on protecting *speech privacy* without handicapping the VAs.

Black-box Attacks and Defenses: Black-box attacks [46, 233, 277, 47] generate adversarial audio samples that can be interpreted by VAs, but are unintelligible to human. For example, [46, 233, 47] generate white-noise-like malicious voice commands, which however can be interpreted as legitimate by VAs. These attacks can target different ASR algorithms including statistical learning models (*e.g.*, CMUSphinx [12]) and deep learning approaches. CommanderSong [277] embeds the voice command into songs to attack VAs. As for defense, certain post processing methods, *e.g.*, audio turbulence and audio squeezing, can single out the adversarial examples. On the other hand, Kumar and Zhang *et al.* [124, 284] investigated the interpretation errors made by ASR algorithms. The key observation is that different words may share similar phonemes which confuse ASR algorithms. An attacker leverages such systematic errors to unconsciously redirect users to malicious applications.

2.2.2 Audio Privacy Leakage and Protection

The always-on microphones on ubiquitous VA devices are imposing a looming threat to speech privacy. By penetrating the VA or the associated cloud, attackers can extract the semantic contents, and thus the personally identifiable information (PII), from the voice records.

Protection by Obfuscating the Audio Signal Waveform: One potential solution to audio privacy leakage is to obfuscate the unintended speech. For example, Tung *et al.* [229] proposed to protect private phone conversations against eavesdropping malware. They mix the private speech with mask signals which are known to a trusted server but unknown to eavesdroppers. The server can eliminate the mask through self-interference cancellations. However, the masking sound needs to be shared in advance as a known audible secret key, which disturbs practical usability. Besides, a third trustable key generating server is needed, and it only uses symmetric key encryption. Therefore, the security guarantee breaks once the shared secrets and/or the key generator server are compromised. Chen *et al.* [55] designed a wearable jamming device using multiple ultrasonic transducers to protect speech privacy. This always-on jamming device make all nearby VAs malfunction and deaf to legitimate voice commands. In contrast, MicShield adopts a simple full-duplex phoneme-level selective jamming mechanism which ensures the wake words and voice commands are audible to the VAs.

Network Level Protection: An alternative protection mechanism is to identify and prevent the privacy leakage through transport layer packet filtering. Prior works, such as PrivacyProxy [207], ProtectMyPrivacy [25] and Meddle [186], introduced VPN proxies that detect the leakage of audio data packets by intercepting the outbound network traffic. Yet these systems cannot discriminate legitimate voice commands from unintended private speech. VoiceMask [181], on the other hand, used an intermediary between VAs and cloud to anonymize speech data. But the unintended semantic content can still be exploited from the private speech. Additionally, these mechanisms all rely on interception and interpretation of TCP/HTTP data. Yet all modern mobile devices are shifting toward encrypted SSL/TLS connections to prevent man-in-the-middle

attacks [57, 2]. Thus, designing a VPN proxy to intercept, and thus control encrypted audio privacy data over transport layer, is practically challenging.

2.3 Threat Analysis

Threat Model: MicShield targets the scenario where adversaries use VA’s always-on microphones to eavesdrop on *private speech*, *i.e.*, the voice signals that are not preceded by a wake word. To establish the threat model, we assume the adversaries are powerful enough to: (i) access the unprocessed speech signals captured by the always-on microphones; (ii) run existing post processing algorithm to enhance the sound quality; (iii) use existing ASR algorithms [43] and human perception to infer the semantic content.

Protection Goals: Under the premise of not breaking the VAs’ functionalities, MicShield aims to prevent the private speech from reaching the VAs. Consequently, the adversaries can no longer exploit semantics of private speech by compromising the VAs, sniffing the network traffics, or hacking the remote cloud. MicShield acts as a companion device to enforce speech privacy without modifying existing VAs’ hardware/software. To achieve this goal, *first*, MicShield should work entirely offline to make sure that the manufacturer of MicShield imposes no privacy threat. *Second*, MicShield should ensure that the wake words still trigger a VA, whereas users’ private speech preceding the voice command is jammed, shielded against the VA (Section 2.4). *Third*, MicShield needs to thwart powerful countermeasures that employ microphone arrays to enhance the speech while weakening the jamming (Section 2.5). We consider the worst case protection scenario: (i) The VA’s received A-Weighting Sound Pressure Level (SPL) [6] is sufficiently high, but less than 75 dBA, known as the maximum SPL in daily conversations without harming human auditory [7]; (ii) The adversary knows the exact location of the speech source, so it can maximize the speech enhancement through array beamforming.

Security Guarantees and Evaluation Metrics: Unlike traditional cryptography-based security systems that define an exact security guarantee using the estimated computational time for

breaking the system, providing similar guarantee for MicShield is challenging. However, this is also a common issue for non-cryptography systems. Inspired by the idea of Wyner wiretap model for a secure wireless system [267], we define the evaluation metrics as follows:

- **Mute Rate:** defined as the ratio between the jamming duration and the entire speech duration, excluding silent periods. Our design of jamming control policy focuses on *wake word mute rate* and *private speech mute rate*. Theoretically, an ideal design would have an 100% private speech mute rate and 0% wake word mute rate, to guarantee the speech privacy without reducing the responsiveness of the VAs. Practically, a less-than-100% private speech mute rate, caused by leakage of less importance phonemes, hardly exposes any threats to private content at word-level. Similarity, a slightly higher than 0% wake word mute rate does not necessarily fail the VAs, since the wake word recognition is imperfect even when no jamming signal exists.

- **Wake Word Misdetction Rate:** defined as the probability that a wake word cannot be correctly recognized. Our design needs to ensure a near equivalent wake word misdetction rate for the VAs, with and without MicShield jamming.

- **Jamming Effectiveness:** We quantify the jamming effectiveness by *PESQ (Perceptual Evaluation of Speech Quality)*, a metric for objective voice quality assessment commonly used by telecom operators [187], and *Speech Recognition Rate*, defined as the probability that the obfuscated speech can be correctly recognized by ASR or human perception. Theoretically, PESQ models the mean opinion score with a range between 1 (bad) to 5 (excellent) [187]. A typical acceptable range for VoIP applications lies between 3.8 and 5 [190], and PESQ less than 2 is known as the extremely low speech quality. To evaluate the speech recognition rate, we compare the ground truth and transcribed words by asking human participants to recognize the speech, and by using cloud based ASR services [43], including Amazon Transcribe [11] and Google Speech-To-Text (STT) [13].

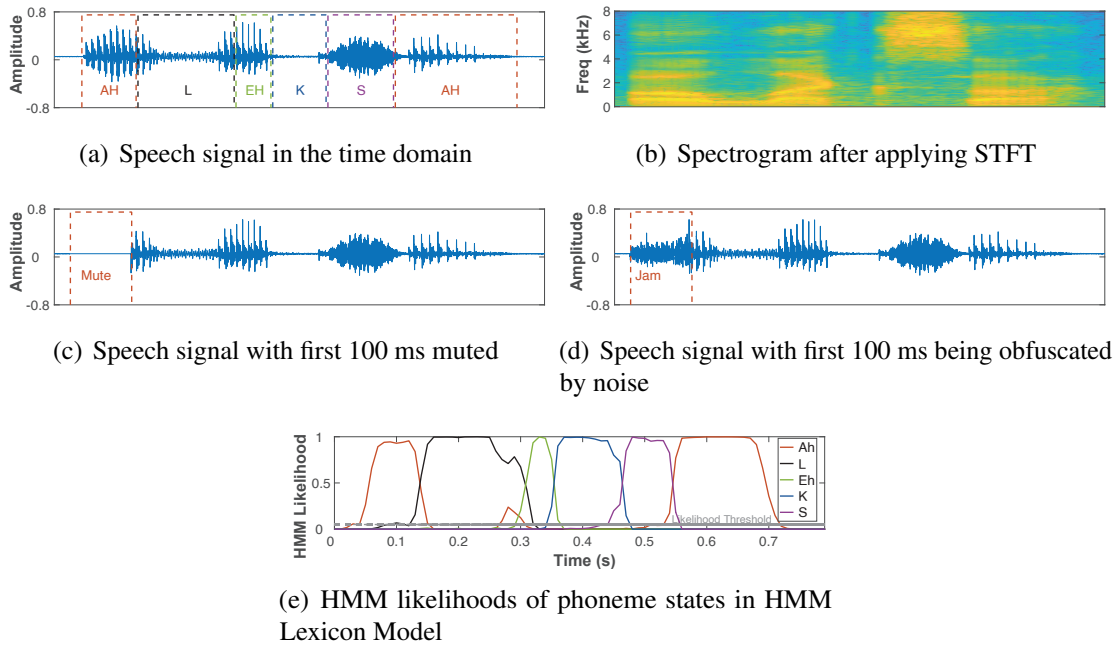


Figure 2.2. Analysis of “Alexa” speech signals. Wake word “Alexa” follows a specific sequence of phonemes and can be recognized when the first few milliseconds of speech is muted.

2.4 Automatic Jamming Control

In this section, we introduce the automatic jamming control algorithm that passes the wake words to VAs while obfuscating private speech. For ease of explanation, we use Amazon Echo Dot as the VA with “Alexa!” as the wake word [3]. Generalizations to other wake words and devices is straightforward and will be discussed in Section 2.7.4.

2.4.1 A Primer of Selective Jamming

To realize selective jamming, an intuitive way is to use a third-party “pre-wake” word detector [115]. The detector keeps jamming, and only stops when it hears a pre-wake word defined by the user. Immediately afterwards, it regenerates the real wake word “Alexa” through an inaudible channel, and passes it to the VA. However, our experiments reveal that, even a short wake word like “Alexa” takes at least an additional 500 ms. Thus, the inaudible “Alexa” will inevitably overlap with the user’s voice query which follows the pre-wake word immediately – a

conflict causing the VA to malfunction.

In contrast to word-level detection, we propose to use phoneme-level features to identify wake words from its early onset, which leads to negligible latency, thus avoiding the conflict. As an example, Figure 2.2(a) and 2.2(b) show the acoustic signal waveform of “Alexa” and the spectrogram after applying Short Time Fourier Transform (STFT), respectively. The “Alexa” follows the fixed phoneme sequence pattern, noted as /ə'leksɪə/, where the first phoneme /ə/ lasts from 43 ms to 136 ms. For the VA to be able to recognize the wake word, MicShield needs to switch off jamming after identifying the first phoneme. To verify the feasibility of such phoneme-level selective jamming, we conduct the following experiment with a dataset [35] containing 369 “Alexa” utterances from 87 users with different accents.

- **Can the VAs be activated when the initial part of the wake word is corrupted by jamming?**

We evaluate the wake word misdetection rate of Amazon Echo Dot when the first few milliseconds of the wake word is corrupted. To identify the beginning of “Alexa”, we use the CMUSphinx [12] phoneme forced alignment algorithm with resolution of 10 ms to compute the beginning time and duration of the first phoneme /ə/. Figure 2.2(c) and 2.2(d) illustrate the signal waveforms when the first few milliseconds of speech is muted and jammed, respectively. The jamming signal is a 4 kHz bandwidth white noise. Finally, the processed audio is played using a smartphone speaker.

Figure 2.3(a) shows the wake word misdetection rate versus predefined initial duration for mute and jamming cases. Surprisingly, we found that *even with the first 60 ms muted or jammed, the wake word can still activate the Echo Dot with 95% accuracy—the same as the case without mute/jamming (i.e., 0 jamming duration)*. Beyond 110 ms, misdetection rate under jamming is slightly lower than muting. This is likely because the amount of residual semantic information that can be exploited from the jammed signal is more than that of muted signals. Therefore, if MicShield stops jamming within the first 60 ms, Echo Dot is still able to recognize the wake word and the subsequent voice commands. This is also true for other common wake words and VAs, as will be discussed in Sec. 2.7.4.

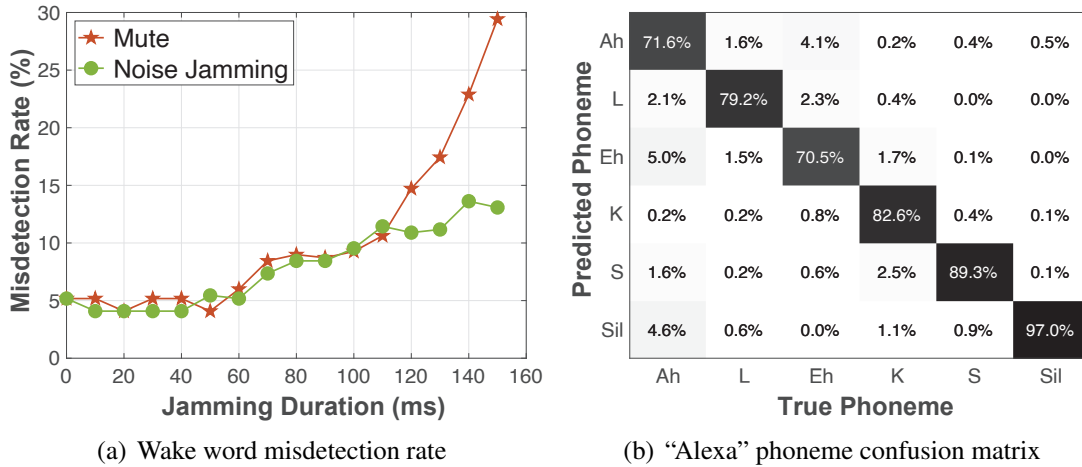


Figure 2.3. Proof-of-the-concept experiments for jamming control pipeline design.

2.4.2 Jamming Control Pipeline

Figure 2.4 elucidates MicShield’s jamming control pipeline built upon the above insights. *First*, we use framewise phoneme recognition to estimate the phoneme-level confidential scores that quantify the likelihoods of a wake word *onset* (Phase A). This enables us to control jamming with frame-level resolution. *Second*, we propose a Hidden Markov Model (HMM) based lexicon model to track the phoneme sequence (Phase B). This allows us to compute the likelihood of the wake word’s *occurrence*. *Finally*, the ultrasonic transducer would start/stop jamming based on previous two outputs (Phase C).

- Phase A: Framewise Phoneme Recognition.** Upon receiving audio stream from the analogue front end, the voice signal is *first* sampled and segmented into individual frames at 16 kHz sampling frequency, 25 ms window, and 15 ms overlap. *Second*, we apply Mel-Frequency Cepstral Coefficients (MFCC) analysis [90] with 12 coefficients and 26 filter-bank channels to each frame. Together with the first order differential coefficients, *a.k.a.* the Deltas and the log-energy, this results in a 26-dimension feature vectors for each frame. *Finally*, to identify each phoneme from the framewise feature vectors, we train a vanilla Recurrent Neural Network (RNN) with one layer containing 275 hidden units and sigmoid activation function [90]. This trained

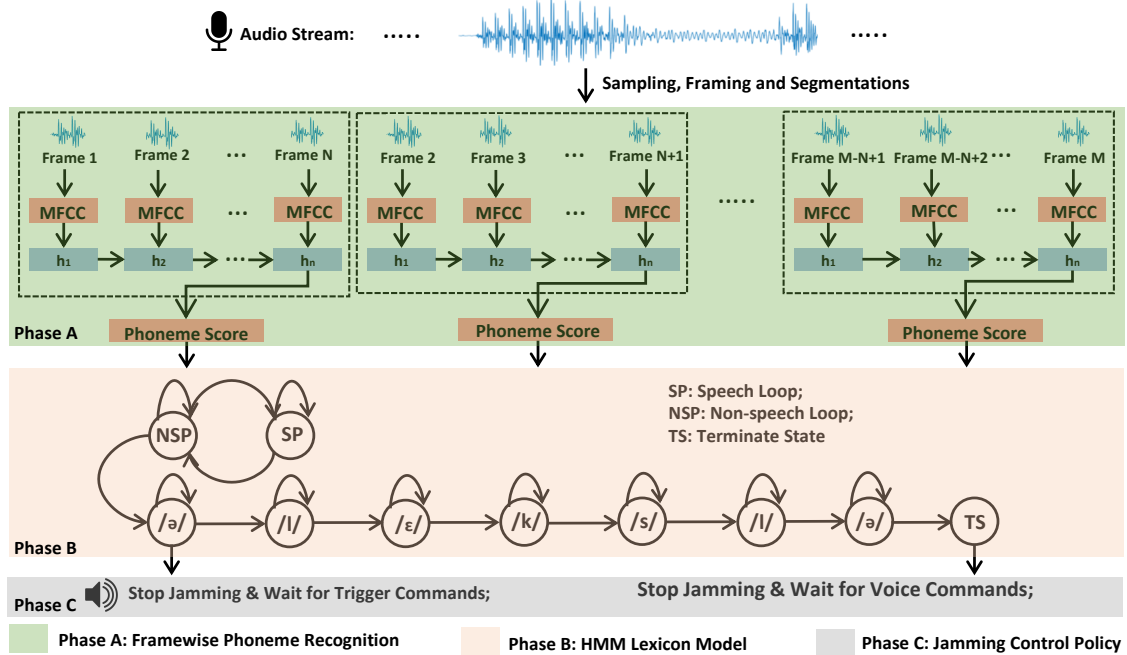


Figure 2.4. MicShield automatic jamming control pipeline.

model is then used to compute the likelihood for each possible phoneme, upon the input of each framewise feature vector. The English speech contains a limited set of 61 basic phonemes, as suggested in the TIMIT Acoustic-Phonetic Speech Corpus [81].

- Phase B: HMM Lexicon Model.** In Phase A, we use the RNN-based approach to recognize the possible phoneme from a sequence of framewise features. Following the lines of traditional ASR methods [90], we leverage an HMM-based lexicon model to specify the transitions of phoneme states, where the state space is defined by the wake word's phonemes. For “Alexa”, we define the state space as: $\{/ə/, /l/, /ε/, /k/, /s/\}$. Shown in Figure 2.4, once not following the predefined wake word sequence, the phoneme state would transition to the speech/non-speech loop. With HMM forward algorithm [184], we can compute the likelihood of each phoneme state over time, as shown in Figure 2.2(e).

- Phase C: Jamming Control Policy.** Our jamming control policy is based on the phoneme level likelihood estimations, shown in Figure 2.2(e). Once likelihood of the initial phoneme $/ə/$ exceeds the predefined threshold, MicShield will suspend jamming. However, it will

immediately switch on jamming again if the subsequent phoneme states deviate from the wake word’s predefined phoneme sequence. Otherwise, if the speech follows the predefined phoneme sequence, the HMM model will start decoding and identify the wake word. Subsequently, MicShield would stop jamming and allow the voice commands to pass through.

2.4.3 Minimizing Wake Word Misdetection

To avoid disturbing the VAs’ basic functionalities, the jamming control pipeline must minimize the wake word misdetection rate.

Phoneme Level: Towards this end, the first challenge lies in the low accuracy of framewise phoneme-level recognition method compared to the word-level recognition [90]. With the aforementioned setup, the overall accuracy can only reach 64% for 61 different phonemes with input length of 15 frames. Fortunately, unlike generic ASR models, our phoneme recognition model can be trained to be sensitive to the specific phonemes associated with the wake words, *e.g.*, /ə'leksɪə/. We thus harness this observation to enhance the detection accuracy for specific phonemes. Specifically, we fine-tune the model using a combination of the “Alexa” utterance dataset [35] and partial DARPA TIMIT Acoustic-Phonetic Speech Corpus containing phonemes in {/ə/, /l/, /ɛ/, /k/, /s/} [81]. Figure 2.3(b) shows the phoneme recognition result of /ə'leksɪə/. With the fine tuned model, the accuracy increases from 64% to 78.7%.

Lexicon Level: Besides, we further minimize the wake word misdetection rate by decreasing the HMM likelihood threshold in Phase B. As a result, even if a phoneme does not achieve the highest phoneme likelihood in Phase A, the HMM-based lexicon model will still pass the frame. However, a low threshold leads to a low mute rate for private speech, which would in turn degrade the jamming effectiveness. Therefore, *an optimal HMM likelihood threshold should consider the trade-off between wake work misdetection rate and jamming effectiveness.* We will elaborate on this design trade-off in Section 2.4.4.

Another side effect of reducing the wake word misdetection rate is the increase of *false alarm rate*. However, this will not affect the VA’s basic functionalities, since the speech signals

mistakenly identified by MicShield as wake word will be passed to and reprocessed by the VA. As long as the wake word is not identified, the VA will not be activated. In addition, although such speech signals are unprotected by MicShield, the false alarm rate is negligible (only once per 12 hours of speech as shown in Section 2.7.1), so the attacker can hardly exploit any sensitive semantic information.

2.4.4 Maximizing Private Speech Mute Rate

Our second goal is to ensure the unintended private speech is successfully obfuscated by the jamming signal, given high wake word detection accuracy as in Section 2.4.3. This requires us to maximize the private speech mute rate.

Recall that MicShield’s selective jamming mechanism may still incur some phoneme-level speech leakage. For example, when the user is saying “Agree” (/ə'gri/), MicShield first stops jamming when it hears /ə/, and then resumes jamming immediately when it hears /g/. Thus, the phoneme /ə/ is leaked. However, because the remaining phonemes of “Agree” are still jammed, such occasional phoneme leakage will not incur word-level privacy issues practically. In some extreme cases, when certain words’ phoneme sequence resembles that of the wake word, these words cannot be properly protected. We have investigated the percentage of the words that have the same phoneme subsequences as “Alexa” in the CMU Pronouncing Dictionary [17]. Only 0.01% of 134000 words share the same first 4 phonemes, and most of these words are human names, *e.g.*, “Alexei”, “Oleksy”, *etc.* Therefore, MicShield can theoretically satisfy the audio privacy protection in a majority of practical settings.

In practice, the framewise phoneme recognition model is not perfect. To reduce the wake word misdetection rate, our fine-tuned model is sensitive to the phonemes appearing in wake words (Section 2.4.3), which in turn increases the *phoneme false alarm rate*, defined as the probability when a wake word phoneme is incorrectly identified. This will in turn degrade the private speech mute rate. For instance, MicShield may confuse the first phoneme of “Alexa” (/ə'lɛksiə/) with that of “Apple” (/ˈæp əl/), and determine not to jam the first phoneme of “Apple”.

The phoneme false alarm rate in Phase A achieves an average 25.62% for those associated with the wake word (see Figure 2.3(b)). To address this issue, we use the HMM lexicon model to track the phoneme sequence pattern based on that of the expected wake word. With this approach, MicShield will immediately restart jamming once identifying unexpected phoneme sequence. Thus, *even with a relatively high phoneme-level false alarm rate, the wake word recognition can still maintain a low false alarm rate.* Combined with the method proposed in Section 2.4.3, we thus are able to ensure the expected functionalities, while protecting the unintended private conversations.

2.4.5 When to Resume Jamming?

MicShield needs to resume jamming once the VA is back to the inactive mode. There are mainly two policies for VAs to return to the inactive mode.

- It detects a sufficiently long silent period after it is triggered.
- It identifies the end of the voice command based on the semantic contents.

Practical VA devices employ voice activity detection (VAD) methods and semantic content interpretation to realize these policies. MicShield needs to realize the same policies to determine when to resume jamming.

MicShield uses VAD methods [120] as in existing VAs to realize the first policy. For example, we empirically found that the Amazon Echo and Google Home use a 7 s and 8 s VAD threshold, respectively. Other VA devices' policy can be reverse engineered in the same way. However, the second policy is challenging for MicShield to implement, as the limited computational resources on the offline devices do not allow for analyzing language semantics. To circumvent this hindrance, MicShield resumes jamming immediately when it detects that the VA begins to respond to the voice command. With this measure, it protects subsequent periods when the user starts speaking again. To differentiate between the user's voice and the VA's response, we use the well known human/speaker sound detection methods in [42, 26]. Note that although

the VAs are not trustable in our threat model, the adversarial VAs still have to respond to the user so that it can pretend to be normal. Furthermore, even the VA cheats the MicShield by intentionally not responding to the voice command, MicShield will still resume jamming after a few seconds of VAD detection.

Our scheme also supports the *follow-up mode*, where users can issue multiple requests interactively without repeating the wake word before each commands [30] by omitting the second policy, because the follow-up mode adopts the same waiting period as the first policy, *i.e.*, the VA will need to be triggered by another wake word if no voice activity is detected across the waiting period. Interrupting commands, *e.g.*, “Alexa, Stop!”, will also be identified by VA since such commands require the “Alexa” wake word even for the follow-up mode in our real-world test. Thus, MicShield will follow the same policy in Section 2.4.2 to recognize the early onset of the wake word and suspend jamming immediately.

2.5 Practical Jamming Design

2.5.1 Inaudible Jamming Sound

To avoid disturbing users, MicShield uses inaudible sound to jam the microphones. Similar mechanism has been employed recently [191, 281, 193] to send inaudible commands and hijack the VAs. Specifically, we use an ultrasonic transducer to transmit the ultrasound signals $S_{in} = \cos(2\pi f_h t)(\alpha + m(t))$, where $f_h = 40$ kHz is the high frequency carrier, and $m(t)$ is the low frequency jamming signal. Due to the microphone non-linearities, the recorded signals can be modeled as $S_{out} \approx A_1 S_{in} + A_2 S_{in}^2$. After passing the low-pass filter and DC removal, the signals received by the microphone become $S_{mic} = A_2 \alpha m(t) + \frac{A_2}{2} m(t)^2$. These signals will be picked up and thus jam the microphones.

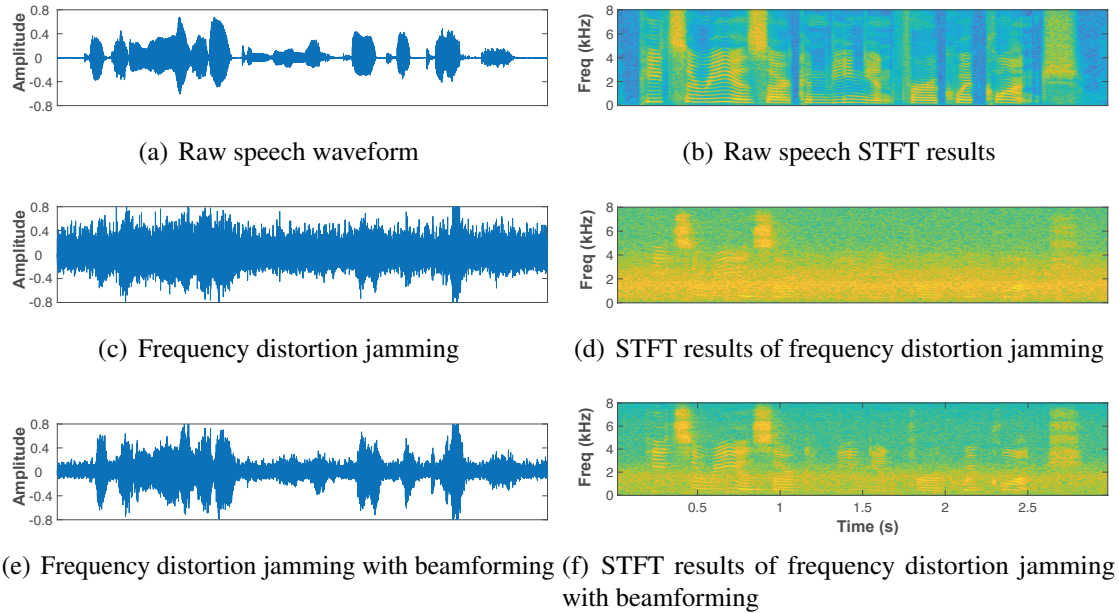


Figure 2.5. Analysis of private speech: “Pizzerias are convenient for a quick lunch.” Although frequency distortion jamming is effective to jam a single microphone ((c) and (d)), beamforming based attack can mitigate the frequency distortion jamming and enhance the private speech for attackers ((e) and (f)).

2.5.2 Jamming a Single Microphone

A single microphone can be easily jammed with traditional *frequency distortion jamming* method, which reduces the speech SNR by transmitting white/color noise. We verify the effectiveness by using a 0 ~ 4 kHz white noise as jamming signal. To control the speech SNR, we gradually increase the noise amplitude, while summing the private speech and the noise with 16-bit quantization. We reuse the TIMIT speech dataset [81] for evaluation.

Figure 2.5(c) and Figure 2.5(d) show the private speech “Pizzerias are convenient for a quick lunch.” time series waveform and STFT results from one microphone. Clearly, the low frequency components (0 ~ 4 kHz) are successfully obfuscated by noise. We further use PESQ and speech recognition rate to quantify the jamming effectiveness. Figure 2.6(a) shows that, by using frequency distortion jamming, the speech recognition rate is 1%, 0.3% and 0.9% for human perception, Amazon Transcribe [11] and Google STT [13], respectively, at -15 dB

speech SNR. Under this condition, the PESQ stays at 1.15 on average, and the maximum PESQ is less than 1.6, *i.e.*, extremely bad speech quality (see Figure 2.6(b)).

The experiment implies that *frequency distortion jamming can effectively protect the speech privacy for the single-microphone VA when the speech SNR is less than -15 dB*. To cap the SNR below -15 dB under the highest speech SPL of 75 dBA (Section 2.3), the corresponding noise SPL should be above 90 dBA. To check the feasibility of this jamming noise volume, we measure the SPL generated by a single transducer at its maximum volume, with frequency range of 10 Hz \sim 20 kHz. The SPL is sampled for each 1 cm in 7 different angles from 0° , to 90° , at a step of 15° . Figure 2.8(a) plots the resulting spatial distribution of SPL, where the contours are smoothed by 3D interpolation [68]. We see that *frequency distortion jamming achieves the required 90 dBA SPL, only when MicShield’s ultrasonic transducer is placed within 4 cm towards the microphone*.

2.5.3 Jamming a Microphone Array

Beamforming based Attack: We now investigate how an attacker can leverage a microphone array as a countermeasure to mitigate the effectiveness of the aforementioned frequency distortion jamming. Since commercial multi-microphone VAs do not allow access to raw recorded signals, we use the ReSpeaker 6-microphone array [16] for experimental purpose. The speech is transmitted by a smartphone 30 cm away from the microphone array, and the received sound has similar volume as a human user 0.5 m away from the VAs (SPL ranges from 50 to 75 dBA). The jamming noise pattern is the same as in the previous experiment, and is generated by an ultrasonic transducer fixed at 5 cm above the center of the microphone array. As discussed in Section 2.3, we consider a scenario most advantageous to the attacker, assuming it knows the exact location of the speech source. Equivalently, under the setup as in Figure 2.11(b), the attacker knows the exact time difference of arrival (TDOA) of each microphones pair on the VA. Then it can perform the classical delay-and-sum beamforming [114, 218] to enhance the sound coming from the source location.

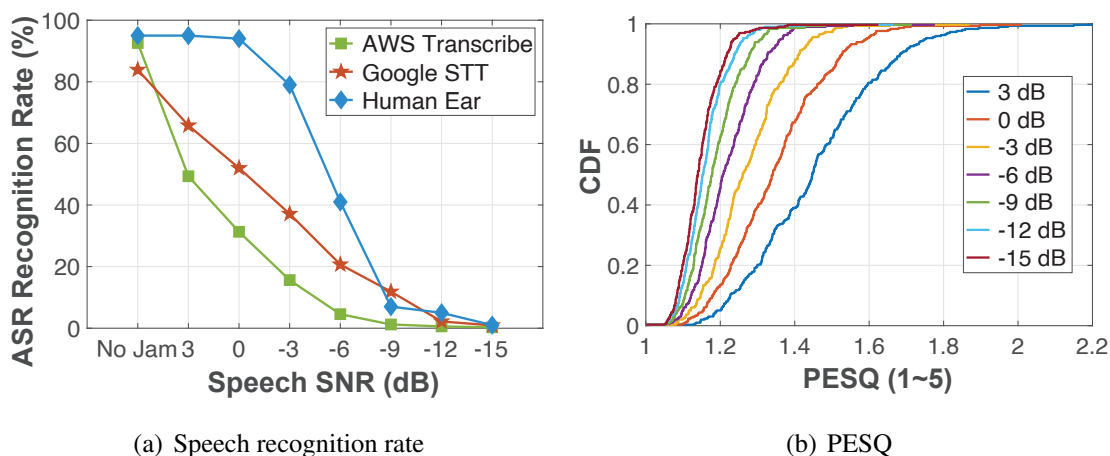


Figure 2.6. Jamming effectiveness v.s. SNR. Frequency distortion jamming can effectively protect the speech privacy for the single-microphone VA when the speech SNR is less than -15 dB.

Figure 2.5(e) and 2.5(f) show that, with the beamforming countermeasure, the speech waveform and STFT results become much closer to raw audio signals than the case without beamforming. *The 6-microphone beamforming countermeasure can enhance the speech SNR by 12 dB.* The private speech recognition rate increases significantly, *i.e.*, to 75.0%, 44.2% and 32.2% for human perception, Amazon Transcribe and Google Speech-to-text, respectively. Some of the speech corpora even achieves a PESQ higher than 2.0 (see Figure 2.7(b)).

Gain Suppression Jamming: To effectively defeat the beamforming-based countermeasure, we explore an alternative *gain suppression jamming* method. The idea is to transmit high-volume sound to saturate the microphone, *i.e.*, force the microphone to reach the Acoustic Overload Point (AOP). AOP occurs when overwhelming input sound pressure causes the microphone output to be severely distorted [258]. In practice, the gain suppression jamming needs to address 2 dilemmas.

- **(D1) Dilemma between jamming noise volume and audibility:** Ideally, a high-power jamming noise can more effectively trigger gain suppression. However, a high output volume will trigger a non-linear effect at the speaker’s diaphragm and amplifier, making the jamming

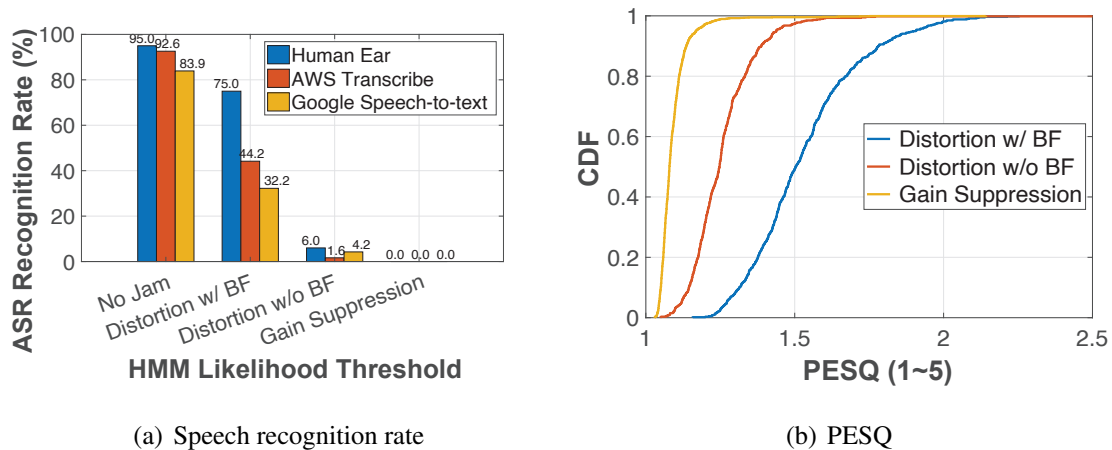


Figure 2.7. Jamming effectiveness for different jamming methods.

sound audible and disturb users [158]. To verify this phenomenon, we add a class D audio amplifier PAM8403 [4] to the ultrasonic transducer, which helps adjust the volume of jamming noise. We then ask 5 volunteers to stay 0.5m away and check if the noise is audible. With 3 W input power, no volunteer hears the transducer. But when the input power reaches 4 W, 2 volunteers can hear the sound.

To avoid the audibility while ensuring gain suppression, we thus fix the input power of the transducer to 3 W and maximize the jamming volume using a *single-frequency jamming signal* close to the resonant frequency [113], where the transducer exhibits the maximum amplitude response.

Figure 2.9 shows the waveform of private speech (see Figure 2.5(a)) jammed by the single-frequency signal. Clearly, the microphone becomes saturated, and the speech signals are clipped and distorted into square-like waveform, losing the typical frequency-domain features as well. Our measurements show that the obfuscated signals have a low PESQ of 1.09 and a 0% speech recognition rate using Amazon Transcribe [11] and Google STT [13].

- **(D2) Dilemma between jamming noise volume and coverage:** Ideally, when gain suppression jamming is applied to each microphone on a microphone array, the speech signals

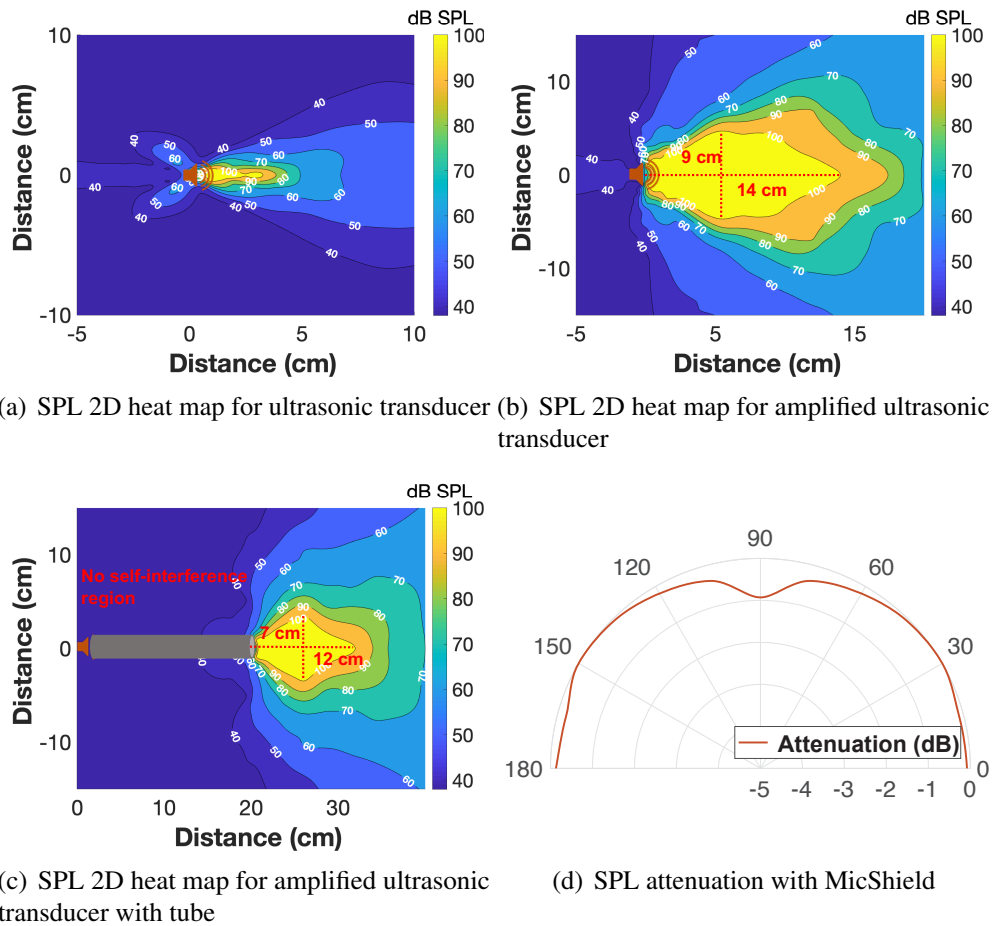


Figure 2.8. SPL heat map for ultrasonic transducer. Compared to (a) and (b), acoustic waveguide design in (c) extends the jamming space to make it possible to saturate the microphone array, and isolate the self-interference from MicShield.

will all become clipped into square-wave like waveform (see Figure 2.9) and completely unintelligible. Beamforming cannot recover the signals, as distortion occurs in the analog front-end. However, the acoustic transducer has a directional gain pattern with a narrow beam angle, whereas the multiple microphones on a VA are usually laid out to form a circular array. One could place the transducer away from the microphone array so as to expand the sound beam’s angular coverage. However, under the 3 W power constraint, it is challenging to ensure a single transducer can generate sufficiently high SPL at all the microphones.

To elucidate this dilemma, we measure the SPL of a single transducer with the 3 W

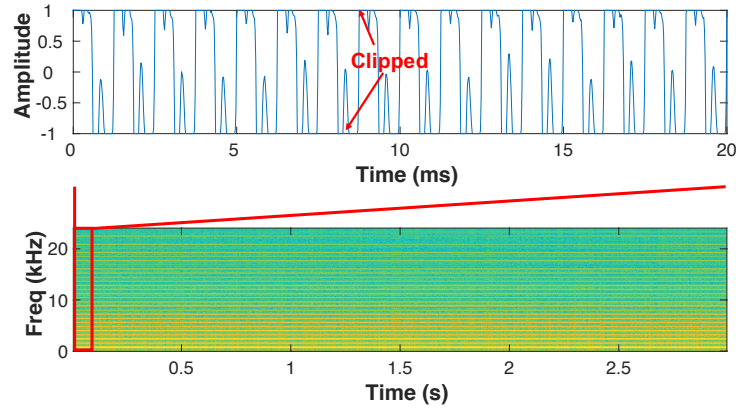


Figure 2.9. Acoustic overloading of received microphone. (a) zoomed-in saturated jammed signal in time domain; (b) spectrogram of jammed signal.

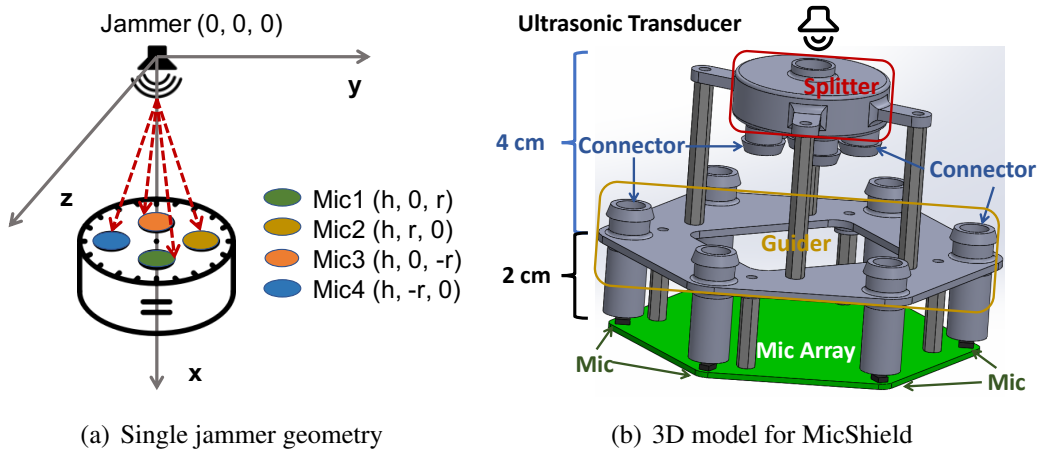


Figure 2.10. Geometric model for MicShield design.

amplifier under the same setup of our previous SPL experiment. Figure 2.8(b) plots the resulting spatial distribution of SPL. Although the microphone hardware specifies 120 dBA SPL, we find that when the SPL exceeds 100 ~ 110 dBA, it already causes complete gain suppression. Correspondingly, when the VA's microphone is placed within the yellow region in Figure 2.8(b), gain suppression occurs effectively.

As shown in Figure 2.8(b), with the amplifier (fixed to 3W to avoid audibility), the gain suppression works when the jamming transducer stays within 14 cm towards the microphone. However, for multi-microphone VAs, it is obvious that a single jamming source cannot cover the

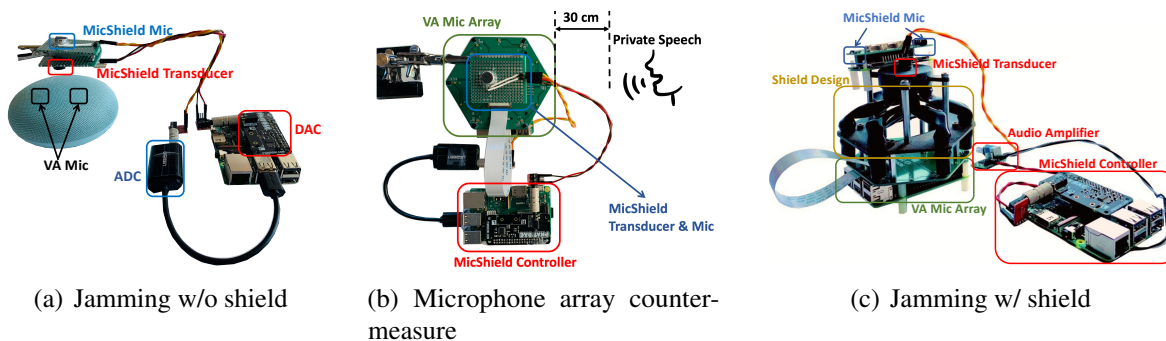


Figure 2.11. Hardware implementations and setups of MicShield.

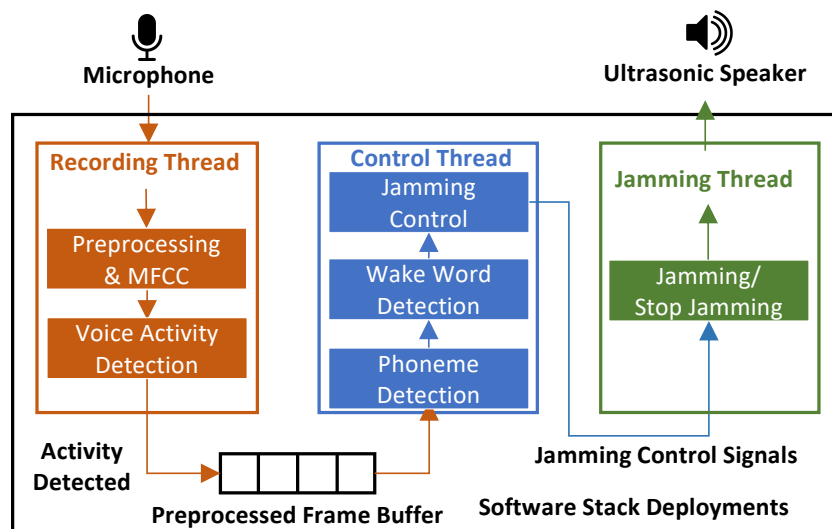


Figure 2.12. Software stack design and implementations.

microphone array. Suppose the transducer is fixed h cm above the microphone array (circular array with radius r), as shown in Figure 2.10(a). Even with the amplifier, the largest radius of the gain suppression region is only 4.5 cm when $h = 5$ cm (Figure 2.8(c)). In contrast, many of the mainstream smart speakers (*e.g.*, Google Home and Apple HomePod) have a radius larger than 4.5 cm, which cannot be covered by the jammer. We present our solution in the next section.

Acoustic Waveguide Design: To address dilemma $D2$, we design a physical shield which can extend the coverage of a single ultrasonic transducer to jam large microphone arrays. Our basic idea is to leverage an acoustic wave guide to redirect the jamming signal to fully saturate the multiple microphones.

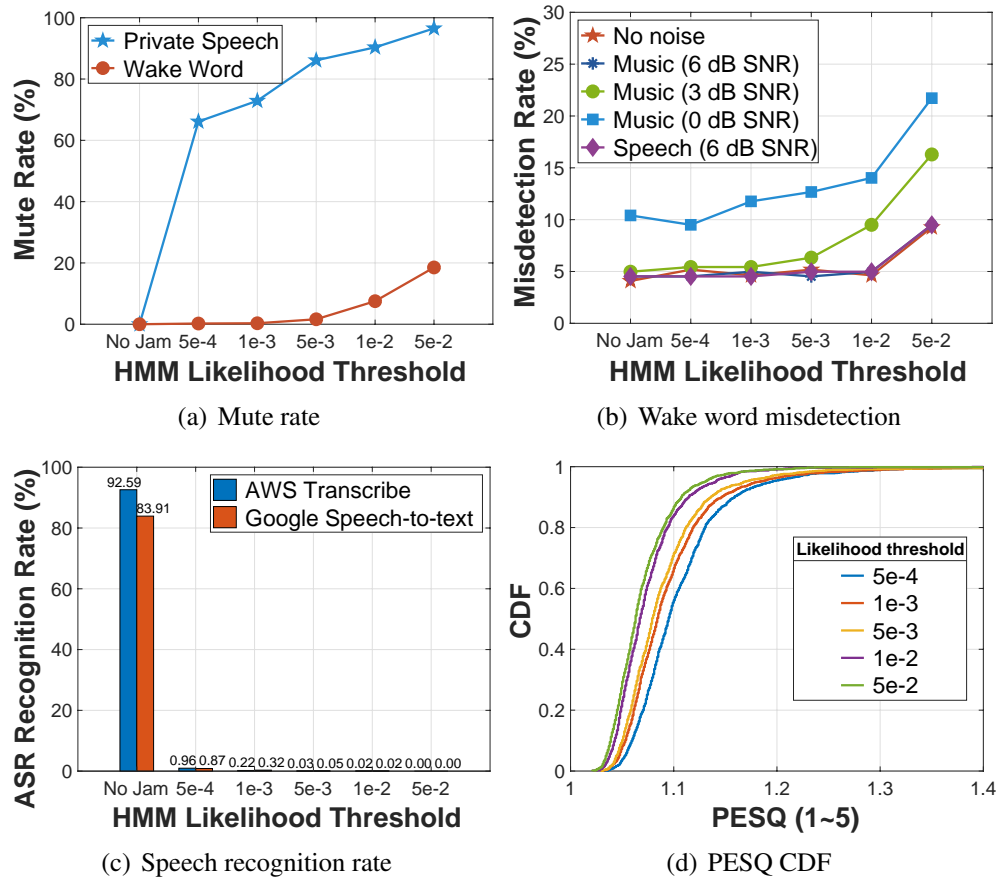


Figure 2.13. Micro benchmark using the wake word of “Alexa”.

Acoustic Waveguide Design: Our acoustic waveguide splits the transducer’s output signals using multiple silicone tubes and redirects them towards the microphones. We choose the 10 mm diameter tube ($>$ half-wavelength (4.2 mm)) to prevent the volume attenuation due to thermo-viscous effects [227]. Figure 2.8(c) profiles the SPL generated by a flexible silicone tube which is 20 cm in length and connected to a single ultrasonic transducer. It shows that, the sound is much more directional than the case without the tube. As a result, the acoustic waveguide also isolates the *self-interference* from the MicShield’s ultrasonic transducer to its own microphone. So it can keep detecting the wake word while jamming the VA’s microphone. Meanwhile, it improves the directionality and hence propagation distance of the sound signal (see Figure 2.8(c)). Therefore, if we can connect the transducer with multiple sound tubes, each tube can jam one microphone in an array.

Geometrical Design: We design a “acoustic multiplexer” to split the acoustic jamming signal across multiple tubes. Figure 2.10(b) shows the 3D mock-up design, which contains two parts, *i.e.*, “splitter” (top) and “guider” (bottom). The ultrasonic transducer transmits jamming signals from the top of the “splitter”. The bottom of the “splitter” comprises multiple connectors which connects to the “guider” through tubes. Then, the “guider” uses another set of tubes to guide the jamming sound to each of the VA’s microphones, thus enabling the gain suppression jamming. Note that the “guider” can be customized for different VAs, depending on the VAs’ form factor and number of microphones. Figure 2.10(b) shows a typical geometrical design for the Respeaker 6-Mic circular array [16].

One might be concerned that the 3D printed shield may block the voice from the user. We conduct one experiment to quantify such degradation. We play the speech sound using smartphone and record the signals using the ReSpeaker 6-Mic Array [16] with or without our mock-up shield in 7 different angles from 0° , to 90° , at a step of 15° . Figure 2.8(d) shows the attenuation with the shield. We observe the degradation is only 1 dB even in the worst case when the sound source is on the top of the VA. This minor effect is unlikely to harm the VA’s sensitivity in voice recognition.

Another practical concern for MicShield is the ultrasound safety issue. As suggested by World Health Organization (WHO), the human-exposure limits for 40 kHz airborne acoustic radiation should be less than 110 dB SPL when the duration of exposure does not exceed 4 hours per day [9]. As shown in Figure 2.8(d), MicShield guarantees that the transmitted SPL is always within the safety limits. Besides, our acoustic waveguide design further isolates the ultrasound and shrinks the high SPL region to prevent harm to users.

2.6 Implementation

2.6.1 Hardware

We implement MicShield using low-cost OTS components. Figure 2.11(a) shows the MicShield prototype for single-microphone case, comprising one ultrasonic transducer, one microphone and RPi. The RPi interfaces with a Pimoroni pHAT DAC 24-bit/192 kHz Sound Card [8] and then an ultrasonic transducer, in order to generate inaudible jamming sound. To minimize self-interference and maximize jamming, the MicShield ultrasonic transducer is facing away from its own microphone while towards the VA microphone. For multi-microphone case, Figure 2.11(c) shows the custom-built shield to guide the jamming sound to each microphone. To achieve a reasonable operating range in detecting wake words, MicShield uses ReSpeaker 2-Microphone Pi HAT supporting up to 3 m sensing distance [16]. The sampling rate is set to 16 kHz. The overall cost of single and multiple microphone schemes are around \$25 and \$35, respectively, exclude the RPi and 3D printed mechanical shield. We expect a great cost reduction and form factor enhancement by additional engineering efforts and integrating majority of components into System-on-Chip (SoC).

2.6.2 Software

Our software stack runs on the RPi, which implements the pipelines described in Section 2.4.2 and Figure 2.4. As shown in Figure 2.12, our implementation has three parallel threads, for *recording*, *control*, and *jamming* purposes. The *recording thread* captures and preprocesses the sound signals based on the policy in Phase A. To reduce energy consumption, this thread also determines whether it needs to emit jamming signals, with an energy-based voice activity detection scheme [120]. The *control thread* takes each MFCC frame features along with the previously preprocessed frames' MFCC features as input, and then determines whether to pass the jamming command to the jamming thread, based on the results of automatic jamming control algorithm (see Figure 2.4). The lightweight RNN and HMM in the automatic jamming control

are implemented using Theano[18]. Upon receiving the jamming control, the *jamming thread* will start jamming. For single-microphone case, we transmit jamming noise to ensure the SPL at the receiver microphone being around 95 dBA, which is measured 3 cm away from the ultrasonic transducer. For multi-microphone case, we ensure the SPL of jamming noise received being approximately equal to 110 dBA, which is measured 1 cm away from the sound tubes.

2.7 Experimental Evaluations

2.7.1 Micro Benchmarks

We first evaluate MicShield following the metrics in Section 2.3, and discuss the trade-off between the wake word misdetection and the jamming effectiveness. We use the TIMIT corpus [81] as the private speech dataset and the Alexa dataset [35] as the wake word dataset. We split the private speech data into training and testing set, containing 190 min and 70 min speech corpus for 630 and 168 users, respectively. Similarly, the Alexa dataset is split into 200 and 167 wake words for 50 users and 37 users, respectively. Since OTS VAs do not allow access to raw recorded signals, we use the ReSpeaker 6-Mic Array with RPi [16] to record both the private speech and the wake word sound in this experiment.

- **Mute Rate:** Figure 2.13(a) shows the private speech mute rate and the wake word mute rate under different HMM likelihood thresholds. A higher threshold, *e.g.*, 0.05, leads to high mute rate for both private speech and wake word. When the threshold falls below 0.001, the wake word’s mute rate becomes almost 0%, whereas the private speech words’ mute rate remains high (72%).

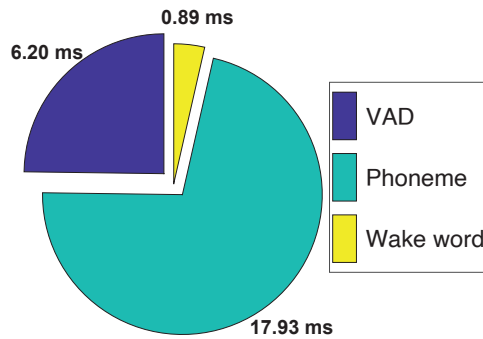
- **Wake Word Misdetection Rate:** Since commercial VAs do not expose their wake word detection algorithms, we playback the recorded wake word sounds under MicShield jamming to Amazon Echo Dot [3], in order to measure the end-to-end misdetection rate. Figure 2.13(b) shows that, compare to the case without MicShield, the wake word misdetection rate with MicShield’s selective jamming does not degrade as long as the likelihood threshold is less than

0.01. When the likelihood threshold is 0.005, only 1/167 wake word from 37 users (16 females, 21 males), which can be recognized without MicShield, is misdetected with MicShield in the Alexa testing set.

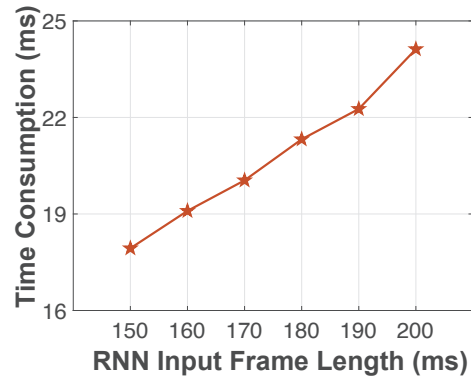
- **Wake Word False Alarm:** When testing the private speech, there is only 1 false alarm of wake word, out of the 12 hours of normal speech. This means the jamming effectiveness is virtually unaffected by wake word false alarms. Meanwhile, as we discuss in Section 2.4.3, the wake word false alarm will not affect the VAs basic functionality either.

- **Jamming Effectiveness:** We evaluate the jamming effectiveness by using PESQ and speech recognition rate. We use ASR algorithms (*i.e.*, Amazon Transcribe [11] and Google STT [13]) and human recognition to evaluate the jammed speech recognition rate. Figure 2.16(c) shows that the speech recognition rates for both ASR algorithms are less than 1% when the threshold is 0.0005 (corresponding to mute rate 66% for private speech). We found that the recognized words are mainly short words with the first phoneme close to /ə/, *e.g.*, “I”, “a”, “as”, “all”, “also”, *etc.* In addition, human users cannot interpret the jammed speech either. The PESQ of all the jammed speech signals is less than 1.4, under different likelihood thresholds, as shown in Figure 2.13(d).

- **Impact of Environmental Noise:** To evaluate the robustness of MicShield, we mix the wake word sound plus private speech with two types of environmental noises, *i.e.*, speech noise and music noise, which come from AudioSet [82], an ambient noise dataset widely adopted in speech recognition research. We play these two types of the environmental noises by an additional loudspeaker while using MicShield. The SNR levels between private speech and noise vary between 6 dB, 3 dB, and 0 dB. Figure 2.13(b) shows the misdetection rate of wake word. When the HMM likelihood threshold is less than 0.005, the wake word misdetection rate remains the same as the case without MicShield. Here we omit the results where the speech SNR is below 3 dB, since the VA itself is no longer usable in such cases — the wake word misdetection rate exceeds 50% even without MicShield. Meanwhile, we find that the wake word false alarm and jamming effectiveness is barely affected in noisy environments, and instead is more sensitive



(a) Contributions of latency



(b) Frame length v.s. Time

Figure 2.14. Analysis of system time consumption.

to the HMM likelihood threshold.

- Real-world Usage Experiments:** We deployed the VA with MicShield (see Figure 2.11(b)) across four different rooms, *i.e.*, bedroom, kitchen, living room, and bathroom, and asked the users to use the VA as usual for 5 days. 120/125 voice commands, including some interrupting voice commands, *e.g.*, “Alexa, stop!”, are correctly responded by VA with MicShield. MicShield is not sensitive to the VA locations since we use the MFCC features, which is more related to the speech characteristics rather than environmental characteristics, as the framewise phoneme recognition input. The only 5 missed voice commands are due to the low-volume of the voice commands which can not trigger the jamming suspending policy.

To summarize, when the HMM likelihood threshold is 0.01, MicShield is able to maximize the jamming effectiveness, achieving 90.4% mute rate and 0.02% speech recognition rate for private speech, without affecting the VA’s ability to detect wake words even in noisy environments across different VA locations.

2.7.2 Latency Analysis

The latency for MicShield from receiving the first frame of the wake word to stop jamming is 25.02 ms (≤ 60 ms). This will not degrade the wake word misdetection rate (see Figure 2.3(a)).

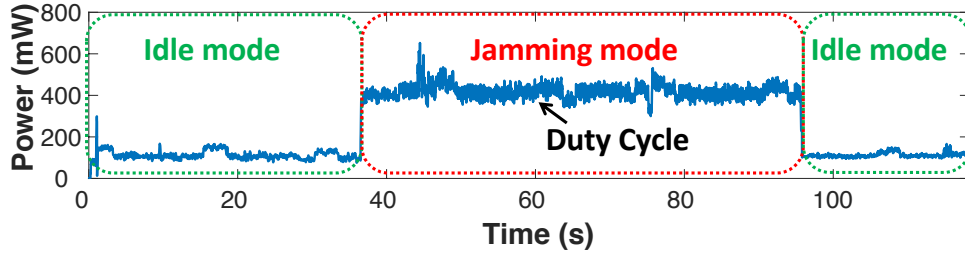


Figure 2.15. An example instantaneous power.

We evaluate this by measuring the processing latency of the MicShield software stack running on RPi 3B+. The processing time includes 3 parts: (i) MFCC & VAD; (ii) RNN-based phoneme recognition; (iii) HMM-based wake word recognition. As shown in Figure 2.12, upon receiving audio stream from the analog front-end, the recording thread first performs the MFCC and Voice Activity Detection (VAD), incurring ~ 6.20 ms latency. Second, the control thread performs the automatic jamming control (phoneme recognition and wake-up word recognition) to determine whether to pass the jamming control signals to the subsequent jamming thread. In this thread, the latency and control frequency are dominated by sequence length of RNN-based phoneme recognition significantly. Figure 2.14(b) shows the measured correlations between control thread latency and input frame length. We choose input length of 15 frames (each frame takes 25 ms-window length and adjacent frames have a 15 ms-overlaps), and the control thread has $17.93 + 0.89 = 18.82$ ms latency each time. The overall latency is 25.02 ms for MicShield to response to a 25 ms frame speech, which is sufficient to pass the wake-up word to VAs based on our previous discussion in Section 2.3(a).

2.7.3 Energy Consumption

Our current MicShield prototype, as our proof-of-concept, incurs an estimated energy consumption of 2 Wh per day.

Power Consumption of Major System Components: We use INA 219 [5] to measure the instantaneous power consumption of major system components, *i.e.*, transducers, microphones and RPi. Table 2.1 summarizes the power consumption averaged over 10 min for the

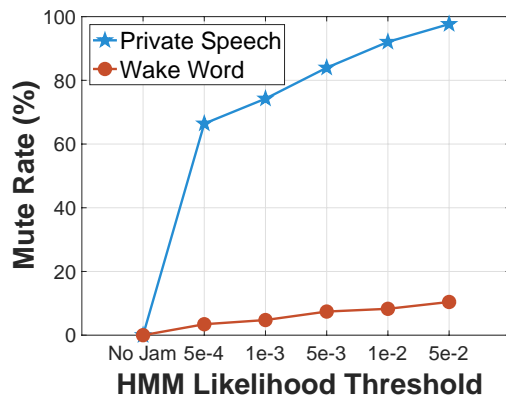
Table 2.1. Power consumption (mW) in controlled settings.

(a) Single-microphone VA				
	Transducer	Microphone	RPi	Total
Idle	/	12.99	84.61	97.60
Jamming	121.03	12.99	262.33	403.35

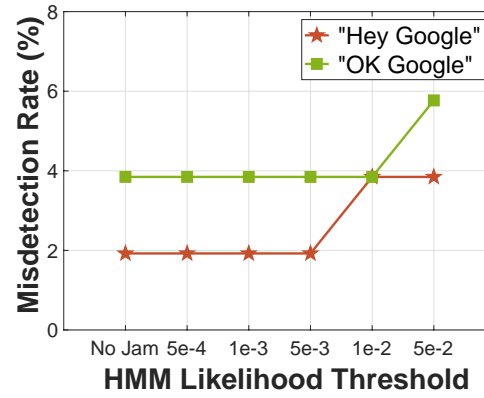
(b) Multi-Microphone VA				
	Transducer	Microphone	RPi	Total
Idle	/	31.52	84.61	116.13
Jamming	156.83	31.52	262.33	450.68

single- and multi-microphone setup. For single-microphone VA, MicShield consumes 97.6 mW and 403.35 mW in the idle and jamming mode, while for multi-microphone VA, the power consumption is 116.13 mW and 450.68 mW, respectively. We notice more than 65.04% of power is attributed to the RPi, which centers on an ARM based SoC and running a full fledged Debian OS. However, we expect the a product version of MicShield can be implemented based on dedicated low power chip, without the power hungry OS kernel modules and background daemon.

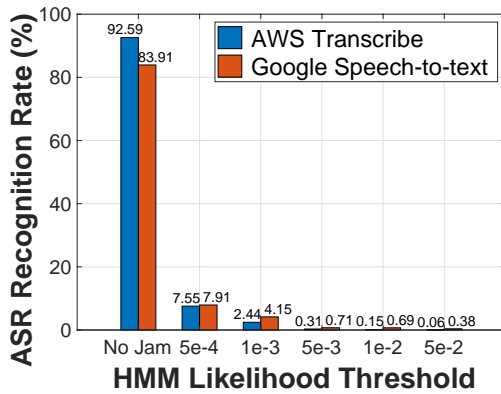
Approximation of Energy Consumption in a Practical Settings: In daily usage scenarios, it is known that a user speaks for about 2 hours on average per day [152]. Accordingly, we assume MicShield works in 14 hours of idle mode and 2 hours of jamming mode in a single day. Thus, MicShield would incur an energy consumption of ~ 2 Wh per day. This means that, with a typical smartphone battery, MicShield can work for about 8 \sim 10 days [62]. The energy would be largely reduced if a low-power DSP or ASIC is used as the substitute of RPi [160]. For example, a recent developed AI chip by Syntiant shows high potentials to be applied in our case with the power consumption down to the order of hundreds μ W [160]. Alternatively, we also noticed a majority of multi-microphone VAs draw power from the mains, which is also applicable to MicShield.



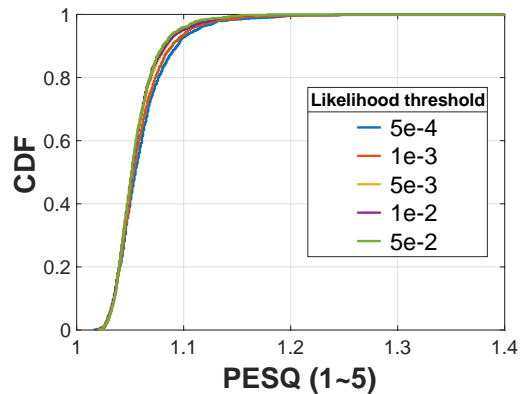
(a) Mute rate



(b) Wake word misdetection



(c) Speech recognition rate



(d) PESQ CDF

Figure 2.16. Micro benchmark using the wake words of “OK Google” & “Hey Google”.

2.7.4 Generalization

Our previous design and experiments use the Amazon Echo Dot by default. However, MicShield can be well generalized to alternative VAs. In this section, we show the generalizations of MicShield for Google Home [85] and Amazon Echo [31], 2 VAs that currently dominate the market [260].

- **Generalizations across Wake Words:** The same VA can use different pre-defined wake words. For example, Amazon Echo supports to change the wake of as one of the 4 words: “Alexa”, “Echo”, “Amazon”, and “Computer” [31]. To accommodate this, MicShield stores the trained RNN and HMM models in the persistent memory for all the wake words that a user needs. It only needs to change the RNN and HMM models to adapt to the wake word modifications. Our current design only supports pre-defined wake words. However, this will not limit the generalizations of our design, as all current VA devices only supports limited number of wake words. Besides, new wake words can be trained following the same pipeline. The main difference for different wake words is the phoneme sequence pattern. Table 2.2 lists the percentages of the words in CMU Pronouncing Dictionary [17] that have different number of same consecutive phoneme as wake words used by Amazon Echo and Google Home. First, relatively large number of words (0.12% of 134000 words) shares the same first 4 phonemes with the wake word “Computer” since “Comp” (/kɑmp/) is a word prefix. This means MicShield will have relatively high risk of leaking private words with the same word prefix as the wake word. Second, Google Home chooses to use the short utterances as the wake word, *e.g.*, “Hey Google” and “OK Google”. This leads to that MicShield will leak the onset word of the short utterances, *i.e.*, “Hey” and “OK”. MicShield can still protect the speech privacy in most of the practical settings, and we encourage users to use the words without the word prefix and onset word, *e.g.*, “Alexa” and “Amazon”, to better protect the speech privacy.

MicShield can be well generalized to multiple wake words at the same time. Some VAs support multiple wake words triggering, *e.g.*, users can trigger Google Home by “OK Google”

Table 2.2. Percentage of words in CMU Pronouncing Dictionary [17] that have similar number of consecutive phoneme sequence as different wake words.

# of Consecutive Phoneme	1	2	3	4	5
“Alexa”	2.05	0.19	0.02	0.01	0.01
“Amazon”	1.46	0.15	0.03	0	0
“Echo”	1.51	0.17	0.01	/	/
“Computer”	9.73	1.13	0.25	0.12	0.01
“Hey Google”	5.00	0.19	0.01	0	0
“OK Google”	0.48	0.03	0	0	0

and “Hey Google”. To evaluate the performance of MicShield in this case, we use the Google Home as an example. Our dataset is composed of 2 wake words speech from existing cloud based Text-To-Speech (TTS) services and real-world participants. For TTS based approach, we use the wake word speech from 24 “virtual” participants, generated by Google TTS [14], Amazon Polly [10] and IBM Watson TTS [15]. Our dataset also includes speech examples from 5 real-world participants. Overall, we collect 79 samples from 29 people for “OK Google” and “Hey Google”. To train HMM and RNN model in automatic jamming control pipeline (see Figure 2.4), we shuffled and used 70% and 30% samples for training and testing purpose. The mute rate, misdetection rate and jamming effectiveness is plotted in Figure 2.16. A comparison with Figure 2.3 shows a negligible performance degradation for the 2 wake words setting.

- Generalizations across VA Devices:** The geometry and sensitivity of microphone array varies across different VAs. To show the generalizations across devices, our evaluation is based on Amazon Echo Dot (4 microphones) and Google Home Mini (2 microphones). Note that these 2 multi-microphone VAs do not allow access to the private speech, but will record the history of the voice command after each wake word. Thus, we disable the jamming control policy for voice command and force the MicShield to only pass the wake word and obfuscate the voice command, so as to obtain the jammed results of commercial VAs. We use a smartphone to transmit the sound signals with 70 dBA SPL to first wake up the VA, and then continue with the voice commands. The voice commands are selected from the top 20 most popular ones [1].

Both Amazon and Google provide the metadata (content of the recognized voice command) for each request. The experimental results show that, all the metadata of these 100 jammed voice commands is “Audio could not be understood” or “unknown voice command”. Amazon also provides the received audio signals of the voice commands. Whereby this, we showed the real participants and ASR algorithms cannot recognize the jammed audio signal neither. This means that MicShield can effectively protect the private speech for these different devices.

2.8 Limitations and Future Work

Sensitivity of Microphones: Our experiments indicate the achievable distance between MicShield and sound source highly depends on the sensitivity of microphone on the MicShield. Our current prototype supports up to 3 m range for wake word detection. For many single-microphone VAs, *e.g.*, smartphones, the user-device distances are generally limited within 1 m. MicShield can well address such usage scenarios. However, commercial microphone array based VAs may be able to achieve longer than 3 m detection range. To ensure comparable detection range, MicShield needs to adopt similar hardware setup as such commercial VAs, *e.g.*, using a microphone array along with self-interference cancellation, acoustic echo cancellation and beamforming algorithms. We expect a commercial buildout of MicShield can be easily equipped with such capabilities.

Attacks on MicShield: MicShield’s threat model assumes that the VAs are untrustable, but MicShield itself still needs to run the wake word detection mechanism, albeit always offline. An attacker may attempt to defeat MicShield by forcing the VA to secretly play the wake word sounds with low volume or inaudible voice [281], so as to cheat MicShield and stop its jamming. The attackers may also employ other speaker devices, *e.g.*, TV, to play the wake words. However, such forged voices can be easily identified, *e.g.*, by installing multiple microphones on MicShield to locate the sound source, or through liveness detection methods [130, 283]. These attacks and countermeasures targeting the wake word detection have been well explored, and are beyond

the scope of our work.

2.9 Conclusion

The always-on microphones on voice assistants (VAs) have raised serious privacy concerns. In this paper, we propose MicShield, the first system to automatically protect speech privacy against always-on microphones. MicShield introduces a novel selectively jamming mechanism, which can obfuscate private speech while passing legitimate voice commands using phoneme-level features. We prototype implementation and experiments verify the feasibility and effectiveness of MicShield in protecting speech privacy without degrading the VAs' basic functionalities. MicShield marks a critical step in addressing the potential privacy risks of VAs.

2.10 Acknowledgments

This chapter contains material from “Alexa, Stop Spying on Me: Speech Privacy Protection Against Voice Assistants”, by Ke Sun, Chen Chen, and Xinyu Zhang, which appears in the 18th ACM Conference on Embedded Networked Sensor Systems (SenSys), 2020 [211]. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Stealing Permission-protected Private Information From Smartphone Voice Assistant Using Zero-Permission Sensors

3.1 Introduction

Voice User Interfaces (VUIs) allow a user to interact with a smartphone through voice/speech commands, which significantly improves the smartphone's usability and accessibility. Voice Assistants (VAs), *e.g.*, Google Assistant and Apple Siri, and voice-guided navigation apps, *e.g.*, Google Maps and Apple Maps, represent the most popular apps that employ the VUIs. To function properly, these apps have to access many strong permissions related to user privacy, *e.g.*, calendar, contacts, locations, microphone, SMS, and storage. Then they can vocally respond to user queries through the built-in loudspeaker. Such VUI responses often contain sensitive private information (see examples in Table 3.1).

On the other hand, motion sensors, *e.g.*, accelerometer and gyroscope, are co-located with the loudspeaker, and can potentially create a side channel to eavesdrop on the loudspeaker through vibration sensing [39, 34, 65, 282, 200]. These sensors pose an alarming threat especially since they are accessible by any app on mainstream mobile OS (*e.g.*, Android and iOS) without user permission. However, existing threats mainly target on classifying a small set of digits and hot words [39, 34], rather than natural speech, from the side channel. The potential risk remains

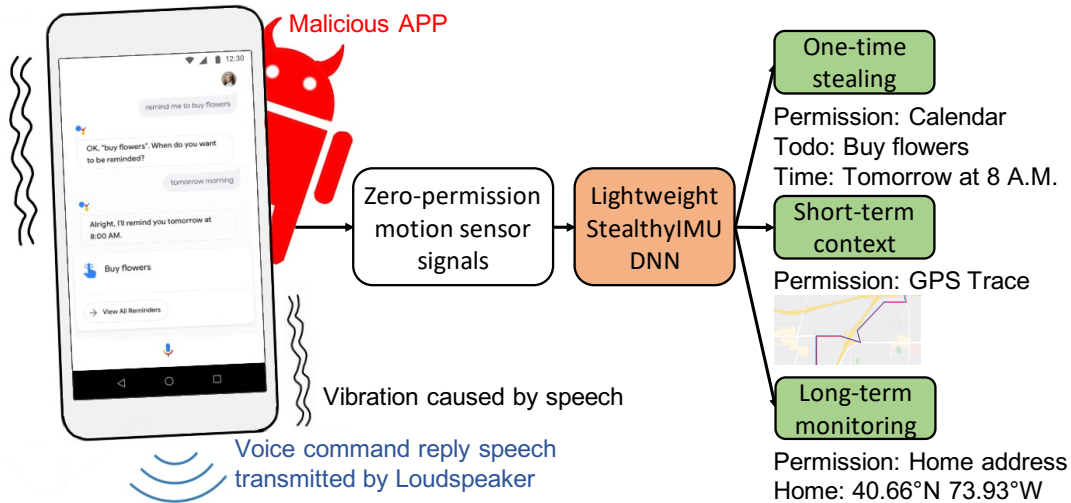


Figure 3.1. StealthyIMU threat model. StealthyIMU is a malware that can be built in as an ordinary app to continuously capture the motion sensor signals, from which it extracts permission-protected private information.

unclear in real-world scenarios.

In this paper, we propose StealthyIMU, a novel and practical threat that *uses the zero-permission motion sensors to extract permission-protected private information from the VUI responses*. These permissions are explicitly granted to the VUI by the user and are strongly associated with user privacy. StealthyIMU is a malware that can be built in or disguise as an ordinary app, to indirectly acquire such permissions through the motion sensor side channel. Fig. 3.1 illustrates our basic threat model. StealthyIMU continuously captures the motion sensor signals (MSS) and identifies the segments associated with the VUI response, from which it recognizes the type and content of the private information. Over time, StealthyIMU can accumulate more context to further improve the accuracy and richness of the information.

To realize StealthyIMU, we need to resolve 4 key challenges. *(i) How to single out the MSS segments of interest with negligible overhead?* First of all, since StealthyIMU does not know when the VUI responses occur, we need to identify the MSS segments associated with VUI responses, out of all the motion signals, without being noticed by the user. Processing the MSS in the cloud is not always feasible since the StealthyIMU app may not have “network

in the background” permission to continuously upload the sensor data. We thus design an on-device two-stage (temporal and frequency domain) detection algorithm, which can identify and segment the sound-induced MSS in the background, and then save them in the app memory with negligible overhead. We further design a lightweight DNN model to single out the subsets of segments that contain an actual VUI response.

(ii) *How to recognize the private content from a segment of MSS containing a VUI response?* Due to the limited sampling rate of smartphone motion sensors (< 500 Hz), recognizing general speech from the motion sensor is a highly underdetermined problem [39, 34]. Unlike existing work, our unique observations are: 1) VUI responses usually come from a small set of machine-rendered voices, resulting in a constrained *user dependent* speech recognition problem; 2) the acoustic characteristics and format of VUI responses are *more deterministic* than natural human speech, making it easier to be recognized. Since the attacker cares more about the meanings rather than wordings, we formulate the StealthyIMU attack as an end-to-end Spoken Language Understanding (SLU) problem, which further evades the need for word level recognition. Specifically, we extract the contents associated with private permissions from the VUI responses and label them in the form of *private entity list* (see Tab. 3.1). Then we design a DNN model to recognize the private entity lists from MSS. To enhance the SLU, we propose a *cross-modality teacher-student training* strategy to distill knowledge from data-rich speech models to guide the training of the MSS model.

(iii) *How to extract the private information by combining the contextual information across multiple VUI responses?* The one-shot SLU from a single VUI response is inevitably error-prone. However, by processing multiple VUI responses targeting the same permission, we can infer the user privacy (*e.g.*, city name and home address) with high accuracy. In addition, by inferring the user’s city and combining the city map along with the contextual information from a series of navigation voices, we can even recover the user’s GPS trace.

(iv) *How to defend against the StealthyIMU attack?* The limited sampling rate of motion sensors brings opportunities to counteract the StealthyIMU attack by modifying the VUI

response speech alone. We propose to *pre-distort* the low-frequency components of speech signals to prevent the motion sensor from capturing the voice-related information, without noticeably affecting the speech quality of the VUI response. Specifically, we design a speech signal processing pipeline to reduce the SNR of the MSS and distort the spectrogram by adding artificial low-frequency components.

To evaluate StealthyIMU, we develop an Android app to collect more than 45,000 VUI responses from Google Assistant and Google Map, along with the MSS in the real world. The VUI responses include 23 types of frequently-used voice commands (see Tab. 3.2). Our evaluation shows that, for the one-shot attack, StealthyIMU can extract the private entities including contacts, location, reminder/ alarm TODO list and time, and search history with an average 85.55% success rate. For long-term monitoring, StealthyIMU achieves 99.8% success rate in recognizing a user’s city name via 5 weather-related queries and locates the user’s home address with 11 m average error simply through 10 navigations to home. For short-term contextual inference, StealthyIMU can combine the city road map and a series of navigation voices to recover users’ GPS trace within 30 m distance error in 80% of cases. Whereas StealthyIMU can be directly deployed on smartphones, we also take the first step to exploit alternative deployment models of StealthyIMU (*e.g.*, in the cloud), and show the trade-off between the required permissions and on-device system resources. On the other hand, our defense approach is able to prevent the StealthyIMU attack, reducing its SNR down to 2.7 dB and success rate to less than 4.94%. For the purpose of reproducing our approach, we release our labeled VUI response, MSS dataset, and the source code ¹.

We make the following contributions in this work:

- We introduce a new threat model that uses zero-permission motion sensors to extract permission-protected private information from VUI responses on smartphones.
- We formulate the StealthyIMU attack as an SLU problem and design a sequence-to-

¹<https://github.com/Samsonsjarkal/StealthyIMU>

sequence DNN model with a cross-modality knowledge distillation strategy to directly extract the private entities from the MSS. Our model is optimized for on-device execution with minimal overhead.

- We design algorithms to realize the short-term and long-term StealthyIMU attack, and demonstrate how it steals user calendar, GPS trace, home address, *etc.* with extensive experiments in real-world scenarios.
- We propose a speech pre-distortion mechanism to defend against StealthyIMU without noticeably affecting the VUI speech quality.

We note that, even if future smartphone OS restricts the motion sensor permission, the StealthyIMU attack can still work—it can pretend to be an innocuous app that needs the motion sensor permission alone, while using it to extract other permission-protected sensitive information.

3.2 Related Works

3.2.1 Motion Leakage via Sensors on Smartphone

To facilitate common user interaction functions (*e.g.*, screen auto-rotation and app layout rendering), motion sensors are designed to be accessible by arbitrary smartphone apps without explicit permissions. Prior work has investigated various security/privacy threats associated with such zero-permission sensors. ACCessory [175] and Adam *et al.* [38] use the accelerometer as a side channel to extract text input, PIN, and unlock pattern on a smartphone touchscreen keyboard. AccelPrint [65] and TapPrints [156] show that accelerometers and users' tapping both possess unique fingerprints and can be used to identify the users. Mole [242], Liu *et al.* [139], Wang *et al.* [239] and Snoopy [142] investigate the motion leakage via smartwatch motion sensors. Further, motion sensors can be used to infer the user's moving trajectories [96, 166, 108, 161]. ACComplICE [96] records the motion sensor signals for more than 15 mins

to recover a relatively long trajectory, and then fits the trajectory to a map based on the shape of the trajectory. Narain *et al.* [166] further designed a graph theoretic model to infer the users' trajectory via accelerometers with 30% top-10 route accuracy. Hua *et al.* [108] demonstrate that accelerometer data can indicate train location. PinMe [161] combines sensory and non-sensory data to infer user location. PowerSpy [154] proposes to use the power consumption change caused by the phone cellular modems to track the user location. Although these attacks are related to the location permission attack in StealthyIMU, they assume either the attacker knows the user's initial location, or the victim is traveling along a small set of known routes. In comparison, StealthyIMU can extract the location by using a single navigation voice, recover the user's GPS route by combining multiple navigation voices, and even reveal sensitive locations such as home addresses without knowing the user's initial location. Further, it can be combined with relative motion tracking [166, 108, 161] to infer more precise user locations. Besides, StealthyIMU can steal a much wider range of private information than motion leakage attacks.

3.2.2 Speech Recognition via Motion Sensors

Typically, the voiced speech of an adult male and female has a fundamental frequency 85 ~ 180 Hz and 165 ~ 255 Hz, respectively. A small portion of the low-frequency speech signals can be captured by the smartphone motion sensor (typical sampling rate 200 ~ 500 Hz). Prior research exploited this side channel to recognize speech from the loudspeaker vibration. Gyrophone [153] achieves about 50% success rate in identifying 10 speakers, and 65% and 26% for speaker-dependent and speaker-independent 10-digit speech classification. Accelword [282] achieves a hot word detection accuracy of 85% in static scenarios and 80% in mobile scenarios among 10 users. Their threat model assumes that the loudspeaker and the attacking sensor are separated but share a common surface (*e.g.*, desktop) which is not always feasible. Speechless [32] analyzes the MSS side channel and shows that through-air human speech does not noticeably affect the motion sensors. Recently, Spearphone [33, 34] and AccelEve [39] further examined the feasibility of using smartphone motion sensors to recognize

the speech reverberations from a co-located loudspeaker. StealthyIMU differs from existing work in two aspects. First, the threat model and end goal are different. Previous work only demonstrates the possibility of using MSS to recognize a predefined set of numbers or words. In contrast, StealthyIMU can extract complete semantic information. It reveals a real-world privacy threat where MSS can steal crucial smartphone permission-protected private information, *e.g.*, GPS trace, location permission, calendar, contacts, *etc.* Second, the attacking vector and methods are different. Previous work casts the eavesdropping of numbers/words into a simplified classification problem. In comparison, we formulate the StealthyIMU attack as an end-to-end private speech understanding problem, and design speech and natural language processing algorithms to efficiently extract the private information from the MSS side channel. We also investigate the one-time stealing, short-term contextual inference, and long-term monitoring threat models.

3.2.3 Spoken Language Understanding

SLU systems infer the intents and meanings of a spoken utterance [230]. This is critical for VUIs, which convert the speaker’s utterance into action or query. SLU typically consists of two tasks: Intent detection, a classification problem where utterances are labeled with predefined intents; Slot filling, a sequence labeling task that identifies the semantic concepts. The basic problem behind StealthyIMU is close to SLU. It tries to first classify the leaky permission (corresponding to intent detection) and then recognize the private information (corresponding to slot filling) from the MSS. Existing SLU systems can be categorized into two classes: a cascaded pipeline approach and an end-to-end approach [199]. The former contains an automatic speech recognition (ASR) system to decode the speech into text, followed by a natural language understanding (NLU) system that interprets the meaning of the text [58]. The performance highly depends on the accuracy of the ASR stage, whose error may be propagated and amplified in the NLU [50]. The latter approach uses a single DNN model to map the speech signals directly to the speaker’s intent without an explicit text transcription [199]. In this paper, we systematically

analyze the unique challenges of StealthyIMU, and choose to design an end-to-end model to extract the private permission from the MSS (Sec. 3.5.1).

3.3 Threat Analysis

To function properly, VUI apps have to access many strong permissions related to user privacy. For example, Google Assistant requires 7 permissions, including calendar, contacts, locations, microphone, phone, SMS, and storage. StealthyIMU targets the scenario where a malware uses zero-permission motion sensors to record the vibrations caused by the voice assistant and voice-guided navigation apps, and then steals the permission-protected private information from these apps. To establish the threat model, we assume the adversary can mislead the victim to install an ordinary app that contains the StealthyIMU malware, which then continuously collects MSS in the background. Once the app receives the MSS, it will detect whether there exist any VUI responses, recognize the types of responses, infer the private entities from the responses, and recover the explicit results of the stolen permission-protected private information. Finally, these information will be saved in the app locally or uploaded to a cloud server (if permitted), to monitor the user's real-time location and trajectory, understand the user's daily schedule (*e.g.*, medication input), personal habits and preferences, *etc.* Notably, popular mobile browsers such as Firefox allow motion sensor access by default. So, besides hiding itself in a normal app, StealthyIMU can also be launched through an executable (*e.g.*, Javascript) on a seemingly innocuous website.

Attacking requirements: The StealthyIMU attacker needs no explicit permission to capture the MSS. However, it needs to ensure neither the smartphone OS nor the user will notice the attack. To this end, it needs to carefully make several trade-offs related to system resource requirements.

- *Voice detection on device v.s. continuous network streaming:* Since StealthyIMU does not assume to know when the motions come from the VUI, the first step of StealthyIMU is

to single out the VUI-induced motion signals from all the MSS. Although such computation overhead of StealthyIMU can be completely outsourced to the adversaries' cloud server, it is not reasonable to assume that the malicious app always has the "network in the background" permission to continuously upload the MSS stream. Consequently, the malicious app should be able to perform on-device identification of the MSS associated with VUI responses, with minimal overhead. Note that StealthyIMU still needs "network" permission to upload private information to adversaries. However, the on-device processing helps StealthyIMU to avoid continuous network streaming. Uploading private information to adversaries can be accomplished when the malicious app is active, and the uploading data size is significantly reduced compared to uploading the MSS stream.

- *Memory v.s. storage*: After identifying the MSS of interest, StealthyIMU can choose to save the segment of signals in the app memory or local storage, depending on whether it has the "storage" permission. Note that if the app saves the segment in memory, only a small amount of memory is needed (16 KB per VUI response on average. see Sec. 3.9.6), so it is indistinguishable from an innocuous app.

This paper takes the first step to exploit these choices and navigate the trade-off between required permissions and on-device resources. *We design StealthyIMU such that it can steal private information even without continuous network streaming and "storage" permissions*, by only using on-device computation. On the other hand, if StealthyIMU has access to the continuous network streaming and "storage" permissions, it can execute the majority of the attack vector on the cloud.

Target permissions: We investigate frequently used voice commands related to reading the VAs' permissions [88] and summarize a list of vulnerable permissions and example attacks in Table 3.1. We put these privacy-related voice commands into 3 categories, *i.e.*, set/check calendars and alarms, make calls and send messages, and ask questions. In addition, we put voice-guided navigation in a separate category, as it only utters the navigation voices without a conversation with the user. StealthyIMU can steal 5 reading permissions, *i.e.*, reading the

Table 3.1. StealthyIMU potential attacking permissions and VUI response examples. StealthyIMU proposes to use the MSS caused by the responses of the VA (In the “Examples” column, utterances start with “A”) to infer the permission-protected private information from the VUI.

Voice Command	Permission	Examples	Private Entity List
Set/Check calendar/reminder	Read calendar	Q: Set a reminder to check bank account. A1: Got it. “[Check bank account]”. When do you want to be reminded? Q: Tomorrow 8 PM A2: Sure. I will remind you [tomorrow at 8 PM].	A1: Calendars [Todo: Check my bank account] A2: Calendars [Time: 8 PM]
Make calls and send messages	Read contacts Read SMS	Q: Send a message to my father. A3: Sure. What’s the message? Q: Call me back. A4: Sending a message to [Bob] saying “[Call me back]”.	A3: A4: Message [Contacts: Bob, Content: Call me back]
Ask Question	Weather	Q: What’s today’s temperatures? A5: Temperature in [New York] tonight is predicted to be 54 °F.	A5: Weather [City: New York]
	Navigation	Q: Navigate me to Los Angeles. A6: OK. [Los Angeles]. Let’s go.	A6: Navigation [City: Los Angeles]
Navigation APP	Stock, Sports, Music	Q: What is the stock price? A7: [Amazon.com] at XXXX dollars.	A7: Search [Company: Amazon]
	Access coarse location	A8: In 600 feet use the right lane to [turn right] onto [Hollywood Boulevard].	A8: Navigation [Intent: Turn right, Road: Hollywood Boulevard]
		Access fine location	A9: Use the left two lanes to [turn left] onto [Western Avenue].

calendar, contacts, SMS; accessing the location (coarse and fine), and search history. The “Examples” column shows the conversational examples between the user (Utterances starting “Q”) and the VA (starting with “A”). The last column, “*Private Entity List*”, shows the information that StealthyIMU aims to extract from the MSS. With the inferred private entities, StealthyIMU indirectly steals the permission-protected private information from the VA and the voice-guided navigation app.

3.4 Motion Sensor Signal (MSS) Preprocessing

In this section, we introduce the lightweight on-device preprocessing mechanisms to prepare StealthyIMU for an attack. Basically, we first select the motion sensor channels based on the SNR. Then we design a near-zero overhead algorithm to continuously detect and segment the sound-induced samples in the MSS. Finally, we use a voice identification model to identify whether a detected segment is associated with a VUI response from the co-located loudspeaker.

3.4.1 Choosing the Sensor Type and Channel

Figure 3.2 shows the accelerometer signal waveform when a smartphone (*i.e.*, Samsung S8) loudspeaker is playing the VUI response at 80% volume. We quantify the motion sensor’s response to sound by $\text{SNR}_{dB} = 10 \log_{10} \frac{P(S)}{P(N)}$, where $P(\cdot)$ is the sum of squared magnitude values; S and N are two series of MSS recorded with and without the loudspeaker voice. Prior work [39] showed that the speech-induced vibration may cause different SNR on different sensors and sampling channels. The accelerometer’s Z axis signals always have the strongest SNR, regardless of the smartphone placement and sound volume, because the vibration generated by the loudspeaker is always perpendicular to the smartphone’s motherboard (along Z axis) [39]. Although using all the IMU sensors for StealthyIMU may improve its performance, it will also increase the attacking overhead significantly. *We thus choose the accelerometer’s Z axis signals alone as the input in StealthyIMU.*

Existing IMU-based sensing systems [39, 33] save an entire stream of signals locally

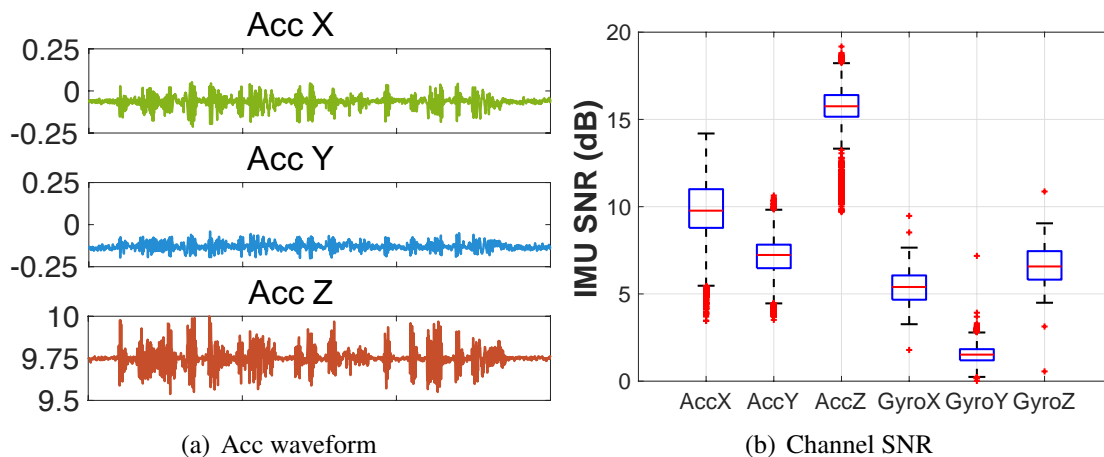


Figure 3.2. Motion sensor channels SNR

and preprocess it offline. However, as discussed in Sec. 3.3, StealthyIMU may not always have permission to do so. Instead, we propose a real-time signal processing workflow to detect and segment voice-induced accelerometer signals with minimal computation and memory overhead. Meanwhile, our detection algorithm needs to achieve a high true positive rate, while maintaining a reasonable false positive rate in order to limit memory usage.

3.4.2 Real-time Detection and Segmentation of Voice in MSS

One major challenge for preprocessing lies in the unstable sampling frequency of the motion sensors, which varies over time and across different smartphones. Prior work [39] resamples the accelerometer signals and normalizes to the same sampling frequency frame by frame by using linear interpolation, which incurs too much computational overhead (1 multiplication and 2 additions per sample). In contrast, our voice detection is an ultra lightweight two-stage algorithm. The first stage detects the vibration of interest based on an empirical *std.* threshold, to rule out the cases where the smartphone is either static or moving abruptly. The second stage resamples the signals to 500 Hz, and then normalizes the magnitude values. Resampling here incurs an acceptable computational overhead since only a small fraction of MSS passes the first stage. Then, we apply a Discrete Fourier transform (DFT), and remove

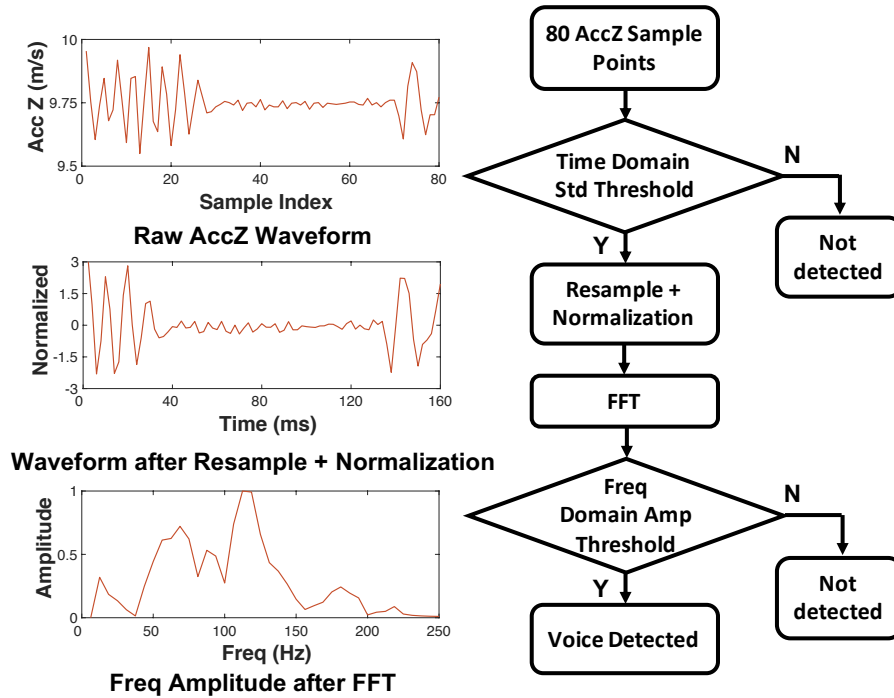


Figure 3.3. Ultra Lightweight Voice Detection Pipeline

those segments that do not contain significant high-frequency components.

Finally, we need to segment the signal buffer associated with an entire voice sequence. Based on a real-world large scale data set that we collected (Sec. 3.8.1), we found most VUI responses last 2 s to 8 s. Thus, we choose to keep a signal buffer of 4000 points (8 s duration at 500 Hz) in the app memory. Within this buffer, we count the number of 80-point segments that are detected as voice-associated. If the detection is positive in the first 80-point segment of the buffer and there exist more than k such segments ($k = 2$ as default), the buffer will be considered as potentially containing the VUI response and stored in memory. Note that a single buffer of float type samples only consumes 16 KB (4000×4 Bytes), which means we can save 500 potential segmented buffers with a small 8 MB memory.

3.4.3 Feature Extraction

The feature extraction module first resamples the signal segments to 500 Hz, and then performs an STFT to obtain the spectrogram. The STFT uses a window length of 80 ms and

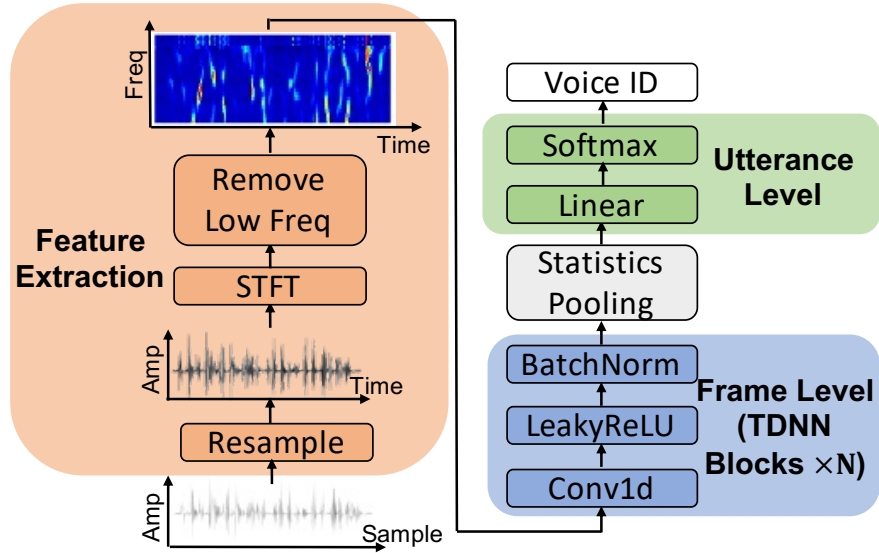


Figure 3.4. Feature Extraction and Speaker Identification

a hop length of 20 ms to guarantee 50 frames per second. This is larger than typical speech recognition window size (25 ms) and hop length (10 ms) because the sampling rate of MSS is much lower. Finally, to mitigate the accelerometer signal noise due to motion artifacts, we remove the frequencies lower than 62.5 Hz following [134, 119], and create 30 frequency bins within the frequency range of (62.5, 250] Hz. The resulting features are 50×30 scalars per second, which serve as input to the DNN model in Sec. 3.4.4 and Sec. 3.5.2.

3.4.4 VUI Response Identification

A segment of sound-induced MSS does not always contain a VUI response. It could originate from other sources such as phone calls. To single out the segments of interest, we observe that there are only a limited set of voice profiles available for VUIs, *e.g.*, 5 male and 5 female voices on Google Assistant. Thus, we can simply identify if the MSS contain a VUI response from a specific voice ID. We design a lightweight DNN model (Fig. 3.4), inspired by XVector [205], to address this problem.

Specifically, the model takes the aforementioned spectrogram feature as input. The output contains 11 classes, including 10 different voices from Google Assistant and a class for other sound sources. We first embed the frame level feature by using TDNN blocks [204], and then

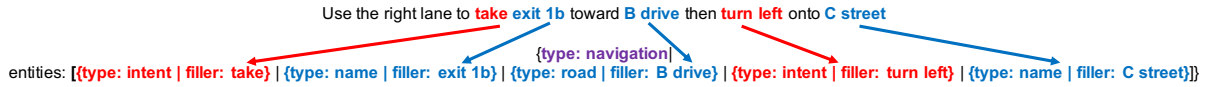


Figure 3.5. Example to convert the navigation voice text to private intents

apply an attentive statistics pooling layer [172] to convert the variable length frame-level features into a fixed-dimensional vector, referred to as deep speaker embedding in text-independent and variable-duration scenarios [162]. Finally, the deep speaker embedding is fed into an utterance-level feature extractor comprised of fully-connected hidden layers.

To train the VUI response identification model, we use an accelerometer data set collected on COTS smartphones, which consists of 10 hours of VUI responses for each voice profile, and 35 hours of accelerometer signals when playing Youtube Videos. Our benchmark experiments (Sec. 3.9.1) show that StealthyIMU achieves 3.41% EER for identifying the targeted VUI voice by using a lightweight model of less than 2 MB. Our experiments show that StealthyIMU achieves 3.41% EER for identifying the targeted VUI voice by using a lightweight model of less than 2 MB.

3.5 Inferring Privacy from a Single VUI Response

In this section, we discuss the problem formulation, DNN model design and training strategy to infer the private entity list from a single VUI response.

3.5.1 Problem Statement

Recognizing general speech from the MSS is a highly underdetermined problem, as the sensor only samples the vibration artifacts and at much lower sampling rate than microphones. However, two key observations enable StealthyIMU to circumvent the barrier. First, there are only a limited set of machine-rendered voice profiles and the user rarely switches between them. So the VUI responses almost always come from the same voice, making the speech recognition problem *user-dependent*. Second, the acoustic characteristics of VUI are more deterministic than natural human speech.

Inspired by recent research in speech and natural language processing, we propose to formulate the StealthyIMU attack as a Spoken Language Understanding (SLU) problem. SLU systems are widely used in VUIs which recognize the intents and meanings of speech directly, rather than word-by-word transcription. In comparison, StealthyIMU takes the MSS as input, and extracts a private entity list within the VUI response, as exemplified in Table 3.1. Since word-level recognition using MSS is error prone [32], StealthyIMU designs an end-to-end SLU DNN model to directly interpret the VUI responses.

DNN input, output, and ground-truth generation: The input into our SLU model is the feature map that has been obtained after the voice response identification and feature extraction (Sec. 3.4.3 and Sec. 3.4.4). The output is the text of private entity list, which comprises *i)* voice response type and *ii)* private entities. The machine-rendered VUI responses tend to follow a fixed format. The same type of responses often contain the same number and pattern of private entities, as shown in Table 3.1. Navigation voice is the most complicated type, where the sentence structure is consistent but the entity number varies. We thus take the navigation voice from Google Map as an example to explain the general steps of how we generate the *ground-truth* private intent lists.

First, by empirically analyzing the grammar of the real-world navigation voices in our data set, we summarize the private entities of interest as follows:

i) Intents, *i.e.*, the next driving intent. There are mainly 16 intents, *i.e.*, “turn left”, “turn right”, “take the next left”, “take the next right”, “slight left”, “slight right”, “keep left”, “keep right”, “continue”, “stay”, “take” (highway and exit), “make a u-turn”, “merge”, “follow sign”, and “arrive at destination”.

ii) Name, *i.e.*, road, highway and exit names.

Second, we take the grammar into consideration. Fig. 3.5 shows an example of converting the navigation text to private intent. We first add the voice response type, *i.e.*, navigation. Then, we identify the private entities including both intent and name. All of the entities except voice response type are in the text and are added into the private intents in natural spoken order. This

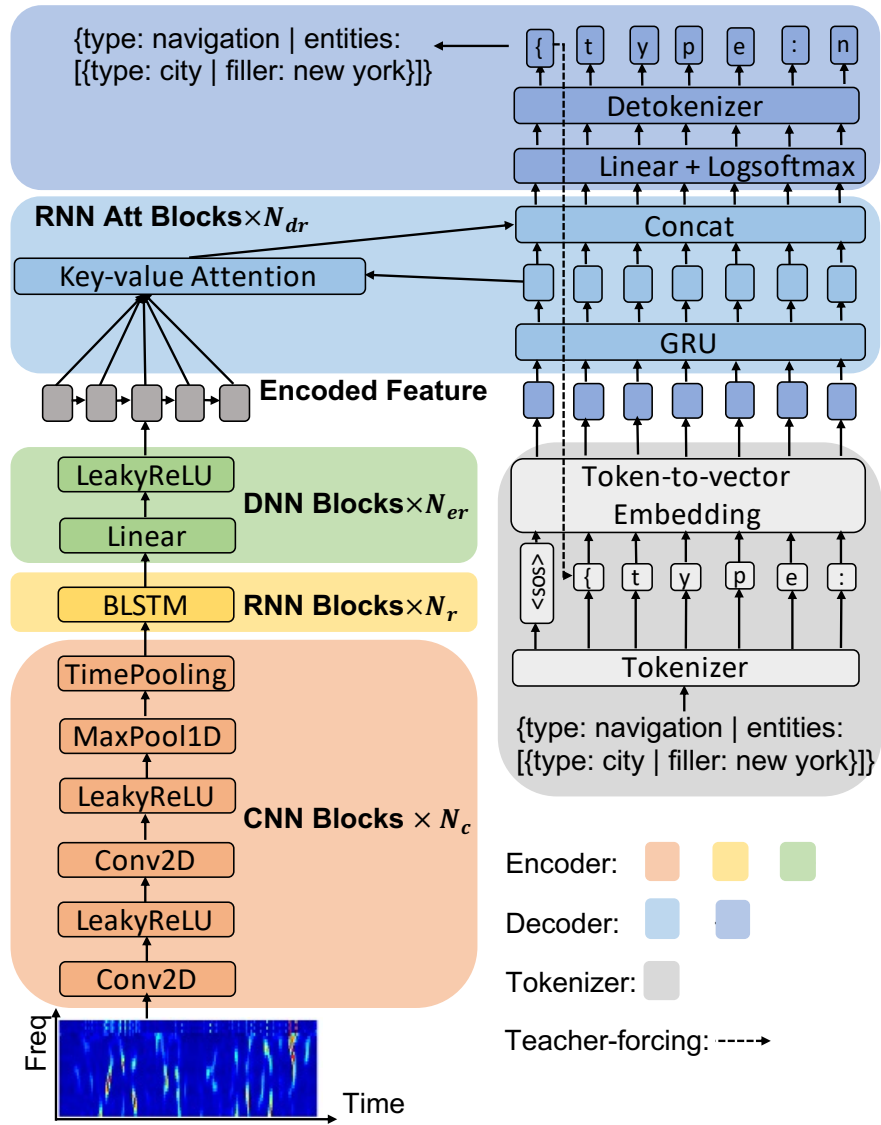


Figure 3.6. Model Architecture

workflow of generating the ground-truth also gives the model prior knowledge about the sentence structure and potential entities' locations within the sentence.

3.5.2 Model Design

Our end-to-end SLU DNN model is a sequence-to-sequence model, which comprises 3 components, *i.e.*, Tokenizer, Encoder, and Decoder. As shown in Fig. 3.6, the input of the model is represented as a sequence of the frequency spectrum, $\mathbf{x} = \{x_1, x_2, \dots, x_t\}$, where t is the temporal length of the spectrogram of the voice-associated MSS. The output is a sequence of

tokens corresponding to the private entity list, $\mathbf{y} = \{y_1, y_2, \dots, y_s\}$, where s determines the length of the list.

Tokenizer: To optimize our model towards the ground truth, we need to quantitatively encode the text of the private entity list. We use a tokenizer to break the raw private entity list into tokens, which can be word-level, subword-level, or character-level. Since the lexicon of the private entity is large (> 4000), using word-level token will increase model complexity. Further, the relationship between the time-frequency (T-F) spectrogram and the word-level tokens is hard to learn for a DNN model. In comparison, subword-level and character-level tokens are closer to the *phoneme*, the basic unit of human voice, which has proven to be suitable for speech processing [280]. To reduce the model size and improve the recognition rate, we choose to create character-level tokens using the Google SentencePiece Tokenizer [123]. The corresponding detokenizer will be used in the inference stage to recover the private entity list from the token sequence.

Encoder: The input sequence \mathbf{x} is fed into the encoder to embed the T-F spectrogram feature. Fig. 3.6 shows the structure of our encoder, which contains N_c CNN blocks, N_r RNN blocks, and N_{er} DNN blocks. We use the “TimePooling” layer in each CNN block to reduce the length of the input sequence by half. Except for the TimePooling layer, the encoder does not change the length of the input sequence, resulting in hidden state length $t/(2^{N_c})$.

Decoder: Our decoder model contains N_{dr} RNN attention blocks (Fig. 3.6). The embedded feature of the last token is first fed into a GRU layer. To leverage the entire encoded hidden state from the encoder, we use a key-value self-attention mechanism [234], which takes the encoder hidden state as the keys and values and the decoder hidden state as the queries for the self-attention layer. Finally, the hidden state with attention is concatenated with the original decoder hidden state as the output of the RNN attention block along with the linear and softmax layer to recognize the output token sequence $\hat{\mathbf{y}}$.

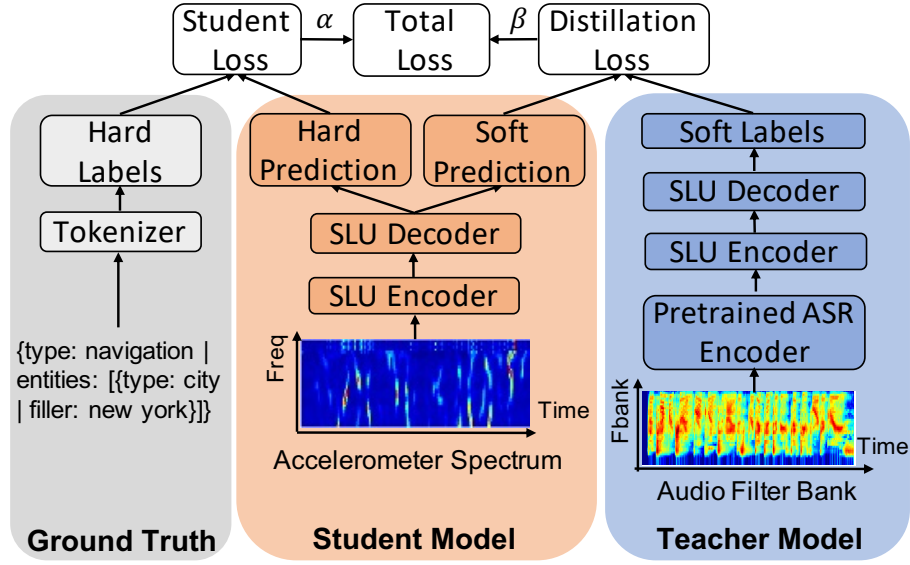


Figure 3.7. Knowledge Distillation Training Strategy. StealthyIMU proposes to distill the knowledge from the speech model to guide the training of the motion sensor model. The “Student Model” corresponds to the model trained by MSS as shown in Fig. 3.5. The “Teacher Model” has the similar architecture with “Student Model”, but is trained by speech signals.

3.5.3 Training Strategy

Teacher-forcing: We apply the teacher-forcing training strategy to train the decoder, which uses the ground truth token from a prior output token as input to the decoder, to prevent error accumulation. We adopt the negative log likelihood loss function

$$l_s = \max_{\theta} \sum_i \log P(y_i | \mathbf{x}, y_{<i}^*; \theta) \quad (3.1)$$

where $y_{<i}^*$ is the ground truth of the previous tokens.

In the inference process, we replace the teacher-forcing by using the last predicted token as the input of the decoder. We use beam search [219] with beamwidth 40 to generate the sequences of tokens with the highest probability globally.

Knowledge Distillation from Speech: To enhance the SLU, we design a teacher-student cross-modal knowledge distillation DNN, which distills the knowledge from the speech model to guide the training of the motion sensor model. The rationale behind this design is two-fold. First,

the speech model is more feature rich and hence accurate than the motion sensor model. Thus it can facilitate the motion sensor model to learn reasonable attention across an entire segment of MSS. Second, due to the massive amount of training data, the speech model is more general than the motion sensor model, which helps overcome the lack of training data for the MSS.

As shown in Fig. 3.7, we simultaneously collect the speech and accelerometer signals associated with the voice responses, through the microphone and accelerometer, respectively. We first train the teacher SLU model by using the speech signals as input. Specifically, we use a *pretrained* ASR encoder, trained by the LibriSpeech dataset [176], as a basic encoder. Then, an SLU encoder and decoder, similar to the model in Sec. 3.5.2, are used to train this speech-based teacher model.

The student SLU model uses the corresponding accelerometer signals as input and has been introduced in (Sec. 3.5.2). During the training of the student model, we freeze the teacher model layers and achieve cross-modal knowledge distillation by narrowing the distance between the output distribution of the two models. We choose to distill the knowledge from the decoder output because the outputs of the corresponding speech signals and accelerometer signals are expected to be the same. We use the Kullback–Leibler (KL) divergence loss [105] as our knowledge distillation loss function $l_d = \frac{1}{s} \sum_{i=1}^s KL(P_S^i, P_T^i)$, where s is the output sequence length, P_S^i and P_T^i are the output distribution of the student model and teacher model respectively. Finally, the combined loss function is: $l_{total} = \alpha l_s + (1 - \alpha) l_d$. We set $\alpha = 1$ in the first few epochs, and then gradually reduce α to increase the weight of knowledge distillation loss and help convergence.

We note that the teacher-student training strategy is only used in the offline training process, and does not increase the model size and computation overhead.

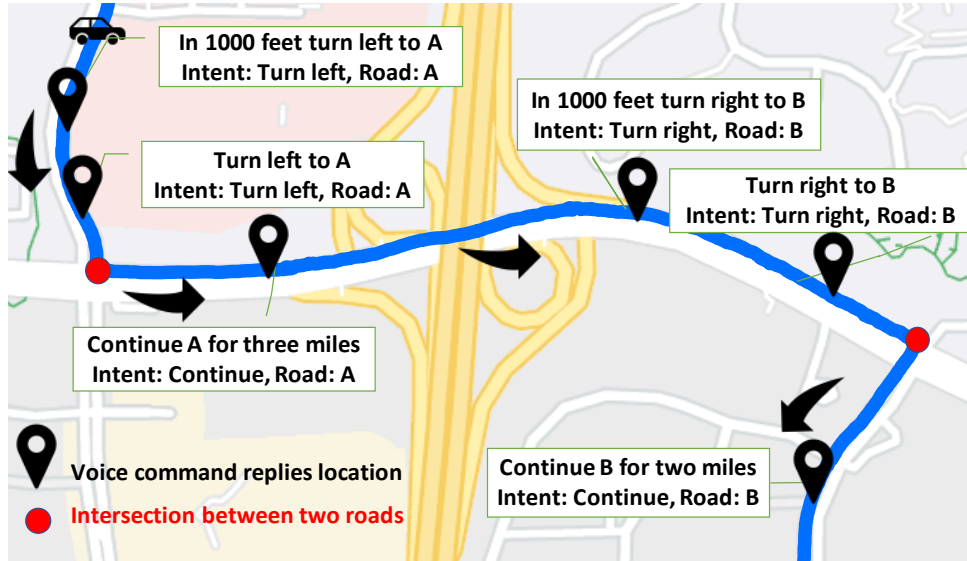


Figure 3.8. Trace and commands in navigation

3.6 Extracting Permission-protected Privacy

In this section, we explain how to extract permission-protected private information by extracting and combining the private intents from single or multiple VUI responses. Based on the availability of contextual information, we categorize the attack model into three different scenarios.

3.6.1 One-Time Stealing

One-time stealing means that StealthyIMU only takes a single VUI response as the input to extract privacy information. Voice commands (e.g., set/check calendars and alarms, make calls and send messages, and search for stock, sports, and music) are usually resolved in a single response, without any contextual information. The private intents inferred from the response directly correspond to the private permissions. Such one-time stealing achieves a relatively lower success rate than the other two scenarios since it fully depends on the SLU from a single voice response. In Sec. 3.9.2, we demonstrate that StealthyIMU achieves an average 85.28% success rate across 12 types of voice commands and is also able to identify other 11 types of voice commands without explicit private information.

3.6.2 Short-Term Contextual Inference

Short-term contextual inference represents the scenario where StealthyIMU can infer the private information by using multiple consecutive (≥ 2) VUI responses. One of the most frequently-used cases is the navigation app. To direct the user towards the destination, the navigation app uses a sequence of navigation voices during navigation. Typically, for each turning point, there exists 1 or few navigation voices, which provide contextual information continuously, allowing StealthyIMU to track the user’s location and even recover the GPS trace. In this section, we show how StealthyIMU extracts a sequence of private intents from a navigation process, combined with the city-scale road map from OpenStreetMap [95], to recover the GPS trace. Note that 2 consecutive VUI responses are sufficient to infer the GPS location of the victim. To recover one GPS route trace, the duration of eavesdropping depends on the navigation time of this route, which ranges from a few minutes to a few hours with 2 \sim 50 VUI responses.

Fig. 3.8 shows an example. Each “black” mark represents a navigation voice that happens along the route. StealthyIMU can already extract both the “intent” and “road name” of each navigation voice following the workflows in Sec. 3.5. The key problem here is to recover the “blue” trace. To this end, we design a GPS trace recovery algorithm (see Algorithm 1). Our basic idea is to first identify the intersection between two roads (red dots in Fig. 3.8) by checking the ordered list of road names that we extracted. Then, we can directly connect each two consecutive intersection points by using the shortest path algorithm on the road map.

To enable private information extraction, the GPS trace recovery algorithm needs to address two challenges uniquely related to StealthyIMU. First, the intent recognition is not perfect while the navigation app itself may also produce occasional incorrect navigation voices due to poor GPS signals. Therefore, sometimes we cannot find the intersection between two consecutive roads because of a missing or wrongly recognized road. To improve robustness, we design a *correction mechanism* to interpolate/skip a road name to prevent the false recognition and enable the algorithm to find as many intersections as possible (Line 3 ~ 10). Further, we

Algorithm 1: Trace Recovery Algorithm

Input: N commands stolen in whole navigation
Output: Complete trace

- 1 Extract (intent, road) pairs that appear at least twice in succession;
- 2 Preprocess the pairs by cleaning, integrating, and sorting;
- 3 **foreach** *road in road list* **do**
- 4 **if** *two disjoint roads have shared road* **then**
- 5 Interpolate the shared road to the road list;
- 6 **end**
- 7 **if** *two roads intersect by skipping other roads in between* **then**
- 8 Remove the skipped roads in the road list;
- 9 **end**
- 10 **end**
- 11 Find all intersection points for the roads in order;
- 12 Connect the point sequence to recover the trace with the shortest path;
- 13 **foreach** (*intent, road, point*) *in trace* **do**
- 14 **if** *first intersection point* **then**
- 15 Infer the direction of the initial path with the intent;
- 16 Estimate the starting point based on the time interval.
- 17 **end**
- 18 **if** *last intersection point* **then**
- 19 Infer the direction of the final destination with the intent;
- 20 Estimate the destination based on the time interval.
- 21 **end**
- 22 Check the direction in the shortest path with the intent;
- 23 **if** *direction in the shortest path is against the intent* **then**
- 24 Take the opposite direction of the same road;
- 25 **end**
- 26 **end**

observe that for each “turning” intent, the navigation app repeats the navigation voice twice, before the turn and right at the turn. We leverage this redundancy to double-check the road name.

Second, our method can not recover the source or destination points for the route due to a lack of corresponding intersections. To overcome this problem, we utilize the intent information to find the names of the first and last road segments. Then, we use the time interval between the first/last navigation voice and the first/last turn navigation voice to estimate the starting point and destination (Line 14 ~ 21). Besides, the intents of each navigation voice can also help check whether the shortest path follows the correct intent. If they are different, we will add another point after the turn intent to the intersection list to correct the GPS route (Line 22 ~ 25).

3.6.3 Long-Term Monitoring

Long-term Monitoring represents the scenario where the users repeat the same type of voice commands in a few days, like check weather, air quality and reminder, etc., while *StealthyIMU* runs the intent extraction workflow continuously to steal privacy. Although Sec. 3.6.1 has shown that *StealthyIMU* can extract the private information with a single VUI response, the long-term monitoring aims to accumulate its knowledge over time and improve the accuracy of privacy extraction. Specifically, we first use the SLU model to detect the voice command type (Sec. 3.5.2). Then, we calculate the probabilities of the top- n private entity list output. Since each voice response is a single individual event, the final success rate of an attack is the probability of multiple independent events in the long run.

For a more clear exposition, we take *city name extraction* as an example. When asked about the weather, the VUI will report the weather along with the city name. For a single VUI response, *StealthyIMU* takes 120 US cities with more than 200,000 population each as the potential city list. Then, for each potential city name, *StealthyIMU* takes the specific MSS of a single VUI response y_0 as the input and uses the DNN model in Sec. 3.5 to calculate the probability $P(x|y_0)$ of the output token sequence of that city, where x is the token sequence of that potential city. With multiple such queries over a few days, *StealthyIMU* calculates the probability of each potential city as $P(x|y) = 1 - \prod_1^K (1 - P(x|y_i))$, where x is still the token sequence of that potential city, K is the number of VUI responses; y_i and y denote the MSS signals of i -th VUI response and the set of K MSS signals respectively. *StealthyIMU* finally estimates the victim’s city name by selecting the result with the highest probability to further improve the attacking success rate of *StealthyIMU*.

Another example is the *home address extraction*. Google Map always utters “Welcome home” when reaching the user’s home as the destination. As discussed in Sec. 3.5, *StealthyIMU* first extract the “road name” and “intent” from each navigation voice. Once *StealthyIMU* identifies that the “intent” for a specific navigation voice is “Welcome home”, *StealthyIMU* runs

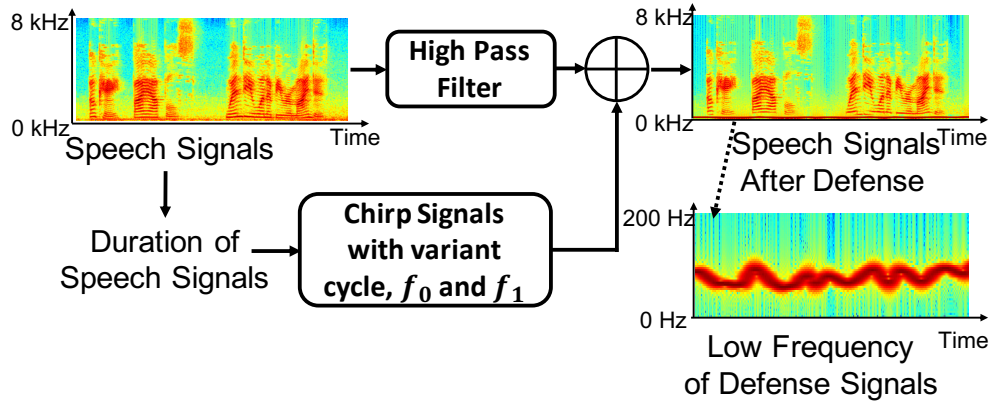


Figure 3.9. Pipeline of the speech predistortion defense.

the GPS trace recovery (Sec. 3.6.2) for the previous consecutive VUI responses to locate the destination of the GPS trace. Then, with multiple navigation attempts that contain such an intent to be back “home”, StealthyIMU can calculate multiple potential home addresses. Finally, to improve the accuracy of home address extraction, we combine these potential home addresses by first removing the outliers and then calculating the centroid of the remaining GPS points. In Sec. 3.9.4, we evaluate these examples in the wild to show the threat of long-term monitoring.

3.7 Defense Against StealthyIMU

In this section, we propose two defense mechanisms that can thwart the StealthyIMU attack.

3.7.1 Predistortion of Speech Signals

Our first defense protects existing smartphones, whose highest MSS sampling rate is < 500 Hz [39]. Our key insight is that *modifying the low-frequency components of speech signals will significantly impact the MSS, without noticeably affecting the human perceivable speech quality.*

Figure 3.9 shows the pipeline of this approach. First, we use a high-pass filter to remove the low frequency components in the speech signals with a cut-off frequency f_c Hz.

We empirically choose $f_c = 350$ Hz based on the measurement in Sec. 3.9.7, to balance the protection of privacy and reduction of speech quality. Second, we distort the T-F spectrogram of the MSS by injecting a structured low-frequency interference signal through the smartphone’s audio interface, more specifically a chirp signal with varying cycle and starting/ending frequency. The cycle range of the chirp is set to 100 ~ 200 ms because we aim to distort the spectrogram of each phoneme which is the basic unit of voice pronunciation and typically lasts less than 200 ms [211]. The advantage of this approach is that it only needs the VUI app vendors, *e.g.*, Google, Amazon, Apple, to add a pre-distortion stage in their voice responses, without any hardware modification.

3.7.2 Redesigning the Permissions

Although existing smartphone firmware and driver limit the MSS sampling rate below 500 Hz, the achievable sampling rate of the motion sensor hardware itself can be much higher. For example, the InvenSense MPU 6500 motion sensor, used in Samsung S6 and iPhone 6S, supports up to 4000 Hz. If such high sampling capabilities are unleashed in the future, the above defense approach may fail. We thus highly recommend the smartphone OS to redefine the permissions of the motion sensor. Since most of the apps do not need extremely high sampling rates for motion sensors, the smartphone OS can discriminate the permission levels for different sampling rate limits, and report the potential threats to the users when an app requests access to high-rate MSS.

3.8 Dataset and Implementation

3.8.1 StealthyIMU Dataset

To evaluate StealthyIMU, we collect a new benchmark dataset that captures the smartphone MSS when the loudspeaker plays voice responses. Meanwhile, we record the corresponding voice commands in order to generate the ground truth labels and facilitate our model training

Table 3.2. Types of Voice Command and Dataset Scale.

Type	Example Voice Command	Privacy	#
Weather	What’s the weather today?	Location	12,527
Sun set&rise	What’s the sunset in Chennai	Location	1,505
AirCheck	AQI for San Francisco	Location	1,601
Clock	What time is it in London	Location	1,595
Reminders	Set a reminder to check my account	Todo	2,950
	Set a reminder for tomorrow morning	Time	2,140
Media Alarms	Set an alarm to go to fedex	Todo	2,630
	Set a music alarm at 8 PM	Time	2,350
Stock Updates	Stock price for Apple	Search	1,318
Calling	Call Sam	Contacts	1,120
Navigation	Navigate to Los Angeles	Location	1,570
Navigation App	/	GPS	7,885
Fun Tricks	What movies are playing?	Others	500
Sports Facts	What’s the news about the NFL?	Others	500
News	What’s the news about the covid?	Others	500
Calculations	What calculation can you do?	Others	500
Google Search	How tall is the Eiffel Tower?	Others	500
Youtube Music	Play music on Youtube Music	Others	500
Voice Mail	Call voicemail	Others	500
Youtube	Open Youtube	Others	500
Chrome	Open the Google Trends website	Others	500
Youtube TV	Play FS1 on Youtube TV	Others	500
Broadcast	Broadcast a message	Others	500
<i>Overall</i>			<i>44,691</i>

(Sec. 3.5.3). The trained model will be used later in our real-world attack evaluation.

Voice response data collection: We adopt a natural language voice command text dataset [198], and use Google Text-to-Speech to synthesize the corresponding speech along with the wake word preamble “Hey Google”. To collect the dataset automatically, we use an additional device, *i.e.*, a Macbook Pro laptop, to playback the voice commands just like a human user. Meanwhile, a victim’s smartphone hears and responds to the voice commands through its loudspeaker while the StealthyIMU app captures the accelerometer signals. We use an open-source test automation framework, called Appium [235], to control the laptop and smartphone

and streamline the process. During the data collection, the smartphone is set under two different real-world scenarios, *i.e.*, on a table and a car phone mount, respectively.

As shown in Tab. 3.2, we collected 23 types of VUI responses by asking Google Assistant “What can you do?”. Among these, 12 types have fixed structures and explicit permission-protected private information while the other 11 types are unstructured with arbitrary contents. We place these voice commands into 5 categories based on the targeted permissions.

- “Access coarse location”: The “Weather”, “Sunset & Sunrise”, “AirCheck” and “Clock” related responses are to steal the private information from “Access coarse location”, *i.e.*, city name of the location. We collect a large dataset for “Weather” with 12,527 VUI responses and relatively small datasets for “Sunset & Sunrise”, “AirCheck” and “Clock” with 1,595, 1,601, and 1,595 VUI responses respectively from 120 US cities with more than 200,000 population each [198]. To generate the training data, we first use the Fake GPS app to change the smartphone GPS to the city, and then playback the these types of voice commands.

- “Read calendar”: The set/check calendar/reminder/alarm related responses are used to steal the private information from “Read calendar” permission. We collect 5,031 and 4,980 different reminder and alarm commands respectively with 300 frequently-used TODO entities and 300 TIME entities. Since we know the corresponding voice command for each response, we can directly extract the TODO entity and TIME entity and generate the ground-truth entity list.

- “Stock Search”: The stock update check command responses are used to steal the private information from “Search history” permission. We collect 1,318 different stock update commands with top 150 companies with the largest capital in NASDAQ.

- “Contacts”: The hands-free calling command responses are used to steal the “Contacts” permission. We collect 1,120 different calling commands with 200 frequently-used name entities.

- “Others”: There exists other 11 types voice commands which has unstructured VUI responses with arbitrary contents. StealthyIMU can not easily extract the explicit private information from these VUI responses. Therefore, we label these VUI responses with “Null” labels.

Finally, we label the ground truth entity list of these voice responses by listening to the recordings. Note that the VUI responses are highly diverse, totaling more than 150 hours and containing more than 4000 frequently-used words. The VUI responses are collected over more than one year.

Navigation app voice data collecting: To collect the navigation app voice with ground truth GPS traces, we first use OpenStreetMap to open a city map. We randomly select two points on the map separated by > 6 km, and use Google Map to generate the navigation route. We export the navigation route to the smartphone, and start the Google Map navigation app to follow this route. Then we use “MAPS TO GPX” to convert the Google Map navigation route to a GPX file which contains the GPS point list of this route. The GPX file is imported to a “Mock Location” app to follow this route by using fake GPS. Finally, Google Map is able to navigate the smartphone from the start point to the destination by checking the fake GPS positions. And again, we use Appium to automate this data collecting process.

To annotate the ground truth of the navigation voice, we design an app to listen to each navigation voice and annotate it based on the potential road name and intent from map and navigation trace. We collect this dataset from two cities.

- City A: a typical city with a population of > 1.5 million and area approximately 800 km^2 . We collect 300 routes with 5,091 navigation voices which contain more than 1,800 km mileage, 419 roads (95% coverage) and 7 highways (100% coverage) and 55 highway (92% coverage) exits.

- City B: a metropolitan city with a population of > 5 million and area of 800 km^2 . We collect 150 routes with 2794 navigation voices which contain more than 900 km mileage, 559 roads (37% coverage) and 8 highways (100% coverage) and 42 highway (84% coverage) exits.

3.8.2 Implementation

DNN Implementation: We implement the voice response identification DNN model (Figure 3.4) and voice response SLU model (Figure 3.6) in PyTorch. For training, we use the

NewBob learning rate strategy [259] at a $3e - 4$ initial learning rate followed by annealing. The default training batch size is 8, and the number of epochs is 30. The pretrained models are then converted into Just-In-Time (JIT) PyTorch model running on smartphones.

Attack Implementation We implement two versions of StealthyIMU, one running entirely on an Android phone and the other facilitated by a cloud server. Our app can choose to *i)* detect voice on the smartphone or stream the signals continuously to the cloud; *ii)* save the signals in the memory or local storage; *iii)* execute the attack either on device or in cloud. We evaluate the system overhead and permission risks of these different attack surfaces in Sec. 3.9.6.

3.9 Evaluation

The evaluation of StealthyIMU consists of four parts. First, from Sec. 3.9.1 to Sec. 3.9.4, we use the default large-scale dataset discussed in Sec. 3.8.1 for training. And then, we test StealthyIMU by collecting specific testing dataset for each attacking mode. Second, in Sec. 3.9.5, we collect additional testing datasets with different real-world scenarios and evaluate StealthyIMU across different factors to see whether the model can generalize to new scenarios. Finally, we evaluate the system overhead and defense mechanisms of StealthyIMU in Sec. 3.9.6 and Sec. 3.9.7 respectively.

3.9.1 DNN Model Ablation Study

We first conduct an ablation study to evaluate the accuracy and efficiency of the Stealthy-IMU DNN models, *i.e.*, VUI response identification (Sec. 3.4) and VUI response private entity recognition (Sec. 3.5). In this study, we use the dataset containing 23 types of VUI responses, as discussed in Sec. 3.8.1. We mixed all of the VUI responses and train a single model to extract the private information from a VUI response across different types of VUI responses. We split the dataset into training, validation, and testing set with 8 : 1 : 1 ratio.

VUI Response Identification: We use EER, a well-known speaker identification metric, at which both acceptance and rejection errors are equal [171], to evaluate our VUI response

Table 3.3. Impacts of Training Datasets.

# of Type	# of Training	TER	SEER	SER
1	1,000	0.00%	28.86%	48.53%
1	8,000	0.00%	8.71%	14.81%
3	15,000	0.00%	7.49%	12.73%
23	35,000	0.00%	8.46%	14.45%

Table 3.4. VUI response identification ablation study.

DNN model				Size	EER
N	Channel	Kernel	FCN		
5	[(512*4),1500]	[5,3,3,1,1]	512	4.5MB	2.43%
5	[(512*4),1500]	[5,3,3,1,1]	16	2.7MB	3.33%
3	[(512*2),1500]	[5,3,1]	16	1.7MB	3.41%
3	[(512*2),1500]	[1,2,1]	16	1.4MB	3.81%
3	[(128*2),256]	[1,2,1]	16	79.6KB	4.95%
3	[(32*2),64]	[1,2,1]	16	6.5KB	8.60%

identification method. Here we aim to identify the 10 voices (5 male and 5 female) on Google Assistant, along with a class for other non-VUI sound sources. Table 3.4 summarizes the results. Since a small model degrades the performance significantly while a large model consumes more system overhead (Sec. 3.9.6), we empirically choose a medium size model of 1.7 MB model as our default model. As shown in Fig. 3.10, *our DNN model is able to accurately identify the different voice sources, with a maximum EER of less than 6.73% (3.41% on average).*

VUI Response Private Entity Recognition: Unlike speech recognition metrics, such as Word Error Rate (WER) and Character Error Rate (CER), a more reasonable way to evaluate the private entity recognition within VUI response is to check whether StealthyIMU can understand the whole entity correctly. Therefore, we use the following 3 metrics: *i) Type Error Rate (TER)* which indicates whether StealthyIMU can recognize the type of the VUI response, *i.e.*, weather, calendar and navigation; *ii) Single Entity Error Rate (SEER)*. There may exist more than one entities in a single VUI response. SEER mainly takes a single entity as a unit to report the error

Table 3.5. VUI Response Private Entity Recognition Ablation Study. N_t : # of tokens in tokenizer; N_c : # of CNN blocks; K_c : kernel size; C_c : CNN channels; N_r : # of RNN layers; H_r : # of RNN neurons; N_{ed} : # of DNN blocks; O_{ed} : # of DNN neurons; N_d : # of decoder layers; H_d : hidden size of decoder; A_d : attention dim of decoder; E_d : # of output neurons.

Token	Encoder (CDRNN)										Decoder (GRU + Key Value Attention)				Model Size	TER	SEER	SER
	CNN Blocks			RNN Blocks			DNN Blocks				Key Value Attention							
	N_c	K_c	C_c	N_r	H_r	N_{ed}	O_{ed}	N_d	H_d	A_d	E_d							
50	2	3	(128, 256)	4	1024	2	512	3	512	512	128	128	106.4 MB	0.00%	13.22%	21.31%		
50	2	3	(64, 128)	4	256	2	256	3	256	256	64	64	9.1 MB	0.00%	9.65%	15.32%		
100	2	3	(128, 256)	4	256	2	256	3	256	256	64	64	11.9 MB	0.00%	11.68%	20.01%		
50	2	3	(64, 128)	4	128	2	256	3	256	256	64	64	4.2 MB	0.00%	10.85%	18.99%		
50	2	3	(64, 128)	4	128	2	128	3	256	256	64	64	4.1 MB	0.00%	11.73%	19.96%		
50	2	3	(64, 128)	2	128	2	128	3	256	256	64	64	3.3 MB	0.00%	17.91%	31.27%		
50	2	3	(64, 128)	4	128	2	128	3	128	128	64	64	3.0 MB	0.00%	26.00%	46.40%		
50	1	3	(64, 128)	4	128	2	128	3	256	256	64	64	3.9 MB	0.00%	8.46%	14.45%		
50	1	3	(64, 128)	4	128	1	128	3	256	256	64	64	3.8 MB	0.00%	13.80%	24.60%		
50	0	/	/	4	128	2	128	3	256	256	64	64	2.9 MB	0.00%	18.46%	34.85%		

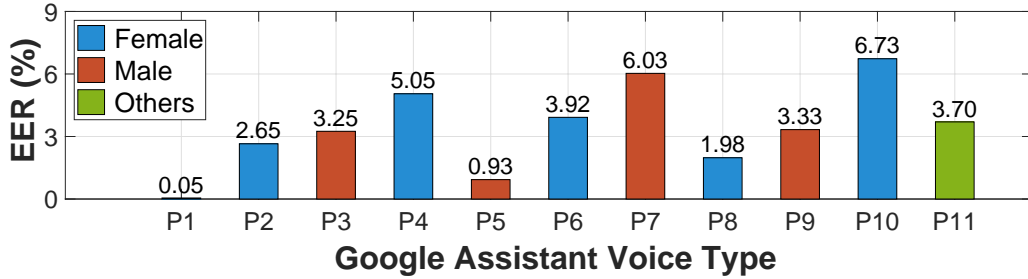


Figure 3.10. Google Assistant Voice Identification.

Table 3.6. SLU model training approach.

	Model Size	TER	SEER	SER
ASR+NLU	26.5 MB	0%	46.45%	77.91%
SLU	3.8 MB	0%	14.76%	25.16%
SLU+KD	3.8 MB	0%	8.46%	14.45%

rate; *iii*) *Sentence Error Rate (SER)* which counts an error-free recognition only if all the entities in a single VUI response are identified correctly.

We compare the performance of 3 different models to resolve the VUI response private entity recognition task, 1) “ASR+NLU”: a cascaded approach with an ASR model to recognize the text from the MSS and an NLU model to extract the entity list from the text (Sec. 3.2); 2) “SLU”: StealthyIMU’s SLU model without speech model knowledge distillation; 3) “SLU+KD”: StealthyIMU’s SLU model with speech model knowledge distillation. Table 3.6 summarizes the results. The key problem of “ASR + NLU” is that the ASR model can only achieve 68.16% WER, which makes it hard for the follow-on NLU to extract the correct private entities. In contrast, SLU model directly optimizes the target of extracting the private entities from MSS. With the help of knowledge distillation from the teacher model, it becomes even more accurate. *On average, “SLU+KD” StealthyIMU achieves a 0.00% TER, 8.46% SEER, and 14.45% SER with a 3.9 MB model size.*

Next, we conduct a ablation study on the StealthyIMU SLU model by modifying the parameters of the Tokenizer, Encoder and Decoder, as shown in Fig. 3.5. We find that: *(i)* A subword-level tokenizer ($N_t = 50$) does not improve performance when the model size is

small since it needs more parameters to learn the subword level characteristics. (ii) RNN is the most important block in our encoder. If we reduce the layers of the RNN blocks, the model performance will degrade significantly. (iii) The attention and hidden layer size impact the performance of the decoder. Based on these observations, we choose the model with 3.9 MB size as default model for VUI response private entity recognition (highlight in Table 3.5).

Finally, we conduct experiments to evaluate the impacts of training and testing dataset. In Tab. 3.3, we evaluate StealthyIMU in three different settings, *i.e.*, training and testing StealthyIMU by using (i) single type of voice command; (ii) 3 types of voice command with large-scale dataset (Weather, Reminders and Navigation App in Tab. 3.2); (iii) all of 23 types of voice command. We find that (i) training and testing StealthyIMU with 3 types of voice commands achieve similar results as training/testing with 23 types of voice commands. (ii) if we use a single type of voice command to train the model, StealthyIMU requires a large-scale dataset to achieve high performance. In comparison, if we train the model with multiple types of voice commands, the scale of a specific type of voice command can be small. We believe that this is because a larger dataset with more diverse voice commands empowers the model to better generalize, so it is able to recognize the private entities from different formats of VUI responses.

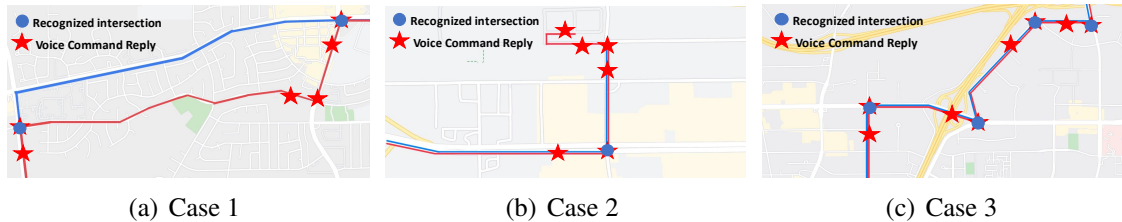


Figure 3.11. GPS trace recovery examples

3.9.2 One-time Stealing

In the one-time stealing experiments, we use the same training and testing setting as in Sec. 3.9.1. A single model is trained to extract the private entities from a single VUI response across 23 types of voice commands. We select the model and parameters with the highest

Table 3.7. One-time stealing results.

Type	Private Entity	TER	SEER	SER
Weather	Location	0.00%	3.05%	5.38%
Sunset & Sunrise	Location	0.00%	7.14%	13.97%
AirCheck	Location	0.00%	1.49%	3.33%
Clock	Location	0.00%	2.13%	3.18%
Reminders	Todo	0.00%	7.36%	13.21%
	Time	0.00%	15.25%	29.94%
Media Alarms	Todo	0.00%	8.24%	15.29%
	Time	0.00%	14.11%	26.50%
Stock Updates	Search	0.00%	7.33%	11.54%
Hands Free Calling	Contacts	0.00%	12.18%	22.64%
Navigation	Location	0.00%	2.19%	3.89%
Navigation App	GPS	0.00%	16.2%	26.79%
Others	/	0.31%	0.31%	0.31%
<i>Overall</i>		<i>0.00%</i>	<i>8.06%</i>	<i>14.45%</i>

performance in Sec. 3.9.1. For the rest of the evaluation, we use this general DNN model as our default model to extract the private entities without training any new DNN models.

Tab. 3.7 shows that *a single DNN model can accomplish one-time stealing with an average success rate of 85.28% even without contextual information*. Moreover, StealthyIMU achieve higher performance for structured VUI responses, *i.e.*, “Weather”, “Sunset & Sunrise”, “AirCheck”, “Clock”, “Stock Update” *etc.* Although voice commands like “Reminders”, “Media Alarm”, “Hands Free Calling” and “Navigation App” has complicated response formats resulting in relatively lower success rate for one-time stealing, they can still achieve > 70% success rate. We believe that with more training data, they will achieve even higher success rate.

3.9.3 Short-term Contextual Inference

We now proceed to verify the short-term continuous navigation voice and the GPS trace recovery algorithm. We collect a testing dataset in City A and make sure there is no overlap with the training dataset in Sec. 3.9.2. Then, we use the DNN model trained in Sec. 3.9.2 to extract the private entities of each VUI response from the testing dataset. Finally, we apply the

GPS trace recovery algorithm to the extracted private entities from consecutive VUI responses and compare the result with the ground truth GPS trace. To better illustrate our approach, we first show three examples of GPS trace recovery in Fig. 3.11. Then we evaluate the effectiveness of our route correction mechanism, and present the results of the end point localization.

In Case 1 of Fig. 3.11, our model only identifies 2 out of 3 intersections, and the shortest path between these two points deviates from the true path. If we could accurately identify the 3 intersections, then the path would be correct. Case 2 shows that if the final destination is on an unnamed road, our algorithm can only assign the last named road as the end point. Case 3 shows a perfect example of GPS trace recovery, where all the intersection points are accurately identified. These examples demonstrate that our recovery algorithm needs to accurately recognize the road names, based on which it can generate the final route through the shortest path between consecutive intersections on the map.

In order to quantify the actual performance of the algorithm, we perform a real-world field test in City A during daily driving and recover over 500-mile real routes. During the navigation process, we simultaneously record the results with and without the route correction mechanism. In terms of route coverage, the average route coverage rate without route correction is 52.46%, and increases to 62.87% after correction. Moreover, as shown in Fig. 3.12, 80% of the GPS distance errors are within 30 m after correction, in contrast to 63% before correction. In addition, we count the distance deviation between the recovered destination and the ground truth. According to our statistics, the min/average/max deviation is 3 m/133 m/420 m. The average deviation is large, because sometimes the destination is on an unnamed road, and we can only use the end point of the last named road as a replacement, as shown in Case 2 in Fig. 3.11.

3.9.4 Long-term Monitoring

Our long-term monitoring experiments focus on city name and home address identification.

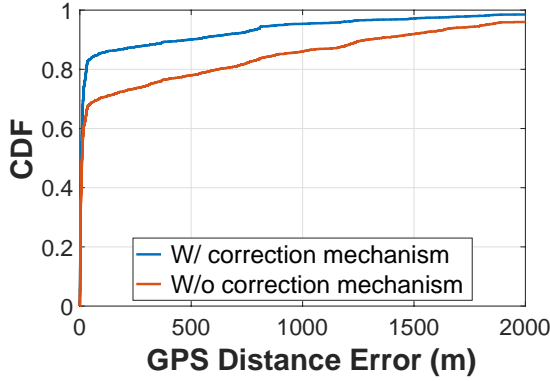


Figure 3.12. Short-term GPS distance error

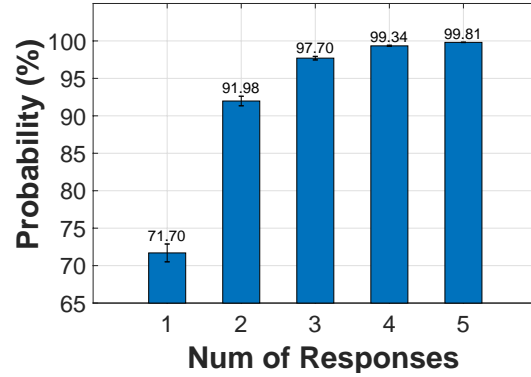


Figure 3.13. Long-term attack results

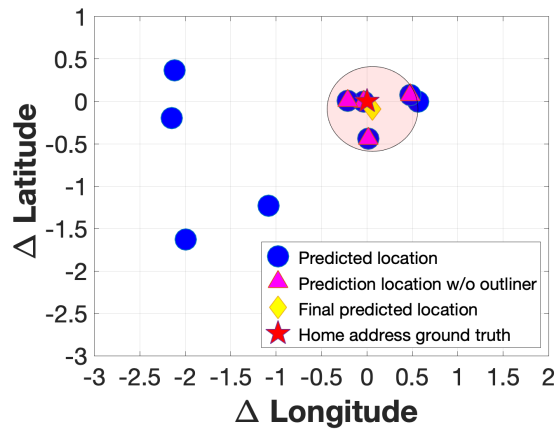


Figure 3.14. Long-term home address inference.

For the city name extraction, we collect a testing dataset including “Weather”, “Sunset & Sunrise”, “AirCheck”, “Clock” and “Navigation” voice commands over one month. Then, we perform private entity recognition on each VUI response in the testing set by using the default DNN model in Sec. 3.9.2. Then, we combine the results from multiple VUI responses by using the methods discussed in Sec. 3.6.3 to demonstrate the scenario where StealthyIMU keeps monitoring the VUI responses over a few days. As shown in Fig. 3.13, as the victim repeats the inquiries over time, the probability that StealthyIMU correctly identifies the city name increases from 70% (1 inquiry) to above 98% (3+ inquiries).

For the home address identification, we collect a testing dataset containing 10-day routes back to home. Then, we follow the steps in Sec. 3.9.3 to recover the GPS trace of these 10 routes, and mark the destination of the recovered GPS trace as the home address. Fig. 3.14 visualizes

Table 3.8. Navigation voice for different cities.

	Routes	Roads	Seg	SEER	SER
City A	300	419/441	5091	9.31%	19.77%
City B	150	559/1511	2794	16.25%	32.41%

Table 3.9. Generalization across different smartphones and sampling rate

Phone	OS Version	Sampling Rate	SER	SEER
OnePlus	Android 11	440 Hz	13.85%	8.99%
Samsung S8	Android 9	400 Hz	13.39%	8.65%
Samsung S8	Android 9	200 Hz	62.69%	38.30%
Samsung S8	Android 9	100 Hz	84.01%	52.50%
Huawei Mate 20	Android 9	500 Hz	17.20%	10.07%
Samsung S7	Android 8	420 Hz	15.57%	9.17%

the results of home address inference. The ● represent the results of repeated attempts on home address inference, and ▲ represent the remaining GPS points after removing outliers. Finally, we use the centroid of these remaining GPS points as the address estimation. In this example, the deviation between the StealthyIMU estimation and the ground truth home destination is only 11 meters by combining 10 attempts of address estimation.

3.9.5 Generalization

In this section, we evaluate StealthyIMU generalization across different factors to see whether the model can work on a new scenario. By default, we use the model trained in Sec. 3.9.2 as our baseline model, and collect additional testing datasets with different real-world scenarios.

Generalization across different sound volumes, smartphone models and sampling rate settings: Loudspeaker volume determines the vibration magnitude of the smartphone during a VUI response, and hence affects the SNR of the MSS. We evaluate 5 different volume levels as

shown in Fig. 3.15(a). When the volume exceeds the default level of our experiments (80%), StealthyIMU achieves > 10 dB SNR, and the SER (SEER) is smaller than 10% (15%). When the volume falls below 40%, the SER (SEER) is $> 80%$ (95%).

We further run StealthyIMU on 5 different smartphones, equipped with IMU sensors from different vendors and different sampling rate (100 ~ 500 Hz). As shown in Fig. 3.15(a), smartphones released within 5 years, *i.e.*, OnePlus, Samsung S7, S8, Huawei Mate20, all support > 400 Hz sampling rate which suffices for the StealthyIMU attack. Although the layout of the IMU sensor on the smartphone motherboard may vary, the signal SNR is similar among these smartphones. When the volume level is higher than 80%, StealthyIMU DNN model achieves an 15.39% SEER and 9.30% SER on average when trained with the mixed dataset collected from these three smartphones. On the other hand, the StealthyIMU attack becomes less effective when the MSS sampling rate falls below 200 Hz.

Generalization under different motion artifacts: We measure the impacts of interference from 3 motion artifacts: holding the smartphone in hand, walking, and driving with the smartphone on a car phone mount. Most of the noise introduced by human motion, like holding and walking, is low-frequency [119]. Thus, after applying a high-pass filter, the signal SNR remains high, while the SEER and SER are 10.3% and 16.8%—only slightly higher than the static case.

In the driving scenario, the main vibration interference comes from the car’s body. The dominant noise frequency can be estimated by $f_n(t) = \frac{R(t)}{60} * (N_c/2)$, where $R(t)$ is the rotational speed of engine in revolutions per minute (RPM) and N_c is the cylinder number of the engine. For example, a car with a 4-cylinder engine generate the vibration noise of less than 100 Hz even at 3000 RPM (the RPM for normal driving is 1500 ~ 2500 [37]). Therefore, to isolate the driving noise, we simply apply a high-pass filter with 100 Hz cut-off frequency and use 24 frequency bins within the frequency range of (100, 250] Hz as the input features to retrain the DNN model. StealthyIMU maintains a high signal SNR of 12.56 dB, and low SER (SEER) of 25.10% (11.93%), when the victim smartphone is in a car with a 4-cylinder engine and speed

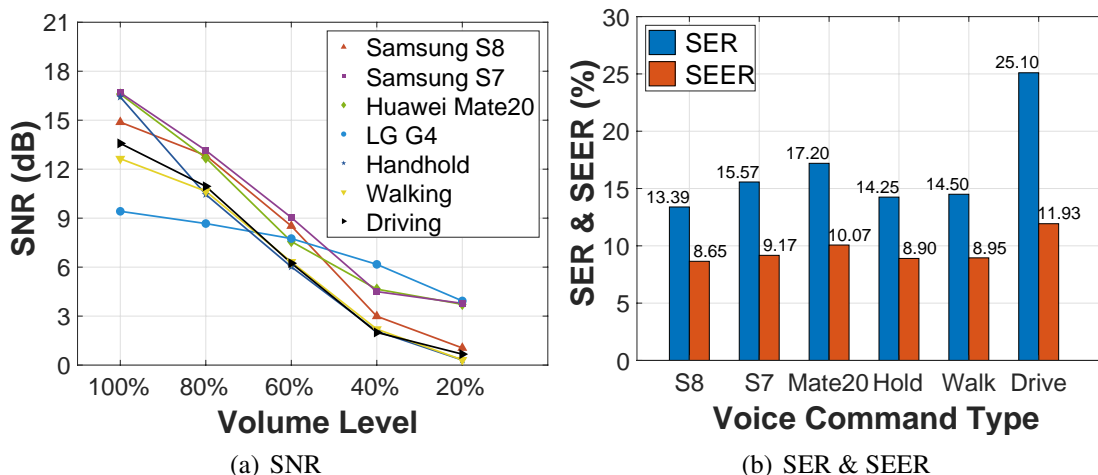


Figure 3.15. Generalization across different smartphones, volume levels, and motion artifacts. “Driving” achieves higher performance because it is evaluated by navigation data set only while others are evaluated by general dataset.

ranging from 30 to 110 km/h.

Generalization across different cities: We perform a fake GPS test of StealthyIMU in two cities with different geographical layout (Sec. 3.8.1). Note that StealthyIMU can only extract the GPS location for a specific city with the training dataset, and the DNN model needs to have the capability to extract the road names in a specific city. Therefore, we train the private entity recognition DNN models for city A and B respectively and evaluate the performance by using the corresponding model. This procedure is practical in real-world because the attacker can easily extract the city name of the victim’s location by using long-term monitoring (Sec. 3.6.3) and select the corresponding model for further attack. As shown in Table 3.8, although our dataset only covers 37% of the roads in City B, the SEER and SER of City B is still reasonably low (16.25% and 32.41% respectively), because even with such a small dataset, StealthyIMU can already recognize the main roads in the city, which will be frequently passed by users. By enriching our dataset, the accuracy can be further improved. We leave such incremental refinement for future work.

Table 3.10. Voice detection and segmentation overhead.

	Mobile Data	Segment Storage	Segment Memory	Peak CPU	Power
1	100 KB/min	/	/	/	/
2	16 KB/Seg	16 KB/Seg	/	5%	36 mW
3	16 KB/Seg	/	16 KB/Seg	5%	35 mW
4	/	/	16 KB/Seg	5%	35 mW

Table 3.11. On-device DNN overhead. ID: VUI response identification model; SLU: VUI response private entity recognition model.

	Model Size	Peak CPU	APP Memory	Time (s/Seg)	Energy (mAh/Seg)
ID	79.6 KB	5%	4.1 MB	$9.8e-3$	$6.5e-3$
ID	1.7 MB	5%	7.4 MB	$53.3e-3$	$1.1e-3$
SLU	9.1 MB	13%	54.5 MB	4.60	0.65
SLU	3.9 MB	12%	34.5 MB	1.19	0.17

3.9.6 System Overhead Evaluation

We evaluate the system overhead and required permissions when processing StealthyIMU in cloud or on device. Based on our threat model (Sec. 3.3), we summarize 4 deployment models of StealthyIMU: 1. The malicious app steams the MSS to the cloud for processing; 2. The malicious app detects the voice-associated signals on device and saves the segments in local storage, and then the segments will be uploaded to the adversaries’ cloud for further processing; 3. The process is similar to 2 except that the segments will be saved in app memory; 4. StealthyIMU attack is deployed fully on device.

To compare the 4 options, we deploy the StealthyIMU attack on Samsung Galaxy S8 with Qualcomm Snapdragon 835 CPU, assuming that StealthyIMU is not able to access the GPU resources. We use Android Profiler [21] and app power monitor to measure the on-device overhead in terms of energy, CPU, memory, storage, and mobile data usage. Table 3.10 shows

the results. Option 1 requires the “network” permission when the StealthyIMU app is running in the background and consumes 100 KB/min mobile data. In comparison, both option 2 and 3 need to upload the data to the cloud only when the voice is detected. The difference is that option 2 needs the “storage” permission whereas option 3 does not. Option 4 executes the entire StealthyIMU on device with zero permission. The peak CPU usage and power consumption of voice detection and segmentation are less than 5% and 46 mW respectively, which means that it only consumes 5.6% battery for 24 hours on a typical smartphone (3000 mAh, 3.7 V).

Table 3.11 shows the on-device DNN overhead. The input of our DNN is the voice-associated MSS segments, so we use seconds per segment and mAh per segment to measure the time and energy consumption of our models. Medium sized DNN models, *i.e.*, identification model with 1.7 MB model size and SLU with 3.9 MB model size, are more feasible for the on-device attack because the performance is close to the large models (Sec. 3.9.1) while the overhead is significantly lower. The app memory consumption is less than 42 MB in total and the peak CPU usage is less than 13%. These DNN models can recognize 176 voice-associated segments with less than 1% battery consumption. Overall, the behavior of the zero-permission on-device StealthyIMU is unlikely to be distinguishable from an innocuous app.

3.9.7 Defense Evaluation

We use three metrics to evaluate the defense capability of the proposed speech predistortion defense: 1) SNR, 2) PSNR, and 3) DNN model SEER and SER. As shown in Fig. 3.16(a), the high pass filter mechanism can reduce the SNR of the MSS by 12.7 dB. When the cut-off frequency f_c exceeds 350 Hz, the SNR will not decrease significantly anymore since this is close to the highest sampling rate (500 Hz) and the harmonics of higher frequencies start to emerge. Further augmented with chirp distortion, our defense mechanism reduces the PSNR down to 16 when the chirp frequency is set to 100 ~125 Hz and amplitude 0.1. This implies the structural features within the spectrogram are substantially corrupted, since intelligible speech requires 25 ~ 30 PSNR [289].

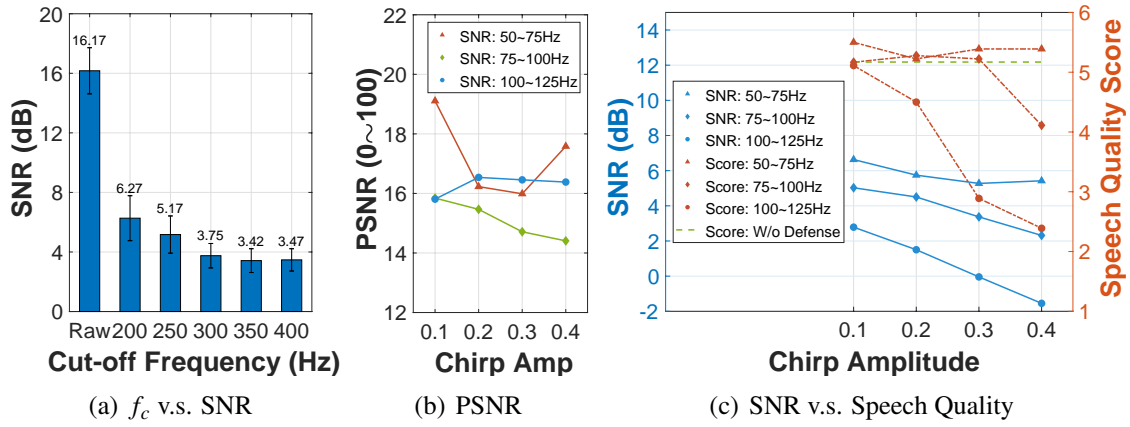


Figure 3.16. Predistortion Speech Defense

Table 3.12. Defense Speech Quality Subjective Assessment

Chirp freq (Hz)	Amp	Comfort	Intelligibility	Noise	Overall
/	0	5.5	5.17	4.83	5.17
50~75	0.1	5.33	6.00	5.17	5.50
50~75	0.2	4.83	5.83	5.00	5.22
50~75	0.3	5.17	6.00	5.00	5.39
50~75	0.4	5.17	5.67	5.33	5.39
75~100	0.1	4.67	6.00	4.83	5.17
75~100	0.2	4.83	5.67	5.33	5.28
75~100	0.3	5.17	5.83	4.67	5.22
75~100	0.4	3.83	5.17	3.33	4.11
100~125	0.1	4.67	5.50	5.17	5.11
100~125	0.2	4.33	5.33	3.83	4.50
100~125	0.3	2.50	3.83	2.33	2.89
100~125	0.4	1.83	3.33	2.00	2.39

Next, we investigate the trade-off between defense capability and speech quality. We conducted a subjective assessment to evaluate the speech quality after our speech predistortion defense. We recruited 30 volunteers (23 males and 7 female with ages in the range of 19 to 27) to assess the speech quality. In each study, we first randomly select a speech signal with a single VUI response with 12 different predistortion defense parameters and a raw signal without predistortion defense. And then the volunteers are asked to listen to the selected speech signals transmitted by 15 different smartphones without any prior knowledge of the signals. After listening to the speech signals, three metrics are assessed through a questionnaire with 1 ~ 7 rating [155] for each question: (i) Comfort: Do you feel comfortable with that speech? (ii) Intelligibility: Can you understand that speech? (iii) Quality: Can you hear noise in that speech signals? Each volunteer will assess all of 13 different speech signals. Table 3.12 shows the results of our subjective assessment. Fig. 3.16(c) shows the mean values of the subjective metrics. We observe that, *with the high pass filter and the default chirp frequency band 100 ~ 125 Hz and chirp amplitude 0.1 in the predistortion, the speech quality score is 5.11—close to that without predistortion (5.17), whereas the attacker’s SNR and PSNR drop sharply (2.78 dB and 15.81) below the threshold for intelligible speech.*

We further verify whether an attacker can circumvent the predistortion defense by using the predistorted MSS to train its SLU model. We first apply predistortion to our MSS dataset, and then use the SLU model configurations with the best performance to train and test the dataset. The result shows 95.06% SER and 89.17% SEER, which means that *the predistortion defense is sufficient to prevent the StealthyIMU from recognizing the private intents.* The attacker cannot reverse the speech predistortion from the MSS, even if it knows how the predistortion is applied.

3.10 Conclusion

We have demonstrated the feasibility and effectiveness of StealthyIMU, a new threat that allows a zero-permission app to steal private information from VUI responses on a smartphone.

The attack surface lies in a side channel, where in a motion sensor “overhears” the low-frequency vibration from the co-located loudspeaker. Although word-by-word transcription of general speech is challenging [32], we leverage the deterministic features of the machine-rendered VUI responses, and design a speech detection and understanding model to extract the private information. We further optimize the model and limit its resource usage, so it is indistinguishable from an innocuous app. Our case studies show that StealthyIMU can accurately steal crucial permission-protected private information, such as contacts, search history, calendar, home address, and GPS routes, from popular VUIs such as Google Assistant and Google Map. We further develop effective defense mechanisms which can help VUI vendors remove the vulnerability.

3.11 Acknowledgments

This chapter contains material from “StealthyIMU: Extracting Permission-protected Private Information from Smartphone Voice Assistant using Zero-Permission Sensors”, by Ke Sun, Chunyu Xia, Songlin Xu, and Xinyu Zhang, which appears in the 30th edition of the Network and Distributed System Security Symposium, 2023 [213]. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Single-Channel Speech Enhancement Using Ultrasound on a Smartphone

4.1 Introduction

Human auditory system is remarkably capable of singling out a speech source amid a mixture of interfering speakers and noises, which remains a key challenge for machine hearing. The problem has witnessed a surge in today's digital communication systems for human-human and human-machine interaction. Examples include mobile VoIP, voice commands, post-production of live speech, *etc.* The related research problem of speech separation and enhancement (SSE) is often considered as the holy grail of audio processing.

Since the problem is inherently ill-posed, classical solutions need to rely on prior knowledge (*i.e.*, per-speaker feature engineering) [244] or directional microphone arrays [73] to isolate the desired source from ambient sounds. In the past several years, deep learning techniques have proliferated and significantly advanced the field, enabling single-microphone speaker-independent SSE [241]. State-of-the-art solutions have demonstrated around 10 dB improvement in average audio quality, in separating a mixture of 2 clean speeches [146]. However, the challenging scenario of more than 2 speakers mixed with background noise received little attention [257]. A very recent preliminary test [71] revealed that existing deep learning models often underperform in such cases, because the unstructured background noise compromises their ability to identify separable structures in the speech streams. In addition, existing audio-only

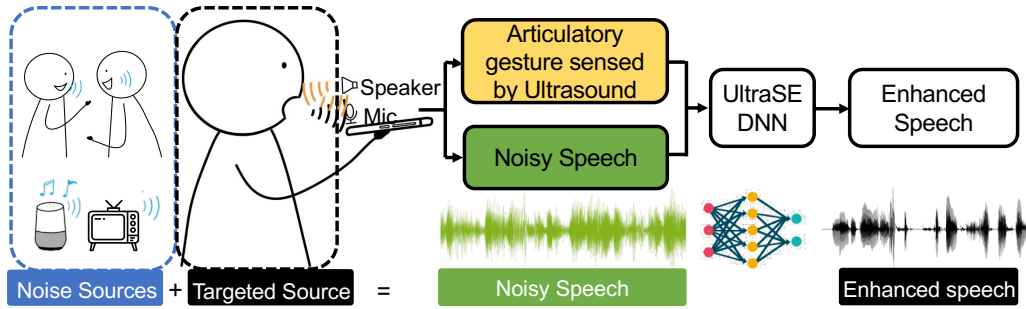


Figure 4.1. UltraSE targets the scenario where the user holds the smartphone to record the speech in a noisy environment. UltraSE uses ultrasound sensing as a complementary modality to separate the desired speaker’s voice from interferences.

approaches cannot solve the *label permutation problem*, *i.e.*, associating the model outputs to the desired speaker. Audio-visual algorithms [71] leverage video recordings of the speakers’ faces to simultaneously solve the SSE and permutation problems. However, the need for a camera at specific view angle and under amenable lighting condition limits their practical usability [24].

In this paper, *we propose to utilize ultrasound sensing as a complementary modality to separate the desired speaker voice from noises and interferences*. Our method, called UltraSE, is applicable to commodity mobile devices (*e.g.*, smartphones) equipped with a single microphone and loudspeaker. Figure 4.1 illustrates our basic idea. During the voice recording, UltraSE continuously emits an inaudible ultrasound wave, which is modulated by the speaker’s articulatory gestures (lip movement in particular) close to the smartphone. The signals recorded by the microphone thus contain both the audible sounds and inaudible reflections. As illustrated in Figure 4.1, whereas the audible sounds (“Green”) mix the targeted clean speech (“Black”) and other interferences plus background noise (“Blue”), the inaudible reflections (“Orange”) only capture the targeted user’s articulatory gesture motion which is correlated with the clean speech. UltraSE employs a DNN framework to capture such correlation and denoise the audible sounds.

UltraSE faces 3 core design challenges. *i) How to characterize the articulatory gestures by ultrasound despite interference?* It is challenging to capture the fine-grained articulatory gestures since they are fast ($-80 \sim 80$ cm/s) and subtle (< 5 cm displacement). Moreover, mutual interference exists between the speech and ultrasound due to harmonics and hardware artifacts.

To address the challenge, we fully exploit the advantages of ultrasound, *i.e.*, high sampling rate and perfect alignment with the clean speech in the time domain. We design the transmitted ultrasonic waveform to capture the *short-term high-resolution Doppler spectrogram*, and apply a one-time transmission volume calibration to reduce the cross-modality interference.

ii) How to design a DNN model to fuse the two modalities and represent their correlation?

Since the physical feature characteristics of the two modalities are different, we design a two-stream DNN architecture to process each and a self-attention mechanism to fuse them. Further, no existing method has addressed the cross-modal noise reduction problem which is fundamental to UltraSE, *i.e.*, using one modality (ultrasound) to reconstruct another modality (speech) which is polluted by noise/interference. We thus propose a conditional GAN (cGAN) based training model, with a novel cross-modal similarity measurement network, to enable this capability.

(iii) How to improve both intelligibility and quality for the enhanced speech? It is known that the amplitude of time-frequency (T-F) spectrogram is critical for speech intelligibility, whereas the phase determines the speech quality [261]. We thus expand UltraSE into a two-stage multi-domain DNN architecture, which prioritizes the optimization of intelligibility in the T-F domain, and then reconstructs phase in the T domain to improve speech quality. We place the multi-modal fusion network inside the T-F domain, based on the empirical observation that the articulatory gestures are more related to the speech intelligibility.

To evaluate UltraSE, we develop an Android app to collect a new speech dataset called UltraSpeech, which contains 22.2 hours of clean speech and corresponding ultrasound sensing signals from 20 users. We then combine UltraSpeech with the DARPA TIMIT speech corpus [81] and AudioSet ambient noise dataset [82] to create a 300 hours noisy speech dataset. Our evaluation results show that UltraSE can separate the targeted speech in a sophisticated environment with multiple speakers and ambient noise, improving SNR by 10.65 to 17.25 dB. UltraSE achieves an SNR gain of 6.04 dB on average over state-of-the-art single-channel speech enhancement methods, across various interference/noise settings. Its performance gain is even comparable to multi-channel (audio-visual) solutions.

UltraSE represents the first audio-only method to bring the SSE performance close to multi-channel solutions, while overcoming the label permutation issue. Through the UltraSE design, we make the following technical contributions:

- We design a *multi-modal multi-domain DNN framework for single-channel speech enhancement* which fuses the ultrasound and speech features, and simultaneously improves speech intelligibility and quality.
- We design a *cGAN-based cross-modal training model* which effectively captures the correlation between ultrasound and speech for multi-modal denoising.
- We collect a new speech dataset—UltraSpeech, and verify UltraSE’s performance in comparison with state-of-the-art solutions.

4.2 Related Work

4.2.1 Audio-only Speech Enhancement

Despite decades of research, speech enhancement remains a challenging open problem that attracts extensive research today [71, 140, 274, 225]. Classical model-driven solutions [69, 44, 70] typically build on various assumptions, such as stationarity of signals, uncorrelated clean-speech and noise, independence of speech and noise in the time-frequency domain, *etc.* Thus, they often lack robustness in real-world environment [241]. More recent solutions adopt supervised learning instead [241], and can be categorized by their domain of feature processing.

T-F domain methods: Time-Frequency (T-F) domain methods aim to learn a spectrogram *mask*, *i.e.*, a weighting matrix that can be multiplied with the noisy speech spectrogram to recover the desired clean speech [274]. The key problem lies in *i)* what type of mask should be used, and *ii)* how to use DNN to predict such a mask. Early stage solutions only estimate the amplitudes of a spectrogram by using real-valued Ideal Binary Mask (IBM) [106], Ideal Ratio Mask (IRM) [167] or Spectral Magnitude Mask (SMM) [253]. They then directly apply the

original noisy phase on each T-F bin to generate the enhanced speech. Although these amplitude masking methods benefit speech intelligibility, they suffer from poor speech perceptual quality due to the unavoidable phase error. Complex Ideal Ratio Mask (cIRM) [262] and Phase-Sensitive Mask (PSM) [72] are then proposed to incorporate phase information. Recently, PHASEN [274] and Ni *et al.* [170] found that the estimated cIRM tends to downgrade to IRM, since the T-F domain phase is close to white noise especially for low-amplitude T-F bins. Thus, they proposed two-stream [274] or two-stage [170] networks to take both the IRM and cIRM and derive a combined training loss. For the model design, most T-F domain methods deem the T-F spectrogram as an image, and design DNN/CNN-based models [262, 178] to minimize the MSE/MAE loss between the estimated mask and ground truth. PHASEN [274] and Ouyang *et al.* [174] observed that the fundamental frequencies and speech harmonics are separated afar, and the correlation cannot be fully captured by CNN. So they adopt dilated convolution and frequency-domain attention instead. Unlike the hand-crafted MSE/MAE loss function, Soni *et al.* [206] further used GAN to discriminate whether the enhanced results are clean or noisy.

T domain methods: Time (T) domain methods divert around the error-prone phase prediction problem by processing the waveform directly. For example, Rethage *et al.* [188] modified the WaveNet; TCNN [177] proposed an encoder-decoder architecture with an additional temporal convolutional net; SEGAN [179] utilized a GAN-based network to generate the 1D waveform of clean speech. Yet the performance of such methods is not among the top tier, since the speech auditory patterns, such as proximity in time/frequency, harmonics, and common amplitude/frequency modulation, are more prominent on a T-F spectrogram [241].

Multi-domain methods: In recent concurrent work TFTNet [225], a learnable decoder replaces the iSTFT in the T-F domain to realize a joint T-F and T domain model for speech enhancement. Unlike TFTNet, our key insight is that the speech intelligibility is much more important than speech quality for speech enhancement. We thus design a two-stage multi-domain DNN network to prioritize the optimization of speech intelligibility in the T-F domain, and then reconstruct phase in the T domain to improve the speech quality.

Speech source separation: Although most of the aforementioned approaches demonstrated acceptable performance for non-speech noise, they still can not handle the *cocktail party* scenario involving multiple interfering speakers. To resolve such speech separation problems, Deep clustering [104] trained speech embedding for each source and then uses clustering algorithms to separate them. PIT [276] iteratively changed the permutation of sources in the training process to train a permutation invariant speech separation model. *These methods still need to know the number of speakers a priori*, and do not work well for the case with more than 3 speakers plus noise [221]. Further, the label permutation problem persists—They can separate multiple sources of speech, but cannot automatically identify which is from the targeted speaker, which may hinder certain machine-operated back-end tasks (*e.g.*, voice assistant on a smartphone). UltraSE overcomes all these deficiencies.

4.2.2 Multi-modal Speech Enhancement

To tackle the permutation issue, audio-visual (AV) methods use a video recording of the subject’s face as a hint for the audio [103, 189]. Specifically, Ephart *et al.* [71] trained a speaker-independent speech separation model based on a large set of YouTube videos [71]. Afourasl *et al.* [24] found that even partially occluded videos of lip motion can assist speech separation. Nonetheless, AV approaches bear many drawbacks. Besides microphone, they need an additional camera pointing to the subject’s face under good lighting conditions, which is inconvenient and even infeasible in many typical use cases. Moreover, camera is unusable in many privacy-sensitive locations.

The idea of using ultrasound as a complementary modality to enhance speech has been explored by previous works [129, 40]. However, these works [129, 40] all require special ultrasonic hardware. In comparison, UltraSE only needs the single audio channel on the smartphone and overcomes practical challenges such as mutual interference between modalities. Besides, they use traditional methods, *i.e.*, non-negative matrix factorisation [40] and nonlinear regression [129], and only show the performance of speech enhancement on ambient noise rather

than speech interference. UltraSE further pushes the limits of this idea by designing a multi-modal multi-domain DNN framework to achieve similar performance for speech separation and enhancement with the audio-visual methods.

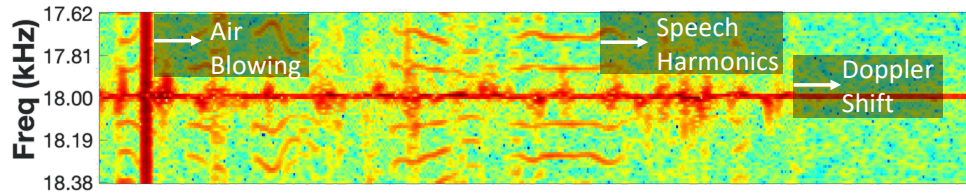
4.2.3 Device-free Ultrasonic Sensing

Device-free ultrasonic sensing techniques can leverage the loudspeakers and microphones on commodity mobile devices to track the distance/direction changes of nearby objects [248]. State-of-the-art ultrasonic gesture tracking schemes [248, 278, 212, 216, 149] can achieve mm-level accuracy. Besides location and hand gesture tracking, recent studies also attempted to use ultrasonic sensing for lip reading [222]. However, due to insufficient spatial resolution, they only fit coarse sensing applications, *e.g.*, liveness detection [283, 131]. SilentTalk [222] uses a model-based method to classify the Doppler shift features caused by 12 basic mouth motions and recognize specific short sentences. SilentKey [223], EchoPrint [287], LipPass [143], and VocalLock [144] use the ultrasonic sensing features introduced by mouth motion for biometric authentication. In contrast, UltraSE is the first to demonstrate that ultrasonic sensing can serve as a complementary modality to solve the *cocktail party problem* and bring speech enhancement to the next level.

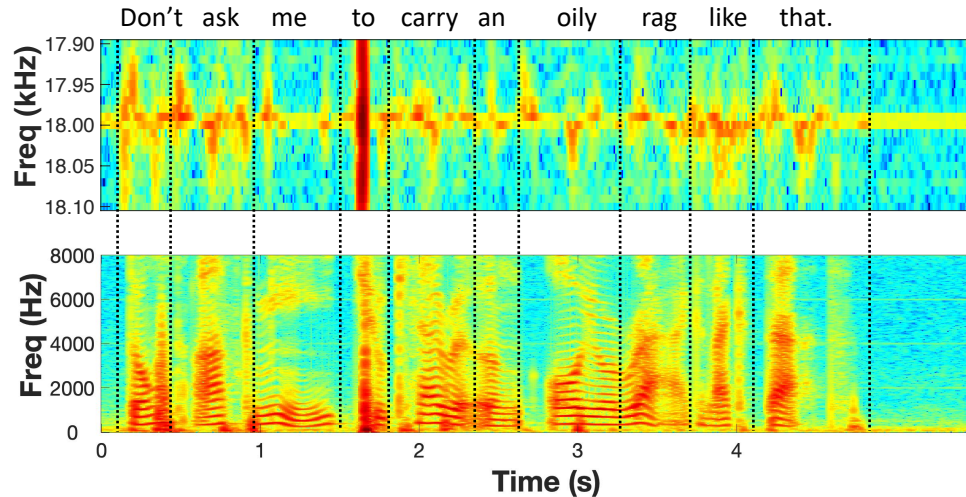
4.3 Sensing the Articulatory Gestures

In this section, we first provide a primer on the relationship between speech and articulatory gestures. Then we introduce UltraSE’s ultrasound sensing signal design, along with mechanisms to mitigate the mutual interference between speech and ultrasound.

Human speech generation involves multiple articulators, *e.g.*, tongue, lips, jaw, vocal cords, and other speech organs [283]. Coordinated movement of such articulators, including lip protrusion and closure, tongue stretch and constriction, jaw angle change, *etc.*, is used to define the phonological units, *i.e.*, phoneme in phonology and linguistics [45]. Thus, assuming that we can fully capture and interpret the articulatory gestures, it would be possible to recover



(a) Speech harmonics create interference within the ultrasound band.



(b) Doppler shift spectrogram of a single-tone 18 kHz transmitted signal and the corresponding T-F spectrogram of speech w/o interference.

Figure 4.2. T-F domain features of an example speech segment: “Don’t ask me to carry an oily rage like that.”

the speech signals. However, it is challenging to capture the fine-grained gesture motion of all articulators by using a single microphone [222]. First, the articulators are close to each other. Some are inside the mouth/throat. So it is hard to discriminate their motion. Second, the articulatory gestures are always fast and subtle. Each typically lasts 100 ~ 700 ms and involves < 5 cm moving distance for lip and jaw [226]. Thus, state-of-the-art sensing methods can only recognize a limited number of words or phrases by using COTS microphones [222], and the accuracy in the wild is typically quite low [56]. In UltraSE, we do not expect that the captured articulatory gesture features can directly synthesize the speech signals. We propose to take these features as coarse complementary information to facilitate the SSE.

4.3.1 Transmitted Ultrasound Signals Design

Modality advantages: Compared to other approaches such as camera, ultrasound possesses two advantages in sensing articulatory gestures. First, the ultrasound sensing signals are captured by using the same sensor (*i.e.*, microphone) as the speech signals. This introduces an automatic “*feature alignment*” in the time domain, which means the captured ultrasound sensing features are well synchronized and matched with the clean speech signals. Second, the *sampling rate* of the ultrasound sensing (typically 48 kHz or 96 kHz) is much higher than vision-based methods (typically 24 ~ 120 fps), which enables finer time resolution when capturing the articulatory gestures.

Design goals: Compared to previous works on ultrasound based gesture sensing especially hand gestures [248, 278, 216, 165], UltraSE needs to satisfy the following additional design goals to fully exploit the modality advantages: *(i)* The extracted features require high sampling rate to achieve high T-F resolution. The velocity of users’ articulatory gestures ranges from $-80 \sim 80$ cm/s ($-160 \sim 160$ cm/s for propagation path change) [226], which will introduce $-100 \sim 100$ Hz Doppler shift when the transmitted signal’s frequency is 20 kHz. Meanwhile, each articulatory gesture corresponds to a single phoneme lasting 100 ~ 700 ms [283], which is approximately 5 times shorter than hand gestures [250]. *Therefore, to characterize the articulatory gestures, the ideal way is to characterize the short-term high-resolution Doppler shift.* *(ii)* The extracted features need to be robust to different kinds of noises introduced by multipath and frequency-selective fading. UltraSE thus needs to remove the reflections from static objects, mitigate the multipath from moving objects (*e.g.*, body parts), and extract the signal features from articulatory gestures alone.

Ultrasonic sensing signal design: To satisfy these requirements, we choose multiple single-tone continuous waves (CWs) with linearly spaced frequencies as our transmitted signals. Although modulated CW signals, such as FMCW [148], OFDM [165] and Pseudo-Noise (PN) sequences [278, 216], can measure the impulse response to resolve multipath, they all suffer

from the aforementioned low sampling rate problem. The fundamental reason is that the modulation processes signal in segments (*i.e.*, chirp period or symbol period). Thus, each feature point of the modulated CW signal characterizes the motion within a whole segment, which is typically longer than 10 ms (960 samples) at a sampling rate of 96 kHz. Thus, only 10 ~ 70 feature points can be output for each articulatory gesture with typical duration of 100 ~ 700 ms [283], which can hardly represent the fine-grained instantaneous velocity of gesture motion. In comparison, each sampling point of the single-tone CW can generate one feature point (Doppler shift estimation) to represent the micro motion with duration of 0.01 ms ($\frac{1}{96000}$) at a sampling rate of 96 kHz. To further resolve the multipath effect and frequency selective fading, we combine multiple single-tone CWs with equal frequency spacing, resulting in a transmitted waveform $T(t) = \sum_{i=1}^N A_i \cos 2\pi f_i t$, where N , A_i and f_i denote the number of tones, the amplitude and frequency of the i^{th} tone, respectively.

To alleviate the spectral leakage across different tones when generating the spectrogram in later stage, we ensure that the STFT window size (1024 points) is a full cycle of all the transmitted tones at the maximum sampling rate (48 or 96 kHz allowable by COTS microphones). We empirically set the first frequency $f_0 = 17.25$ kHz, the frequency interval $\Delta f = 750$ Hz, and the number of tones $N = 8$. We decrease the amplitude A_i of the sub-20kHz frequencies to make sure that the transmitted signals will not disturb users.

4.3.2 Mitigating Sensing Interference

Despite the orthogonality in frequency, mutual interference exists between speech and ultrasound in the following two cases, which causes ambiguity of Doppler features.

First, the speech harmonics may interfere the Doppler features due to non-linearity of microphone hardware. The speech and ultrasound signals generated in UltraSE are combined in the air, resulting in $S_{in}(t) = v(t) + \sum_{i=1}^N A_i \cos 2\pi f_i t$, where $v(t)$ represents the speech signals, and $\sum_{i=1}^N A_i \cos 2\pi f_i t$ is the high-frequency ultrasound sensing signals. Due to the microphone non-linearity [191, 193, 211], the captured signals can be modeled as $S_{out} \approx A^1 S_{in} + A^2 S_{in}^2$ [191], which

contains speech harmonics on the inaudible ultrasonic band, *i.e.*, $S_{noise} = \sum_{i=1}^N A_i^2 v(t) \cos 2\pi f_i t$. As shown in Figure 4.2(a), these speech harmonics often leak into the ultrasonic band, and will corrupt the articulatory gestures’ Doppler features. Fortunately, when we decrease the amplitude of the ultrasound A_i , the second order term (harmonics’ amplitude A^2) decreases faster than the first order term (Doppler shift amplitude A^1). Our empirical experiments reveal that, when the total amplitude of transmitted ultrasound is set to < 80 dBz (flat weighting) sound pressure level (measured at 5 cm away from the speaker), the interference effect becomes negligible. We thus always use this setting as the *default ultrasound amplitude* in UltraSE. It is worth noting that previous ultrasound based hand gesture sensing schemes [248, 278, 216] did not address this interference issue because they are typically tested without strong close-by speech interference.

Second, when a user speaks close to the microphone, some phonemes, *e.g.*, /p/ and /t/, may blow air flow into the microphone which generates high-volume noise. As an example, Figure 4.2(a) shows the T-F spectrogram introduced by the phoneme /t/. Amid the high-volume air flow, the microphone has to prevent saturation by calling on its auto gain control (AGC), which scales down all incoming signals and consequently renders the Doppler features negligible. In UltraSE, instead of removing the corrupted samples, we harness them as part of the ultrasonic sensing features, which helps characterize the sampling period corresponding to specific phonemes (*e.g.*, the /t/).

4.4 An Overview of UltraSE DNN Model

For ease of exposition, we will first introduce the basic DNN architecture of UltraSE, and then discuss the challenges and design principles of each design component in the following sections. Our first step is to create the DNN input features from the raw signals (Section 4.5). Then, we design a *two-stage, multi-modal, multi-domain DNN model*, which comprises three key modules, as briefed below.

T-F domain multi-modal amplitude network (Section 4.6). This network module

generates the amplitude Ideal Ratio Mask (aIRM), *i.e.*, the ratio between the magnitudes of the clean and noisy spectrograms, by using both speech and ultrasound as the input. It consists of two subnetworks.

Subnet (i) Two-stream feature embedding: Our model starts by using the noisy speech’s T-F spectrogram and the concurrent ultrasound Doppler spectrogram as input (Section 4.5). We then design a two-stream feature embedding architecture, to transform the different modalities into the same feature space, while maintaining their time-domain alignment.

Subnet (ii) Speech and ultrasound fusion network: Then, we concatenate the features of each stream in the frequency dimension. A self-attention mechanism is further applied to fuse the concatenated feature maps to let the multi-modal information “crosstalk” with each other. The fused features are subsequently fed into a BiLSTM layer followed by three FC layers. The resulting output is an *amplitude mask* which is multiplied with the original noisy amplitude spectrogram to generate the amplitude-enhanced T-F spectrogram.

cGAN-based cross-modal training (Section 4.7). As shown in Figure 4.7, we design a cGAN-based training method to further denoise the amplitude-enhanced T-F spectrogram. In our cGAN model, the generator is the above T-F domain multi-modal amplitude network; the discriminator is designed to discriminate whether the enhanced spectrogram corresponds to the ultrasound sensing features.

T domain phase network (Section 4.8). We use the iSTFT (a fixed 1D convolution layer) [91] to transform the amplitude-enhanced T-F spectrogram into T domain waveform. To fine-tune the phase of the enhanced signals, we design an encoder-decoder architecture to reconstruct the phase to be close to the clean speech in the T domain.

4.5 DNN Input Feature Design

In this section, we discuss the preprocessing steps to generate the DNN input features for the two signal modalities. Figure 4.3 illustrates the workflow.

Speech feature extraction: Typical speech sound ranges from approximately 300 Hz to 3.4 kHz [228], and the signals above 8 kHz barely affect the speech intelligibility and human perception [159]. Thus, we first use a low-pass elliptic filter to extract the signals below 8 kHz. Then we resample the signals to 16 kHz by using a Fourier method. The final enhanced speech is also sampled at 16 kHz which suffices to characterize the speech signals. Higher sampling rate may unnecessarily increase the optimization space and model complexity.

The speech feature input for the DNN model is the T-F domain speech spectrogram, generated by applying STFT on the time domain waveform. The STFT uses a Hann window of length 32 ms, hop length of 10 ms, and FFT size of 512 points under 16 kHz sampling rate, resulting in $100 \times 257 \times 1$ complex-valued scalars per second.

Ultrasound sensing features: We first use a high-pass elliptic filter to isolate the signals above 16 kHz. Then, we create the ultrasound sensing features within the T-F domain, by extracting the Doppler spectrogram induced by articulatory gestures and aligning it with the speech spectrogram. The key consideration for this step is to balance the trade-off between time resolution and frequency resolution of the STFT under the limited sampling rate (96 kHz maximum). First, to guarantee time alignment between the speech and ultrasound features, their hop length in the time domain should be the same. The STFT uses a hop length of 10 ms to guarantee 100 frames per second, resulting in 10 ~ 70 frames per articulatory gesture which is enough to characterize the process of an articulatory gesture (Section 4.3). Second, the frequency resolution, determined by the window length, should be as fine-grained as possible to capture the micro Doppler effects introduced by the articulatory gestures, under the premise that the time resolution is sufficient. A window length 85 ms is the longest length for STFT to make it shorter than the shortest duration of an articulatory gesture (100 ms) [283]. Overall, under the 96 kHz sampling rate, the STFT is computed using a window length 85 ms, hop length of 10 ms, and FFT size of 8192 points, resulting in 11.7 Hz ($\frac{96000}{8192}$) frequency resolution.

In addition, to mitigate the reflections from relatively static objects, we remove the 3 central frequency bins and leave $8 \times 2 = 16$ frequency bins corresponding to Doppler shift

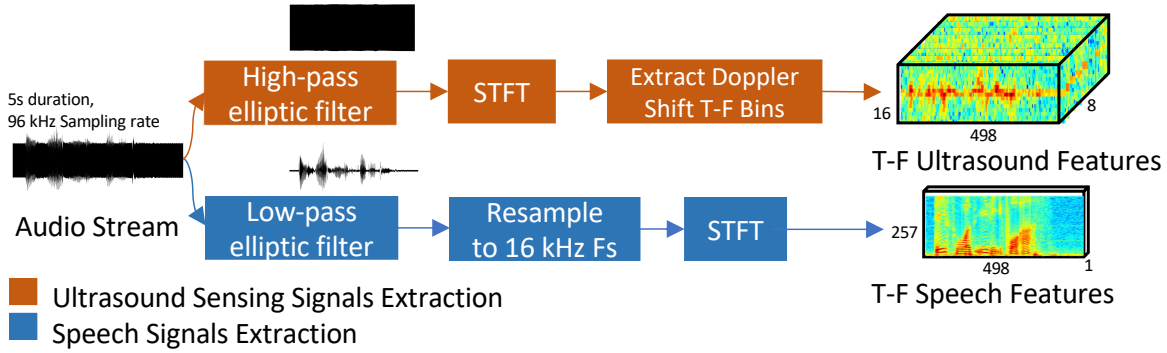


Figure 4.3. DNN input feature design

$[-11.7 \times 8, -11.7)$ and $(11.7, 11.7 \times 8]$ Hz. Finally, we run a min-max normalization on the ultrasound Doppler spectrogram. The resulting T-F domain ultrasound features is $100 \times 16 \times 8$ scalars per second, where 8 is the number of ultrasonic tones. The reason why we fuse the ultrasound sensing features in T-F domain instead of T-domain will be evident in the latter multi-domain design (Section 4.8).

The origin of the ultrasound feature and its correlation with the speech feature:

Figure 4.2(b) uses one example speech segment to visualize the alignment between the ultrasound Doppler spectrogram and the clean speech spectrogram. The ultrasound sensing features mainly consist of the $-100 \sim 100$ Hz Doppler shift introduced by relatively large motion from the lip, tongue and jaw. It can not capture the high-frequency micro-vibration motions introduced by the vocal folds [268], since the vocal vibration displacements (about $20 \mu\text{m}$ [52]) are much shorter than the ultrasound wavelength (about 2 cm).

Some obvious characteristics in this example corroborate the correlation between the ultrasound sensing features and corresponding clean speech features. For example, each word in the speech signals is well aligned with a burst of Doppler shifts from the articulatory gestures. Meanwhile, negative Doppler shift is introduced by mouth open gestures slightly before the onset of each word. Our DNN model is designed to learn such cross-modality correlation for the purpose of SSE.

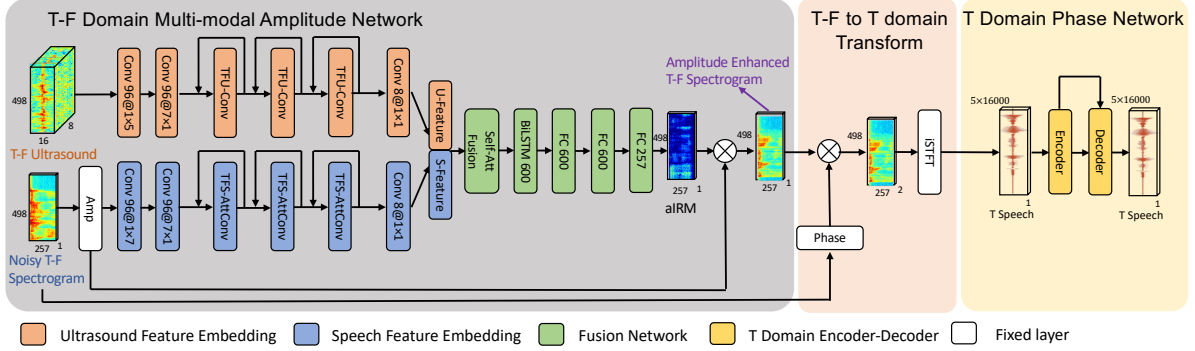


Figure 4.4. Overview of UltraSE’s multi-modal multi-domain DNN design. Convolution layer notation: Channels@Kernel size

4.6 Multi-modal Fusion Design

The multi-modal fusion network aims to first appropriately learn the F domain features of the two modalities respectively, and then fuse them together to exploit the T-F domain correlation. The F domain of the ultrasound signal features represents the *motion velocity* (Doppler shift) of the articulatory gestures, while that of the speech sound represents the *frequency* characteristics such as harmonics and consonants. Meanwhile, the size of the two modalities’ feature maps are different (Section 4.5). So one cannot straightforwardly concatenate these two feature maps into a scalar. We thus design a two-stream embedding architecture to transform them into the same feature space.

4.6.1 Two-stream Feature Embedding

Speech feature embedding: The input of the speech feature embedding subnetwork is the T-F domain amplitude spectrogram, denoted as $S_{noise}^a \in \mathbb{R}^{1 \times T \times F^a}$. $F^a = 257$ is determined by the STFT window size. The “blue” part in Figure 4.4 shows the architecture of this subnetwork, which comprises traditional 2D convolution layers and 3 “TFS-Conv” blocks. The “TFS-AttConv” block, borrowed from PHASEN [274], employs both the ResNet [101] and self-attention mechanism [107] to learn the global correlation of sound patterns across T-F bins. In contrast, the small kernels of CNN cannot capture such long-range correlations. Figure 4.5 shows

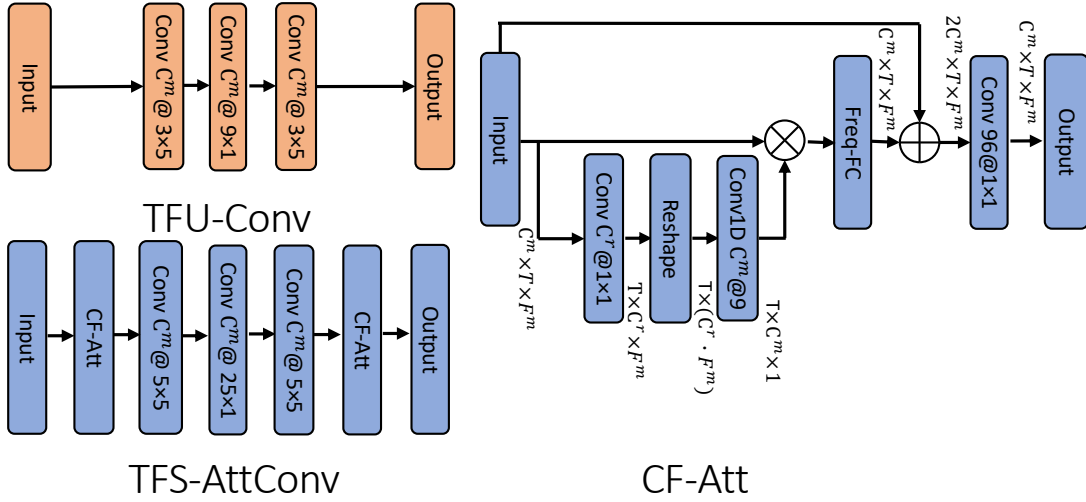


Figure 4.5. Two-stream feature embedding. Channels@Kernel size in convolution layer.

the structure of a single “TFS-AttConv” block. It contains 2 “CF-Att” blocks at the beginning and the end to learn the global correlation. In each “CF-Att”, a self-attention mechanism is used to fuse the channel-wise information following a SENet-based design [107]. Then, the “Freq-FC” layer applies a learnable frequency transformation matrix to enable frequency-domain self-attention at each point in the T domain. We omit other details of this block which has been covered in PHASEN [274].

Ultrasound feature embedding: The input of the ultrasound feature embedding is $U^s \in \mathbb{R}^{T \times F^s \times C^s}$, where $C^s = 8$ is the number of ultrasound tones, and $F^s = 16$ is the maximum number Doppler shift frequency bins introduced by the articulatory gestures (Section 4.5). Since the motion speed always changes continuously, the F domain ultrasound features are mainly local Doppler shift features. Small kernels suffice to capture such feature correlation because the size of the F domain is only 16. Therefore, instead of the “TFS-AttConv”, we design a “TFU-Conv” block which removes the attention layers and reduces the kernel size of the F domain in all the 2D convolution layers. To maintain the time alignment of the two modalities after feature embedding, we keep the T domain kernel size the same as in the “TFS-AttConv” block. For convenience of concatenating the two modalities’ features, we choose the same output channel number for all the 2D convolution layers.

Finally, after 3 “TFU-Conv” and “TFS-AttConv” blocks respectively, the channel number of the two streams reduces to $C_r^s = 8$ and $C_r^a = 8$ by applying a 1×1 2D convolution.

4.6.2 Speech and Ultrasound Fusion Network

After the feature embedding, we concatenate the feature maps of the two streams: $S_{in}^f = \text{concat}(M_o^a, U_o^s)$, where $S_{in}^f \in \mathbb{R}^{C^r \times T \times F^{as}}$ and $F^{as} = F^a + F^s$. This concatenated feature map is then fed into the “Self-Att Fusion” to learn the relationship between the two modalities. The “Self-Att Fusion” block is similar to the “CF-Att” block, but the size of the feature maps differs. *First*, since the meaning of channel in ultrasound sensing and speech is different, we first use a channel self-attention to learn the correlation across different channels. *Second*, to enable these two modalities’ features to “crosstalk” with each other in the F domain, the self-attention for the F domain is realized by using a learnable transformation matrix on the fused features. *Third*, the feature after self-attention fusion is concatenated with the original feature and fused by a 1×1 2D convolution.

Finally, the whole feature map is fed into a BiLSTM and 3 fully connected (FC) layers to predict the aIRM $\in \mathbb{R}^{T \times F_m \times 1}$ of the noisy speech. The predicted aIRM is then multiplied with the original noisy speech’s amplitude spectrogram to generate the *amplitude-enhanced T-F spectrogram*.

Note that all the convolutional layers in the multi-modal fusion network use zero padding, dilation= 1 and stride= 1 to make sure the output feature map size is the same as the input speech/ultrasound spectrogram. Also, each 2D convolutional layer is followed by batch normalization (BN) and ReLu activation.

4.7 cGAN-based Cross-modal Training

The fundamental problem for UltraSE is multi-modal noise reduction, *i.e.*, using one modality (ultrasound) to recover another modality (speech) which is polluted by noise/ interference. The former modality has low sensing resolution but is interference-free and correlated

Table 4.1. Layers comprising ultrasound subnetwork.

	Conv1	Conv2	Conv3	Conv4	Conv5	Conv6	Conv7	Conv8	Conv9	BLSTM10	FC11	FC12	FC13
Num Filters	48	48	48	48	48	48	48	48	8	300	600	600	5
Filter Size	1 × 7	7 × 1	3 × 3	3 × 3	3 × 3	3 × 3	3 × 3	3 × 3	1 × 1	Hidden Size	Output Size		

Table 4.2. Layers comprising speech subnetwork (BLSTM and FC layers parameters are the same as the ultrasound subnetwork.).

	Conv1	Conv2	Conv3	Conv4	Conv5	Conv6	Conv7	Conv8	Conv9	Conv10	Conv11	Conv12	Conv13
Num Filters	48	48	48	48	48	48	48	48	48	48	48	48	8
Filter Size	1 × 7	7 × 1	5 × 5	5 × 5	5 × 5	5 × 5	5 × 5	5 × 5	5 × 5	5 × 5	5 × 5	5 × 5	1 × 1
Dilation	1 × 1	1 × 1	1 × 1	1 × 2	1 × 4	1 × 8	1 × 16	1 × 1	2 × 2	4 × 4	8 × 8	16 × 16	1 × 1

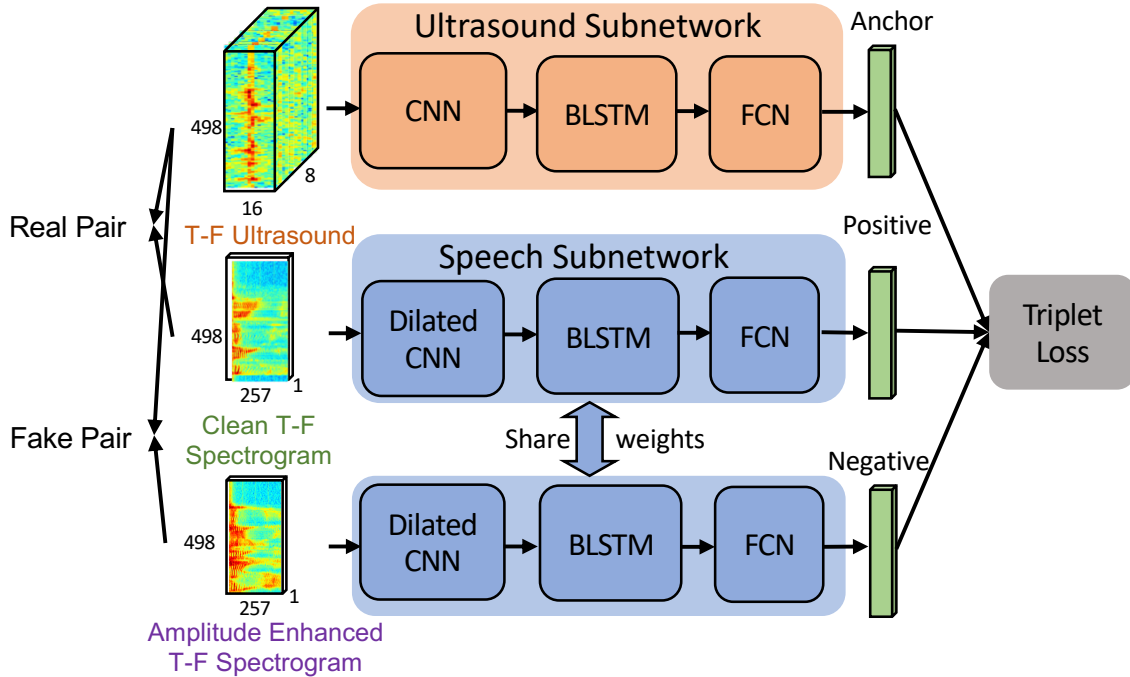


Figure 4.6. Architecture of the T-F domain cross-modal similarity measurement network (*i.e.*, the Discriminator).

with the latter. Although we intentionally maintain the time alignment between the two (Section 4.6), it is hard to force the multi-modal fusion network to “understand” such multi-modal correlation, because a traditional loss function (*e.g.*, MSE) can only train the network to clean up the T-F spectrum end-to-end. We thus propose a cGAN-based training method, which implicitly incorporates the maximization of cross-modal correlation itself as a training goal.

4.7.1 Cross-modal Similarity Measurement

A key element in any GAN design is to define the similarity metric used by the discriminator. Unlike traditional GAN applications (*e.g.*, image generation) which compare between the same type of features, our cross-modal cGAN needs to discriminate whether the enhanced T-F speech spectrogram matches the ultrasound Doppler spectrogram (*i.e.*, whether they are a “real” or “fake” pair). We propose a *cross-modal Siamese neural network* to meet this challenge.

A Siamese neural network uses shared weights and model architecture while working in tandem on two different input vectors to compute comparable output vectors. It is traditionally

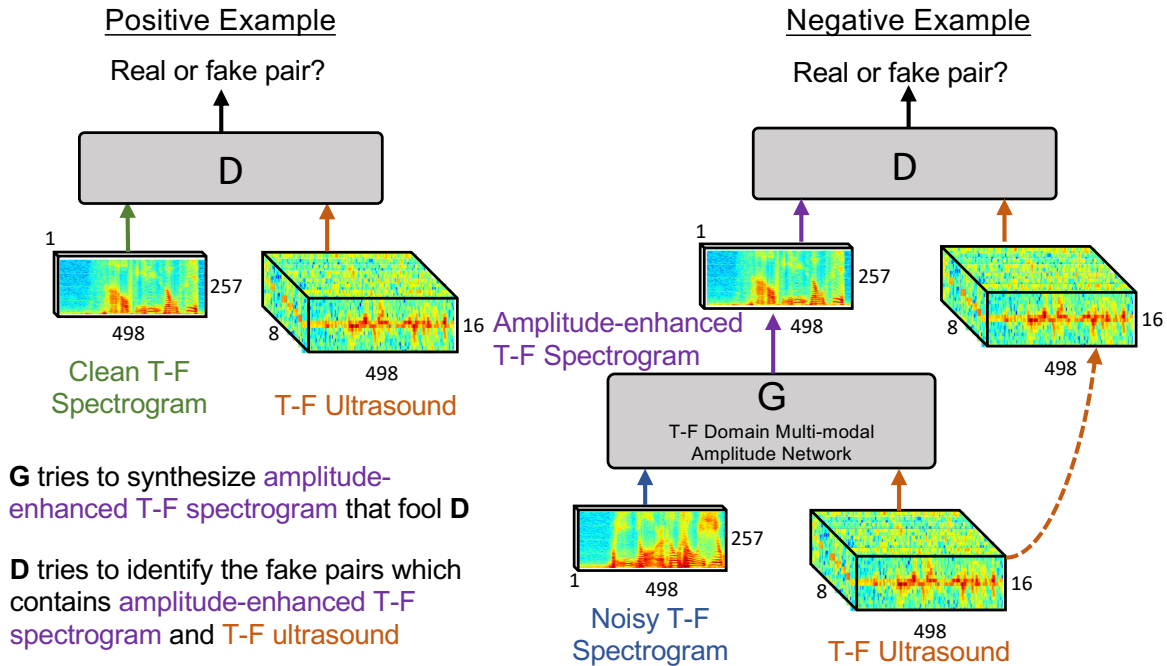


Figure 4.7. Overview of UltraSE’s cGAN-based cross-modal training.

used to measure the similarity between two inputs from the same modality, *e.g.*, two images [121]. To enable a *cross-modal* Siamese neural network, we create two separate subnetworks (Figure 4.6), aiming to characterize the correspondence between the T-F domain features of the speech and ultrasound, respectively. The basic architecture for these 2 inputs is a CNN-LSTM model. Since human speech contains harmonics and spatial relationship in the F domain, the speech CNN subnetwork uses *dilated convolutions* for *frequency domain context aggregation*. The Doppler shifts from ultrasound sensing mostly encompasses local features. Thus, the ultrasound CNN subnetwork only contains traditional convolution layers. Following the convolution, a Bi-LSTM layer is used to learn the long-term *time-domain information* for both modalities. Finally, three fully connected (FC) layers are introduced to learn two comparable output vectors respectively. We emphasize that the architecture and parameters are not shared in this cross-modal design, which differs from the traditional Siamese networks.

As shown in Figure 4.6, we use the Triplet loss [94] to train the cross-modal Siamese network. The triplet loss function accepts 3 inputs, *i.e.*, an anchor input U^s is compared to a

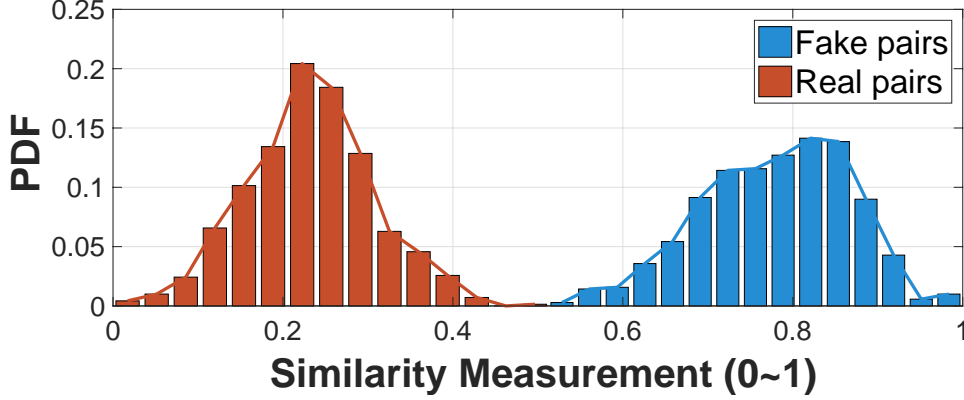


Figure 4.8. PDF of outputs from the cross-modal similarity measurement network.

positive input S_{gr}^a and a negative input S_{out}^a . It aims to minimize the distance between “real” pair U^s and S_{gr}^a , and maximize the distance between “fake” pair U^s and S_{out}^a . In our model, the anchor input U^s is the ultrasound sensing features, the positive input S_{gr}^a is the corresponding clean speech amplitude spectrogram, and the negative input S_{out}^a is the noisy speech amplitude spectrogram. Thus, our network model minimizes the following Triplet loss:

$$\begin{aligned} \mathcal{L}_{Triplet}(D) = & \mathbb{E}_{U^s, S_{gr}^a, S_{out}^a \sim p_{data}(U^s, S_{gr}^a, S_{out}^a)} \\ & [(\|f_u(U^s) - f_s(S_{gr}^a)\|^2 - \|f_u(U^s) - f_s(S_{out}^a)\|^2 + \alpha, 0)] \end{aligned} \quad (4.1)$$

where f_u is the ultrasound subnetwork, f_s is the speech subnetwork, and α is a margin distance between “real” and “fake” pairs.

We use a speech and ultrasound dataset collected on COTS smartphones (Section 4.9.1) to train the cross-modal Siamese network, and verify its effectiveness in a benchmark experiment. The training and testing sets contain 3 h and 0.5 h speech corpus for 15 and 5 users, respectively. The T-F domain speech feature input is a $1 \times 498 \times 257$ scalar (5 s segment), and the T-F domain ultrasound feature input is a $8 \times 498 \times 16$ scalar.

Figure 4.8 shows the probability density function (PDF) of outputs, where a smaller value indicates higher similarity. It is obvious that the output PDFs for the real pairs and fake pairs are perfectly separated, which means that *our similarity measurement network can effectively discriminate whether a pair of speech and ultrasound inputs are generated by the*

same articulatory gestures.

4.7.2 cGAN-based Model Training

Now we discuss how to leverage such similarity measurement as a discriminator in a cGAN to further fuse the multi-modal information. Our cGAN model aims to not only minimize the MSE of the speech amplitude spectrogram (relative to the ground-truth), but also guarantee high similarity between the “fake” pair (*i.e.*, the enhanced speech and ultrasound sensing features) and the “real” pair (*i.e.*, the clean speech and ultrasound sensing features).

cGAN has been widely used to add a conditional goal to guide a generator to automatically learn a loss function which well approximates the goal [109]. Figure 4.7 shows the structure of the UltraSE cGAN model. The generator “ $G(S_{noise}^a, U^s)$ ” is the aforementioned multi-modal network (Section 4.6), which takes the noisy speech amplitude spectrogram S_{noise}^a and ultrasound sensing spectrogram U^s as the input. $G(\cdot)$ is trained to output amplitude-enhanced T-F spectrogram of the speech S_{out}^a , which not only minimizes the traditional amplitude MSE loss [274], but also tries to “fool” an adversarially trained discriminator “ $D(S_{out}^a, S_{gr}^a, U^s)$ ”, which strives to discriminate the fake pair (S_{out}^a, U^s) from the “real” pair (S_{gr}^a, U^s) under the aforementioned triplet loss function. More specifically, The “D” loss is $\mathcal{L}_{Triplet}(D)$ (see Eq. (4.1)), and the “G” loss is

$$\begin{aligned} \mathcal{L}(G) = & \mathbb{E}_{U^s, S_{gr}^a, S_{noise}^a \sim P_{data}(U^s, S_{gr}^a, S_{noise}^a), z \sim P_z} \\ & [\mathcal{L}_{Triplet}(D(G(U^s, S_{noise}^a), S_{gr}^a), U^s)] + \lambda \|G(U^s, S_{noise}^a) - S_{gr}^a\|^2 \end{aligned}$$

where $\lambda \|G(U^s, S_{noise}^a) - S_{gr}^a\|^2$ is the traditional MSE amplitude loss. The reason why we use the amplitude MSE loss here rather than complex-valued loss or combined loss [274] will be clarified in Section 4.8.

Our cGAN design represents a general model for cross-modal noise reduction, which may be reused in other sensor fusion problems involving heterogenous sensing modalities.

Table 4.3. Layers comprising T domain phase network. Kernel size = 32, Stride = 2, Padding = 15.

	Enc1	Enc2	Enc3	Enc4	Enc5	Enc6	Enc7	Dec7	Dec6	Dec5	Dec4	Dec3	Dec2	Dec1
Num Filters	16	32	32	64	64	128	128	128	64	64	32	32	16	1

4.8 Multi-domain Speech Enhancement

In this section, we first investigate the pros and cons of T-F domain vs. T domain speech enhancement by using statistical analysis and experimental validation. Our key insight is that improving *intelligibility* is more critical than enhancing *quality*, since the top priority for speech enhancement lies in helping users/machines to understand the speech in noisy environment. This motivates us to expand the aforementioned T-F domain network into a two-stage multi-domain model, which first pushes the limits of intelligibility and then refines the speech quality.

4.8.1 Understanding the Pros and Cons of T-F Domains Speech Enhancement

Speech sounds and interferences usually exhibit rich auditory patterns in the T-F spectrogram. In this section, we intend to understand the impact of phase in the T-F spectrogram to enlighten our multi-domain model design.

How does the T-F spectrogram phase affect the speech intelligibility and quality? We first conduct an experiment by using the UltraSpeech dataset (detailed in Section 4.9), where we keep the clean speech’s amplitude in the T-F spectrogram while replacing its phase with the noisy speech phase, just as in aIRM (Sec. 4.2). We use two metrics to evaluate the impact. (i) *Scale-invariant Signal-To-Noise Ratio (SiSNR)* characterizes the speech *quality* [128]:

$$\mathcal{L}_{SiSNR} = 10 \log_{10} \left(\frac{\left| \left| \frac{\langle \hat{s}, s \rangle}{\|s\|^2} \right\|^2}{\left| \left| \frac{\langle \hat{s}, s \rangle}{\|s\|^2} - \hat{s} \right\|^2} \right)} \right) \quad (4.2)$$

where s and \hat{s} are the T domain clean speech and enhanced speech signals, respectively. (ii) *Word Error Rate (WER)*, representing speech *intelligibility*, is the probability that a word cannot be correctly recognized by an automatic speech recognition (ASR) algorithm [11] and human perception.

As shown in Figure 4.9(a), when applying the noisy T-F spectrogram phase directly, the

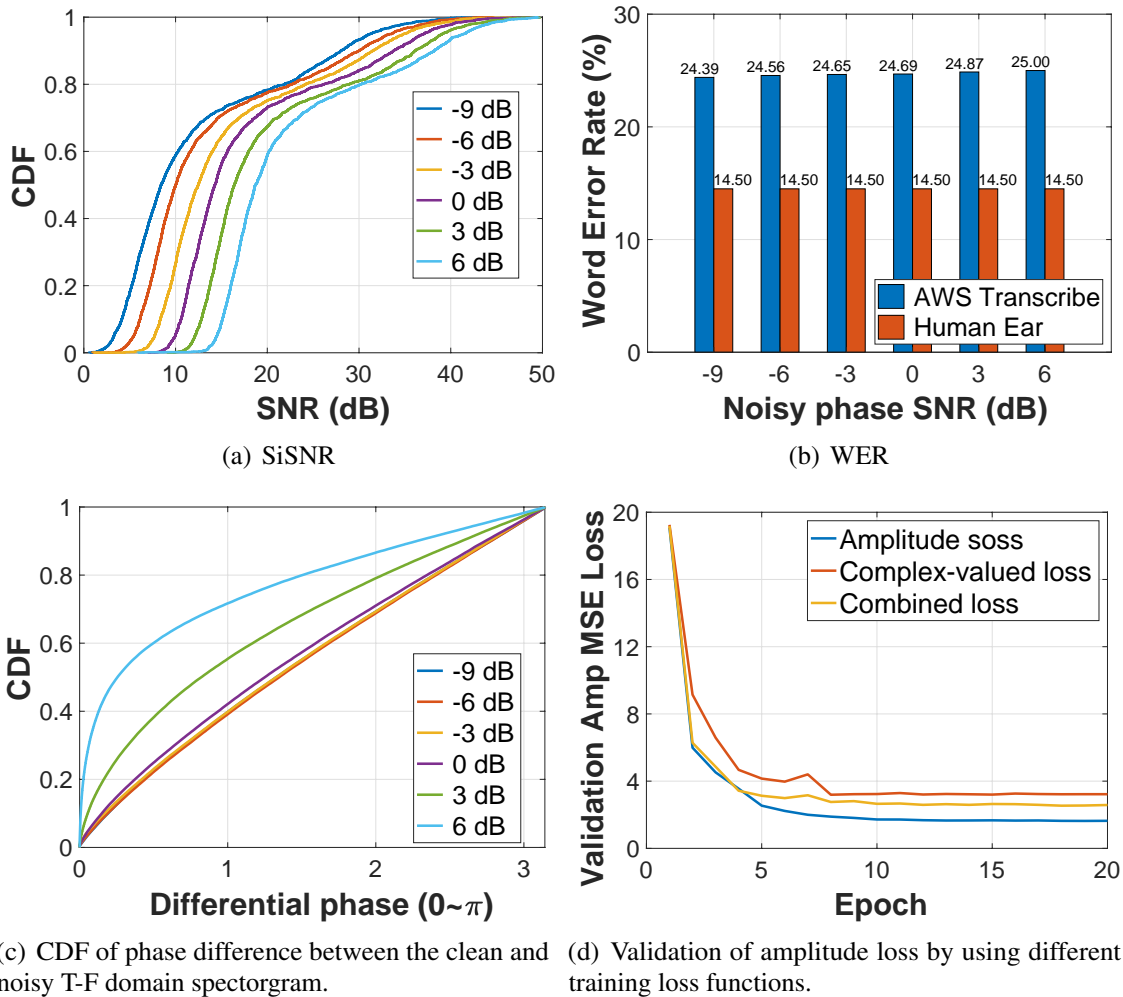


Figure 4.9. Benchmark of the T-F domain methods.

SiSNR degrades slightly. On the other hand, phase does not affect the WER in a noticeable way. The noisy phase with a very low SNR of -9 dB only decreases the WER by 0.7% when using AWS Transcribe [11]. Meanwhile, human subjects can clearly understand the speech and only feel a little jittering effect. In summary, *the phase in the T-F spectrogram barely affects the speech intelligibility and only slightly degrades the speech quality.*

What is the appropriate training loss function for recovering the speech intelligibility?

Figure 4.9(c) plots the CDF of phase difference between the clean and noisy speech spectrogram across the T-F bins. We see that the phase difference is almost uniformly distributed for low-SNR

speech. This means the phase values in all the T-F bins are distorted in the spectrogram which makes the phase recovery challenging. Since phase is not critical to intelligibility, we proceed to study the performance of different DNN loss functions in recovering the T-F spectrogram *amplitude*.

We examine 3 different loss functions. The first is the amplitude MSE loss which only considers the T-F spectrogram amplitude: $\mathcal{L}_a = \lambda \|G(U^s, S_{noise}^a) - S_{gr}^a\|^2$. The second is the complex-valued MSE loss which accounts for both the T-F spectrogram amplitude and phase: $\mathcal{L}_p = \|S_{out}^c - S_{gt}^c\|^2$. The third is a combined loss used in PHASEN: $\mathcal{L}_{combined} = 0.5 \times \mathcal{L}_a + 0.5 \times \mathcal{L}_p$, where S_{out}^a, S_{gt}^a and S_{out}^c, S_{gt}^c are the power-law compressed ($A^{0.3}$) amplitude spectrogram and complex-valued spectrogram. We apply these 3 training loss functions to the architecture in Section 4.6 and 4.7. Figure 4.9(d) shows the validation amplitude MSE loss. Obviously, upon convergence, training with amplitude MSE loss leads to lower validation error in amplitude MSE, and hence better speech intelligibility, than the two alternative loss functions.

4.8.2 Two-stage Multi-domain Network Design

Based on the above studies, we derive 3 design principles for our multi-domain architecture: (i) The T-F spectrogram amplitude contributes to the speech intelligibility whereas the phase is related to the speech quality. (ii) The T-F spectrogram phase is hard to predict by using DNN models. (iii) Training DNN models with aIRM MSE loss in the T-F domain optimizes speech intelligibility. We now elaborate on the detailed design, which follows the flow in Figure 4.4.

Stage 1: T-F domain multi-modal amplitude speech enhancement. The DNN architecture and training model of this stage has been covered in Sec. 4.6 and Sec. 4.7. The amplitude-enhanced T-F spectrogram output is multiplied with the original noisy phase to generate a complex-valued T-F spectrogram. Then, the iSTFT [91] is used to transform the T-F spectrogram to the T domain waveform and output the *amplitude-enhanced T-domain waveform*.

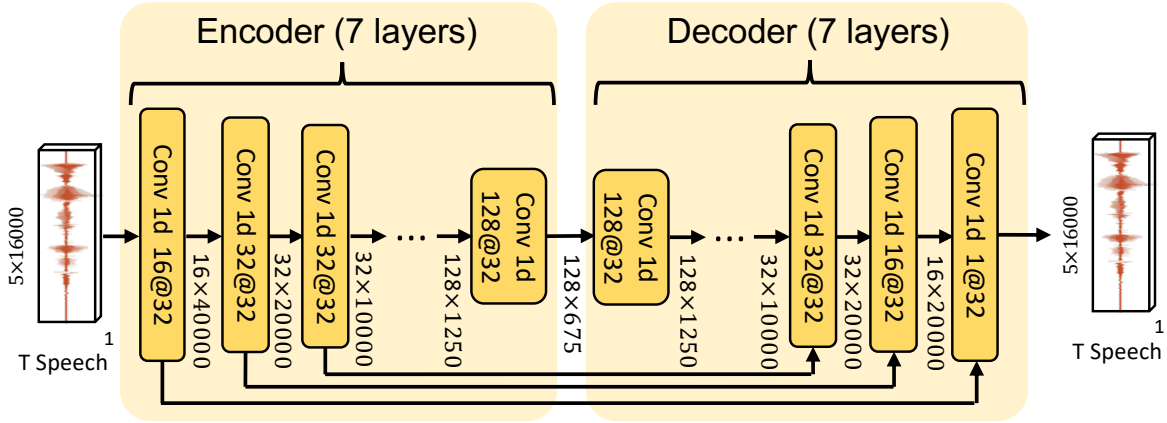


Figure 4.10. T domain phase network. Channels@Kernel size in convolution layer.

Stage 2: T domain speech phase enhancement. The goal of this stage is to fine tune the T-domain waveform to further improve the speech *quality*, using the SiSNR (Eq. (4.2)) as the training loss function. Inspired by SEGAN [179], our T domain network is an encoder-decoder network as shown in Figure 4.10. The encoder contains 7 1D convolution layers to transform 5 s of time domain waveform to a 128×675 scalar. The decoding stage reverses the encoding operation by means of fractional-strided transposed convolutions. We connect each encoding layer to its homologous decoding layer to fully capture the low-level details of the original features. The network parameters are listed in Table 4.3. All the 1D convolutional layers are followed by parametric rectified linear units (PReLU) [100]. We also tried a cGAN training model similar to Section 4.7 in this stage, but observed negligible performance gain. Thus, we only enforce the cGAN training in the T-F domain.

Notably, the first and second stage output can be used to satisfy different applications, *e.g.*, for ASR and human listener, since they are trained for speech intelligibility and quality, respectively.

4.9 UltraSE Implementation

4.9.1 UltraSpeech Dataset

Traditional speech datasets only contain raw speech without ultrasound sensing signals [81, 176]. To evaluate UltraSE, we thus create a new dataset called UltraSpeech which comprises both.

Data collecting: We recruited 20 fluent English speakers (4 female, 16 male, average age 25) to collect the UltraSpeech dataset. Each participant was asked to say at least 300 sentences in the TIMIT speech corpus [81] by using 2 typical phone holding styles (“Phone Call” mode and “Towards Mic” mode, shown in Figure 4.12(b)) in quiet environment. Meanwhile, we use a custom-built Android app called UltraRecord, to emit the ultrasonic signals and capture the audio segments at 96 kHz sampling rate, through the bottom speaker and microphone on a smartphone. Note that we do not constrain the user to hold the smartphone at a specific distance from the mouth. In total, we collected 8k 5-second clean speech segments for each holding style.

We follow existing SSE work [71, 274] to generate the noisy speech dataset through synthetic mixture. The interfering speech comes from the TIMIT data set [81], which contains 6300 different English sentences, generated by 630 speakers lasting 3.5 hours in total. The ambient noise dataset comes from AudioSet [82] which contains more than 1.7 million 10-second segments of 526 types of noise from real-life, including a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds.

Training/testing dataset generation: Each segment of training/testing data is synthesized by a linear combination of 3 pieces: $\langle S_j, U_j, S_{i_{noise}} \rangle$, where S_j and U_j are the clean speech segment and corresponding ultrasound features from UltraSpeech; $S_{i_{noise}}$ is the i_{th} noisy sound segment.

Besides, we generate a training set where the interfering speech and clean speech come from the same speaker. This is widely recognized as the *most challenging case* of SSE [77], since the interference bears the same auditory patterns that are indistinguishable from the desired

speech. We add this into the training dataset to force the model to exploit the ultrasound features in addition to the audible features.

Our training dataset contains 15 participants' clean speech collected by the Samsung Galaxy S8 smartphone. Each participant's clean speech is mixed with 20 different noise settings. For each noise setting, the number of interfering speakers n is uniformly distributed in $[0, 4]$, and the SNR is uniformly distributed in $[-9, 6]$ dB (-1.5 dB average). In total, the training data contains 120k 5-second segments of noisy speech (300 hours).

4.9.2 UltraSE DNN Implementation

We implement the UltraSE DNN model in Pytorch. The dimension of feature maps and the parameters of each layer are shown in Figure 4.4, 4.5, 4.10 and Table 4.1, 4.2, 4.3. ReLU activations follow all layers except for the last layer, where a sigmoid is applied. For training, we use Adam optimizer with a $1e - 04$ initial learning rate, dropping by 25% every 5 epochs for a total of 20 epochs. UltraSE has 15.5 M and 3.1 M parameters for the first and second stage DNN.

4.10 Experimental Evaluation

We evaluate UltraSE using 4 metrics commonly adopted in SSE research.

- SDR [237]: Signal-to-distortion ratio, which considers not only noise/interference, but also acoustic artifacts (*e.g.*, burbling sound) as distortion to the ground-truth speech;
- SiSNR [145]: Scale-invariant signal-to-noise ratio (Sec. 4.8.1) which, unlike the classical SNR, ensures rescaling the estimated signal will not unfairly improve the metric;
- STOI [220]: Short-time objective intelligibility measure (from 0 to 1);
- PESQ [187]: Perceptual evaluation of speech quality, which models the mean opinion score ranging from 1 (bad) to 5 (excellent);

4.10.1 Micro Benchmark Comparison

In this section, our default testing dataset includes another 5 participants’ clean speech in the “Towards mic” mode, collected using Samsung S8. Our testing environment includes 6 different interference plus noise settings: $1s + a$, $2s + a$, $3s + a$, $> 3s + a$, $2s$ (“s” and “a” denotes interfering speaker and ambient noise) and the hardest case ≥ 2 same-speaker interferences plus noise ($\geq 2ss + a$). The SNR level of noisy speech signals is uniformly distributed in $[-9, 6]$ dB. All the results of UltraSE are from a single model generated from the training dataset.

Table 4.4. UltraSE micro benchmark.

Environment	Methods	SDR	SiSNR	STOI	PESQ
$1s + a$	UltraSE	17.14	17.25	0.87	3.52
	PHASEN	15.63	15.20	0.82	3.05
	SEGAN	5.48	5.50	0.64	2.32
	AVSPEECH	16.0	/	/	/
	Conv-TasNet	12.23	12.58	0.76	2.48
$2s + a$	UltraSE	10.55	10.65	0.76	2.80
	PHASEN	5.20	5.22	0.65	2.23
	SEGAN	2.01	1.96	0.54	1.69
	AVSPEECH	10.1	/	/	/
	Conv-TasNet	10.23	10.38	0.74	2.40
$3s + a$	UltraSE	10.88	10.94	0.76	2.81
	PHASEN	5.14	5.15	0.66	2.15
	SEGAN	1.74	1.78	0.55	1.68
	Conv-TasNet	6.31	6.50	0.71	2.11
$> 3s + a$	UltraSE	12.10	12.17	0.78	2.66
	PHASEN	5.13	5.13	0.67	2.14
	SEGAN	0.71	0.72	0.53	1.67
	Conv-TasNet	6.23	6.41	0.71	2.15
$\geq 2ss + a$	UltraSE	8.90	8.97	0.72	2.52
	PHASEN	5.03	5.05	0.62	2.10
	SEGAN	1.27	1.29	0.56	1.69
	Conv-TasNet	5.69	5.93	0.73	2.21
$2s$	UltraSE	14.85	14.86	0.86	3.35
	AVSPEECH	10.3	/	/	/
	Conv-TasNet	14.98	15.02	0.85	2.97

We compare UltraSE with 4 state-of-the-art SSE methods, PHASEN [274] (T-F domain method), SEGAN [179] (T domain method), AVSPEECH [71] (Audio-visual method), Conv-TasNet [146] (Speech separation method). For a fair comparison, we reimplemented PHASEN, SEGAN and Conv-TasNet and train and test them on the UltraSpeech dataset. PHASEN and SEGAN only use the $1s + a$ training set, since they are designed for speech enhancement, not separation. The results for PHASEN and SEGAN under $1s + a$ (see Table 4.4) is similar to the original work, which shows the correctness of our implementation. For the speech separation method, *i.e.*, Conv-TasNet, we first train and evaluate it in the “ $2s$ ” environment to check the correctness of our implementation. Then, we use the “ $2s + a$ ” dataset to train the model with the 2 speakers’ clean speech as ground truth, and compare the results in other environments in Table 4.4. For AVSPEECH, since our data set does not have the video recordings, we directly use the results in [71] as baselines.

Compared to the state-of-the-art speech enhancement methods, UltraSE significantly improves the speech quality and intelligibility in both noisy and multi-speaker environments. Table 4.4 shows the testing results under all input SNR levels uniformly distributed in $[-9, 6]$ dB. UltraSE outperforms PHASEN and SEGAN across all the 4 metrics. In the $1s + a$ environment, UltraSE achieves an average 17.25 SiSNR (18.75 Δ SiSNR) and 3.50 PESQ. In other environments with multi-speaker interference, the ultrasound sensing modality plays a more prominent role, improving SiSNR by 6.04 dB and 9.77 dB on average over the 2 baselines respectively. Even for the hardest case $\geq 2ss + a$, UltraSE still achieves 8.97 dB SiSNR and 2.52 PESQ. In addition, UltraSE achieves slightly higher performance than AVSPEECH, likely because the ultrasonic features are sampled at finer time granularity than video frames, and can better align with the speech signals.

Most of the existing speech separation methods can only work with limited number of interfering speakers (2 ~ 3) and without ambient noise [104, 276, 145, 266]. As shown in Table 4.4, when training the Conv-TasNet by using the “ $2s + a$ ” dataset, Conv-TasNet achieves good performance in the “ $2s + a$ ” and “ $2s$ ” setup, but is not general in other sophisticated environments.

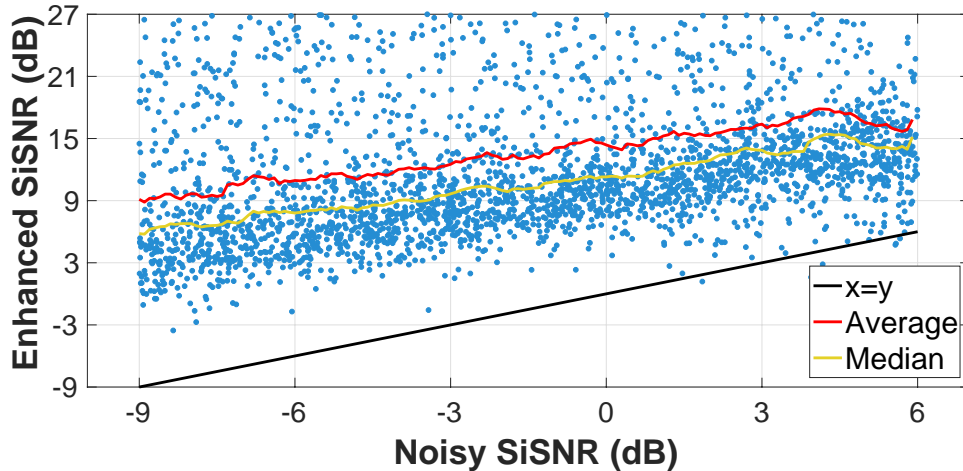


Figure 4.11. Noisy SiSNR v.s. Enhanced SiSNR

In comparison, UltraSE outperforms Conv-TasNet by around 6 dB of SDR or SiSNR, 10% in STOI and 24% in PESQ, under the $> 3s + a$ setup.

The scatter plot in Figure 4.11 shows the input and output SiSNR for each sentence in the testing dataset which includes all 6 environments. UltraSE consistently achieves high performance across different environments and sentences, with an average 14.75 dB SiSNR gain. Even in the worst case with -9 dB input, the enhanced speech achieves 8.86 dB SiSNR on average.

4.10.2 Ablation Study

We conduct an ablation study to better understand the performance of different design components in UltraSE. The testing dataset here includes all the environments except the “ $\geq 2ss + a$ ” which is not very common in practice. Table 4.5 summarizes the results.

“No T domain” represents the DNN model without the “T domain waveform speech enhancement”. The results indicate that this module barely influences the STOI, a metric for speech intelligibility. But it helps gaining 0.46 dB SDR, 0.58 dB SiSNR, 0.12 PESQ respectively, which proves it can further improve the perceptual quality of the speech generated from the T-F domain multi-modal network.

“No cGAN” represents the model without the “cGAN-based cross-modal model training”.

All the metrics significantly improves when applying the cGAN, since our cGAN design forces the network to learn the correlation between the ultrasound and speech, which is the key principle behind the UltraSE design.

“No Fusion Network” means that the feature maps of ultrasound and speech signals are directly concatenated in the T-F domain without the fusion block. The performance slightly decreases, since the fusion block helps the multi-modal features to “cross-talk” with each other.

“No Ultrasound” represents the network without the ultrasound stream at the beginning of the network. The result becomes close to the traditional speech enhancement method without ultrasound sensing, *e.g.*, PHASEN.

4.10.3 System Efficiency

Time Consumption: We evaluate the run-time processing latency of UltraSE on 3 platforms, including a NVIDIA GTX 2020 (GPU), an Intel i9-9980 3.00GHz (CPU) and Samsung Galaxy S8 with Qualcomm Snapdragon 835 CPU (Smartphone). The first two correspond to the case where UltraSE is offloaded to a trusted cloud or edge server. Table 4.6 summarizes the results. The GPU server only experiences 14.85 ms latency which is acceptable for VoIP applications (150 ms maximum [132]). The smartphone case is measured by using Pytorch Mobile [20] on Samsung Galaxy S8. Note that the latest version of Pytorch Mobile [20] only supports *single-CPU processing without any GPU/NPU support*. Thus, the latency is relatively high (25.08 s to process 5 s speech), which is acceptable only for offline processing applications, *e.g.*, audio message and audio recording. There exists a rich literature [246] on improving DNN efficiency on smartphones, which demonstrated more than 50× latency reduction by using mobile GPU/NPU. We will explore such solutions for our future work. Also note that UltraSE needs to process the input in segments of 5 s due to the use of Bi-LSTM blocks. This means its SSE starts taking effect after a 5 s initial bootstrapping period.

Energy Consumption: Our experiments show that a typical smartphone (Samsung S8) can continuously use UltraSE to record speech while emitting ultrasound signals for 60.57 hours

Table 4.5. UltraSE ablation study.

	SDR	SiSNR	STOI	PESQ
UltraSE (Testing data 96 kHz)	13.10	13.21	0.80	3.01
UltraSE (Testing data 48 kHz)	13.08	13.18	0.79	2.99
- No T domain	12.64	12.63	0.80	2.89
- No cGAN	10.80	10.85	0.77	2.60
- No Fusion Network	9.96	10.00	0.76	2.54
- No Ultrasound	7.78	7.68	0.70	2.39

Table 4.6. Inference time for processing 5 s speech.

	Preprocess	Stage 1	Stage 2
GPU	0.55 ms	12.02 ms	2.28 ms
CPU	0.05 s	1.38 s	0.26 s
Smartphone	0.25 s	23.02 s	1.81 s

with display off. Our measurement using Android Profiler [21] reveals that UltraSE’s CPU load is 48.7% on average, and power consumption is at the level of “1” in between the scale of 0 to 3. When offloading to servers, the computational energy consumption becomes negligible. The only overhead is that UltraSE needs to upload the original 48/96 kHz sampling rate audio stream with both audible sounds and ultrasounds to the server, and then download the enhanced speech from the server. Our experiments show that Samsung S8 can continuously run UltraSE and upload/download the audio streaming via WiFi in the offloading mode for 10.82 hours. Server offloading may incur additional issues such as security, but this is beyond the scope of our current work.

4.10.4 Generalization

Sampling Frequency: UltraSE model trained by 96 kHz sampling rate dataset can be directly used to enhance the testing speech recorded at 48 kHz sampling rate. The feature resolution at 48 kHz sampling rate is identical to the case at 96 kHz sampling rate as long as

the FFT window length and hop length of ultrasound sensing features keep 85 ms and 10 ms respectively. Table 4.5 shows a negligible performance degradation when testing the 48/96 kHz sampling rate dataset on the 96 kHz sampling rate trained model.

Holding Styles: In the “Phone call” mode (Figure 4.12(a)), the user’s face partially occludes the ultrasonic signals, so we train a model which is different from the “Towards mic” mode (Figure 4.12(c)). UltraSE can automatically select the model using the IMU-based holding style detection algorithm built into smartphones [83]. Our experiments show that, under -1.5 dB average input SNR, the performance of “Phone call” (12.47 dB SiSNR) is slightly lower than the “Towards mic” (13.12 dB SiSNR) due to the occlusion.

We further evaluate the sensitivity of each model under different mouth-to-mic distances. Figure 4.12(b) and Figure 4.12(d) show the average SNR of ultrasound (SNR_g) vs. the SiSNR of enhanced speech. For both holding styles, SNR_g well exceeds 10 dB, and speech SiSNR stays around 12 dB within 20 cm distance. *The experiment implies that the UltraSE model performs consistently as long as the mouth-to-mic distance remains within 20 cm.*

Motion interference: We measure the impacts of interference from 3 major motion artifacts, *i.e.*, respiration, hand gestures and walking. The experiments were conducted when the mouth is 15 cm and 2 cm away from the microphone in the “Towards mic” and “Phone call” mode, respectively. *(i)* The respiration frequency (~ 30 bpm) is far less than the articulatory motions (> 10 Hz), so it creates negligible impacts on UltraSE. *(ii)* Hand gestures introduce similar Doppler effect as the articulatory motion [165, 248, 149], which may cause non-negligible interference. We measure the articulatory gestures’ SNR_g under the pushing hand gesture interference. The SNR_g is sampled for each 2 cm in 7 different angles from 0° to 90° , at a step of 15° , close to the user’s mouth. Figure 4.13 shows the spatial distribution [68] of SNR_g for the “Towards mic” mode. As long as the hand gesture is > 25 cm away from the mouth (which is typical in daily usage scenarios), the SNR_g remains above 10 dB which suffices for UltraSE (Figure 4.12). A microphone array can be used to focus on the user’s mouth region to further mitigate interference [149], but this is beyond the scope of UltraSE. We omit the “Phone call”

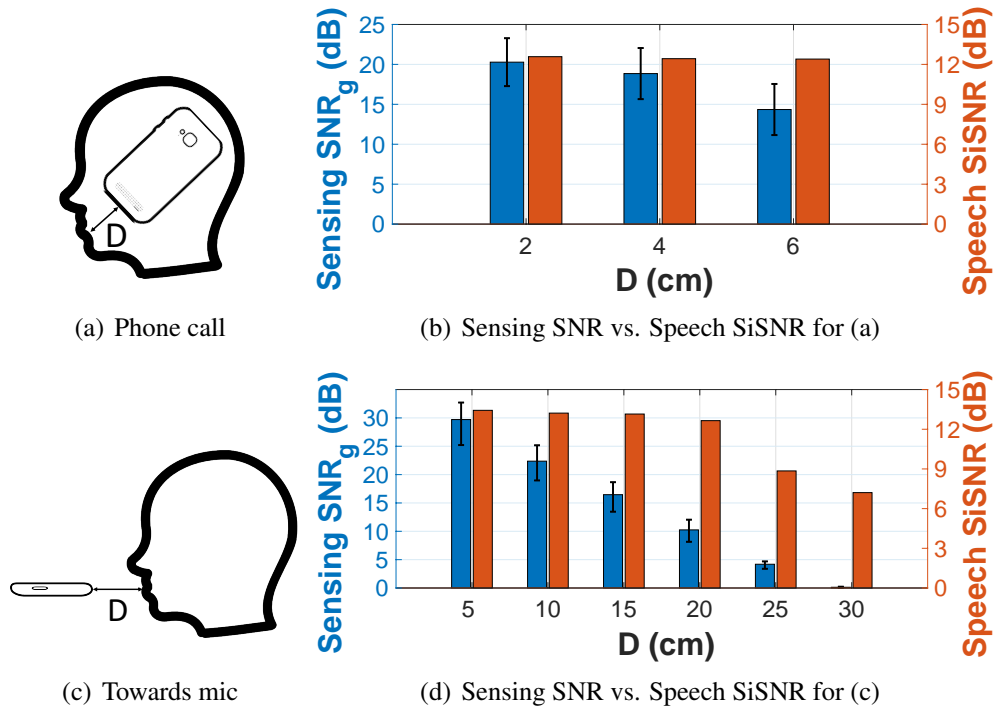


Figure 4.12. SNR of articulatory gestures.

mode since the microphone is much closer to the mouth and the sensing SNR_g stays high. (iii) When other people walk nearby (0.8 m away), we found that SNR_g is barely impacted since the ultrasound volume is relatively low, and the user’s mouth is much closer.

Overall, the articulatory gestures’ SNR_g is sufficiently high (> 10 dB), and the UltraSE model is unaffected by the motion artifacts in daily usage scenarios.

Generalizations across smartphones: Different smartphones may have different speaker and microphone layout. For example, the distances between the bottom microphone and speaker are 5 mm, 25 mm and 25 mm for Samsung S8, LG G8S ThinQ and VIVO X20 respectively. The high-frequency response of the speaker and microphone may also vary across phone models [19]. When applying the DNN model trained by the Samsung S8 dataset directly to LG G8S ThinQ and VIVO X20, the SiSNR of enhanced speech becomes 9.21 dB and 9.53 dB, respectively. which is lower than the same-phone case (13.21 dB), but still higher than the SiSNR without ultrasound sensing (7.68 dB). To maintain the optimal performance, a straightforward way is

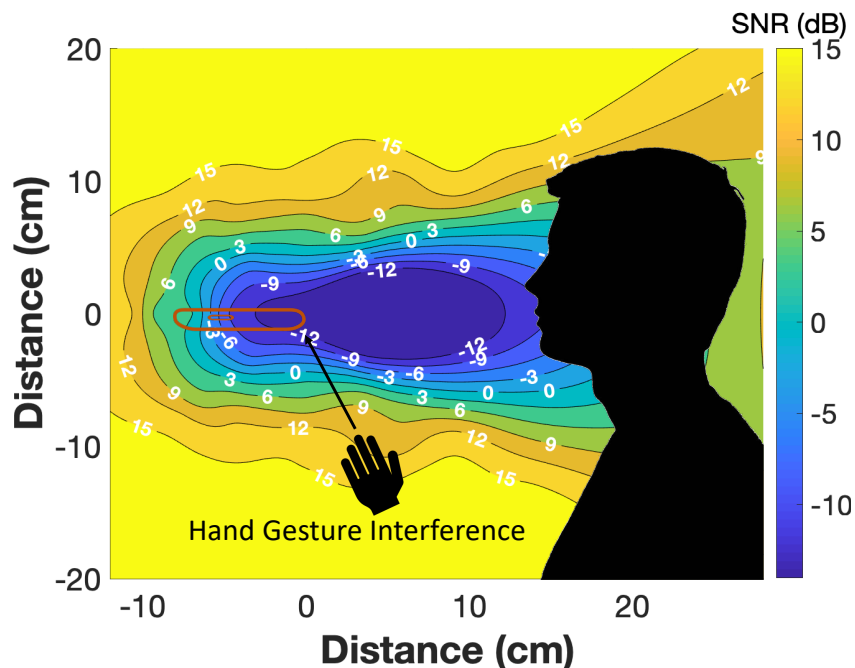


Figure 4.13. SNR_g under hand gesture interference

to perform a one-time training data collection for each phone model. Alternatively, we can enrich the UltraSpeech dataset with a diverse set of smartphones that cover the typical hardware configurations. This is left for our future work.

Real-world Usage Experiments: We asked the users to use UltraSE across 4 different real-world environments, *i.e.*, 1). a bathroom environment with exhaust fan and running water noise (75 dBA on average); 2). a living room environment with television noise (55 dBA on average); 3). an indoor conference environment with conversation noise (60 dBA on average); 4). an outdoor roadside environment with vehicle noise (60 dBA on average). Unlike synthetic noisy speech, we can not capture the ground truth clean speech and evaluate the metrics like SDR, SiSNR, STOI and PESQ in these scenarios. Thus, to evaluate the performance of UltraSE for real-world usage, we use the ASR Word Error Rate $WER = \frac{S+D+I}{N}$ as the metric, where S , D , I , and N are the number of substitutions, deletions, insertions, totals of targeted user's spoken words respectively. Specifically, we asked the users to speak at least 50 sentences in the TIMIT speech corpus [81] across different environments. Figure 4.14 shows the WER

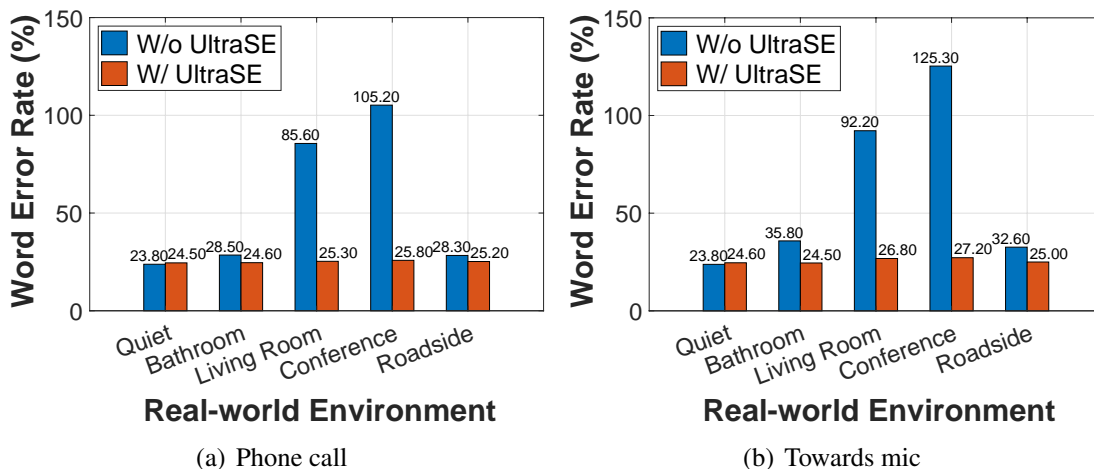


Figure 4.14. Real-world Usage WER.

with and without UltraSE across different environments. In non-speech noisy environments, *i.e.*, bathroom and roadside, UltraSE slightly improve the ASR speech recognition rate since ASR itself has the ability to mitigate background ambient noise interference. In speech noisy environments, *i.e.*, living room and conference, WER is higher than 100% since there exists many word insertions and substitutions introduced by non-targeted user’s speech. UltraSE achieves significantly improvement in such cases since it is able to separate the desired speaker voice from noises by using ultrasound sensing.

4.11 Conclusion

We have demonstrated that ultrasonic sensing can serve as a complementary modality to solve the cocktail party problem. Our UltraSE system introduces general DNN mechanisms to enable such capabilities, *e.g.*, multi-modal multi-domain fusion network and cGAN-based training model based on a novel cross-modal Siamese network. UltraSE points to a novel direction that fuses wireless sensing capabilities to bring machine perception to a new level.

4.12 Acknowledgments

This chapter contains material from “UltraSE: Single-Channel Speech Enhancement Using Ultrasound”, by Ke Sun, and Xinyu Zhang, which appears in the 30th annual International Conference on Mobile Computing and Networking (MobiCom), 2021 [215]. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Multimodal Daily-life Logging in Free-living Environments Using Non-Visual Egocentric Sensors on a Smartphone

5.1 Introduction

The emerging Internet of Things (IoT) promises to embed a massive population of sensors in the environment to form an *ambient intelligence* [97]. Such omnipresent IoT sensors can generate huge personalized data to enable life-logging and support many activity-aware applications. In particular, they can monitor a subject's activities of daily living (ADL), including not only body motion (*e.g.*, walking, bathing), but also interaction with the physical environment and daily objects (*e.g.*, kitchen appliances, water cups, faucets). They can transform the healthcare domain which, to date, has been relying on laborious monitoring and subjective questionnaires/reports for diagnosis, assessment, and emergent response. Examples include tracking medical compliance, evaluating rehabilitation (*e.g.*, for stroke patients), detecting onset of chronic diseases (*e.g.*, the Alzheimers' disease), *etc.*

To approach the vision of ubiquitous ADL sensing, substantial research has investigated the egocentric sensing scenario, where the sensors are co-located with subjects [164]. In particular, egocentric visual sensing relies on a head-mounted camera or smart glasses to capture the first-person views [60, 201, 89]. Due to the limited angle of view and constrained wearing

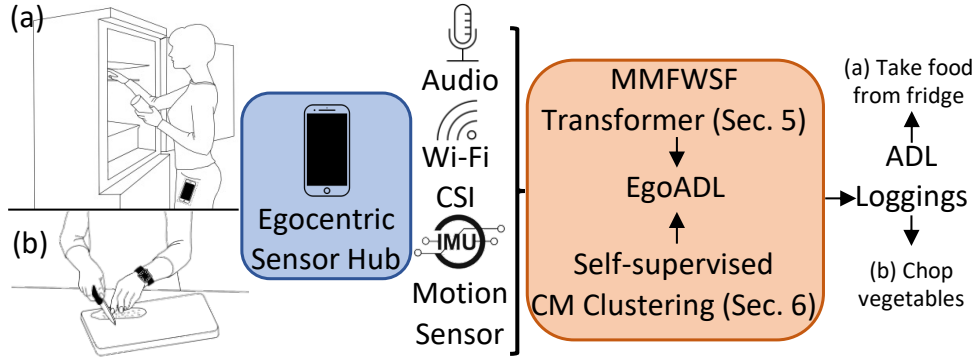


Figure 5.1. EgoADL is an egocentric ADL sensing system, leveraging an on-body smartphone as a sensor hub to capture the audio, Wi-Fi CSI, and motion sensor signals simultaneously. EgoADL employs a self-supervised cross-modal (CM) clustering to encode a general feature representation from the large-scale unlabeled data (Sec. 5.6) and a supervised Multi-Modal Frame-Wise Slow-Fast (MMFWSF) transformer model to recognize the ADLs (Sec. 5.5).

style, they can only partially capture the user’s ambulatory activities, leading to low accuracy [201, 157]. Furthermore, these approaches inherit the limitations of camera sensing—They are privacy intrusive, energy hungry, and crippling for continuous sensing. On the other hand, egocentric non-visual sensors such as wearable motion sensors are limited to capturing only ambulatory actions without the interaction with the physical environments and daily objects [53]. Therefore, audio [116, 49], and motion sensor [270, 163] are typically used to assist the egocentric video to improve accuracy.

In this paper, we design EgoADL, a multimodal ADL sensing system that employs ubiquitous non-visual sensors on an egocentric in-pocket smartphone to log ADL in free-living environments. Compared with vision-based approaches, EgoADL is less intrusive and better approximates Mark Weiser’s pioneering definition of ubiquitous intelligence [256], *i.e.*, sensing technologies that quietly serve human in the background. As shown in Fig. 5.1, a user performs daily routines with the sensor hub, *i.e.*, an in-pocket smartphone. EgoADL is designed to log a comprehensive range of basic ADL, which are characterized by open-ended, fine-grained activities encompassing both body motion and human-object interactions. These activities include, but are not limited to, routine physical movements (*e.g.*, walking, sitting, bending down), common household tasks (*e.g.*, cooking, washing dishes, mopping floor), and interactions with

various objects in daily life (*e.g.*, using utensils, chopping vegetables, taking food from fridge) (see Fig. 5.6 and Fig. 5.7 for detailed ADL list). The focus is on detailing these everyday actions in a manner consistent with the definitions used in computer vision-based ADL recognition [203, 202, 89, 22].

To realize EgoADL, we resolve three key challenges:

ADL representation–beyond activity classification. Most of the traditional DNN-based ADL analysis models are designed to perform *classification* [60, 202], which assigns integer IDs to various ADL. However, it requires prescribing a known set of ADL, which falls short of extensibility when new ADL of interest emerge. In contrast, EgoADL is designed to enable comprehensive daily life logging for humans, covering a wide spectrum of distinct ADLs. Therefore, we propose to use a transformer-based sequence-to-sequence model to decode these feature representations as label name semantics using a sequence of words. Moreover, these ADL representation establishes a connection between sensor data and natural language. This enables us to harness the semantic information inherent in these natural language labels. By seamlessly integrating this information with language models, we significantly elevate the overall performance of EgoADL.

Egocentric multi-modal fusion–overcoming limited resolution of non-visual sensors. The second major challenge is that the non-visual sensors have much lower resolution than camera, for sensing both human body motion and interaction with daily objects [217]. To overcome the challenge, we select and synthesize 3 specific modalities which have already been embedded in commercial devices, *i.e.*, motion sensor for ambulatory actions of leg; wireless sensing for full-body motion and interactions with ambient environments; audio recording for motion and human-object interaction with unique sound events. Our empirical analysis of real-world ADL data indicates that each sensing modality is amenable to different ADL patterns, and a judicious combination of them can potentially achieve near-vision resolution. Therefore, we propose a Multi-Modal Frame-Wise Slow-Fast (MMFWSF) encoder to learn the feature representation of multi-sensory data, which characterizes *multi-modal fast-changing*

motion, continuous scene sounds, and cross-modal frame-wise alignment. We then use a transformer-based sequence-to-sequence model to decode these feature representations as label name semantics using a sequence of words.

Self-supervised ADL learning—achieving generalization with limited labeled data. To achieve high sensing accuracy, extensibility for non-frequent ADL and generalization (across different ADL, users and environments), EgoADL needs a massive amount of training data, which entails exorbitant labeling cost due to the high variability and “invisibility” of the sensing records. We observe that capturing the data without labeling is relatively easy for EgoADL, since all the sensors are embedded in an in-pocket smartphone. Thus, we collect large-scale *unlabeled* data and adopt a self-supervised learning (SSL) framework to encode a general feature representation. Specifically, we design a cross-modal deep clustering model that extrapolates two self-supervisory signals from unlabeled data: *i)* Audio captures human-object interaction which can inform the motion sensor and Wi-Fi CSI. *ii)* Correspondence between different modalities when observing the same ADL. In addition, we leverage the vast amount of existing labeled audio datasets [82] to pretrain a feature embedding DNN. These datasets already encompasses the natural logic of ADL, thus further alleviating the training workload of the self-supervised ADL learning.

To validate the design principles behind EgoADL, we implement an Android app to collect the multi-modal data from in-pocket smartphones, when users freely perform daily activities. Our implementation leads to the *first* non-visual multimodal dataset for egocentric ADL. The dataset consists of large-scale unlabeled data, along with a labeled subset for users who are willing to wear a head-mounted camera to capture the ground truth. We implement a labeling software that allows the users to playback the audio/video recordings and annotate the sensor data accordingly.

Using this platform, we have collected 20 hours of labeled data and more than 100 hours of unlabeled data, which includes 221 different types of ADL involving 70 actions and 91 objects. The data was collected from 20 different home environments and 30 users who performed an

unrestricted set of activities, encompassing both ambulatory motion and interaction with daily objects. Our evaluation results show that EgoADL achieves 72.5% top-1 and 90.8% top-5 mean Average Precision (mAP) for recognizing 105 frequently-used ADL, which are 21.0% and 14.7% higher than the baseline model using traditional modal-wise sensor fusion. When considering the 35 state-based ADL which typically last more than 5 seconds, EgoADL achieves 85.9% top-1 and 94.5% top-5 mAP. Our results suggest that EgoADL can achieve comparable performance with vision-based egocentric sensing, particularly for non-ambiguous actions and objects using non-visual sensors.

The main contributions of EgoADL are as follows.

- We introduce a new concept of multi-modal egocentric ADL sensing based on non-visual sensors on in-pocket smartphones. We build a platform for EgoADL sensor data collection and labeling, and establish the dataset for multi-modal egocentric ADL sensing by using non-visual sensors. Both the platform (<https://github.com/Samsonsjarkal/EgoADL>) and dataset (<https://doi.org/10.5281/zenodo.8248159>) are released to facilitate further research.
- We design multi-modal fusion approaches to learn the feature representation of multi-sensory data by leveraging the complementary advantages of audio, motion sensor and wireless sensing.
- We propose an SSL framework that extrapolates single-modal and multi-modal supervisory signals from unlabeled data, in order to boost the model accuracy, generalization and extensibility.
- We propose to leverage the semantic information from the natural language labels by distilling knowledge from external text datasets and refining the labels to fit the sensing capability.

Table 5.1. Representative ADL systems using Wi-Fi CSI, IMU and Audio. In “Scenario” column, Ego: Egocentric; DF: Device-free. In “Task” column, HA: Human Activities; HOI: Human-Object Interaction. HPC: Human Pose Construction; SE: Sound Event; SEDL: Sound Event Detection and Localization; In “Method” column, SM: Single-Modal; MM: Multi-Modal; SL: Supervised; SSL: Self-Supervised Learning. In “User Study” column, S: Subjects; E: Environments. This table focuses exclusively on ADL systems employing Wi-Fi CSI, IMU and Audio. There may exist additional sensors applicable in ADL systems, such as PIR sensors, magnetic sensors, sonar sensors, *etc.*[217]

	Modality	Scenario	Range	Task	# of Activities	Method	User Study	Receiving Sensors
E-eyes [251]	WiFi CSI	DF	Apartment	HA Classification	9	SM+SL	1 S, 2 E	1 WiFi AP
CARM [247]			Single Room					
EI [110]			Single Room					
WiPose [111]			Static					
RF-Diary [74]	FMCW radar	DF	Single Room	HA + HOI Captioning	157 Acts, 38 Objs	SM+S	10 S 10 E	FMCW Radar (need floormap)
DeepSense [272]	Motion sensor	Ego	/	HA Classification	6	SM+SL	9 S	Wearable Devices
LIMU [269]		Ego	/	Classification	7	SM+SSL	73 S	Wearable Devices
DESED [231]	Audio	DF	/	SEDL	10	SM+SL	/	Mic Array
Ubicoustics [126]		Ego /DF	/	SE Classification	30	SM+SL	7 S	Wearable devices
Cosmo [173]	Ego For motion sensor DF For Depth Cam, Radar	Ego	/	HA Classification	14	MM+SSL	30 S 1 E	Wearable devices, Radar, Depth Camera
Wrist-ADL [41]	Audio + Motion sensor		/	HA Classification	23 Acts	MM+SL	15 S 15 E	Smartwatch
EgoADL	WiFi CSI + IMU + Audio	Ego	Whole Apartment	HA + HOI Captioning	221 Acts 91 Objs	MM + SSL	30 S 20 E	Smartphone

5.2 Related work

Egocentric vision-based ADL sensing: The wide availability of head-mounted or body-worn cameras has resulted in massive first-person vision data, and fueled research in egocentric ADL analysis [201, 60, 136, 89]. However, egocentric vision approaches still face fundamental deployment barriers. In particular, wearing a camera is inconvenient and invasive [66]. Besides, due to the limited field of view (FoV), the egocentric video data are highly heterogeneous and lack compatibility [60, 201, 89]. For instance, the EPIC-KITCHENS [60] captures the users' hands whereas Charades-Ego [201] misses them, causing disparate inference results. Recently, more egocentric vision research uses additional modalities to assist the egocentric vision including audio [116, 49] and motion sensor [270, 163]. However, these approaches inherit the limitations of camera sensing including privacy intrusive, energy hungry, and crippling for continuous sensing. EgoADL proposes to bring the egocentric ambient intelligence to real life by using non-visual sensors which are less intrusive and insensitive to the FoV problem, yet achieving similar performance as the vision-based approaches.

ADL sensing using non-visual sensors: There exist a huge portfolio of non-visual modalities for ADL sensing, including motion sensor [194, 272, 269], audio [82, 122, 231, 265, 23], RF signals [111, 247, 252] and others [217].

Within this area, motion sensor-based ADL sensing has mostly focused on classifying a small set of prescribed activities associated with specific body parts [194, 272]. SSL has been introduced recently to train feature representation models based on motion sensor data [269, 98, 224, 147]. EgoADL wildly expands this strand of research towards cross-modal SSL, which fuses multiple modalities to log more complex ADL, similar to a human transcriber.

Sound event detection, as one modality to understand ADL, has been extensively studied [82, 122, 231, 265, 23]. Existing solutions use an external microphone array, *e.g.*, one equipped on a voice assistant, to capture the sound. Ubicoustics [126] is the only work that has a smartphone-based egocentric sound capturing setup similar to EgoADL, but it only classifies

30 acoustic activities. Unlike traditional sound event classification, EgoADL is more extensible due to its SSL architecture, and evades majority of the labeling burden by SSL and distilling knowledge from existing sound datasets.

Device-free RF sensing has gained major traction, demonstrating abilities to classify a dozen of prescribed activities [111, 247, 252]. However, due to limited antenna aperture and hence spatial resolution, commercial RF signals, like Wi-Fi sensing, cannot capture the nuances of human-object interactions (HOI), unless augmented with dedicated hardware. For example, LiveTag enables HOI by attaching passive touch-sensitive tags on the objects [79]. RF-Diary [74] employs a powerful FMCW radar and a floor map of object locations to detect HOI. Besides the hardware complexity, cost and labeling burdens, the device-free sensing systems bear a few common limitations. First, the coverage area is typically limited to a single-room. Second, without dedicated hardware [79, 74], the sensing performance is highly sensitive to environment and transceiver locations. State-of-the-art device-free WiFi sensing systems can only achieve $< 75\%$ accuracy in recognizing 6 activities, when tested across different environments [110]. In contrast, EgoADL is the first to explore egocentric Wi-Fi sensing by capturing ambient Wi-Fi CSI using a in-pocket smartphone, combined with other sensors to overcome such limitations.

SSL for ADL sensing: One of the major challenges for ADL analysis is lack of labelled data, especially for relatively rare ADL [203]. This challenge exists even for egocentric vision due to the limited FoV and the complexity of ADL. SSL has proven to be a promising solution for vision [112], motion sensor [269, 98, 224, 147, 99], and other modalities [173]. In particular, recent work adopted SSL on unlabeled audio-visual data [28, 27, 51]. By understanding the video-audio correspondence, such methods achieve the state-of-the-art performance on egocentric ADL recognition. EgoADL introduces new modeling mechanisms (*e.g.*, joining single- and multi-modal SSL) to tackle a disparate set of modalities. Furthermore, EgoADL distills knowledge from external datasets to guide the SSL towards a better cross-modal feature representation.

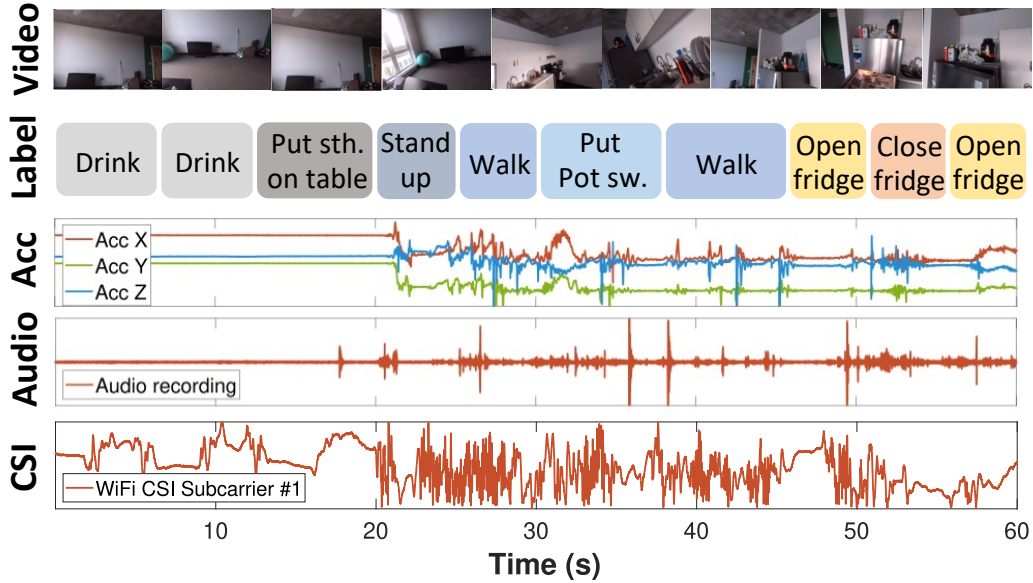


Figure 5.2. EgoADL data in the time domain, including Wi-Fi CSI, audio, accelerometer signals, ground truth labels and egocentric video from a head-mounted GoPro for ground truth labeling.

5.3 EgoADL Setup and Data Collection

EgoADL employs a commodity smartphone as an egocentric sensor hub, which captures the audio, wireless sensing signals (*i.e.*, Wi-Fi), and motion sensor signals continuously. Users can perform *arbitrary* daily routines with the sensor hub, *i.e.*, an in-pocket smartphone, in free-living environments. EgoADL will recognize ADLs including both human activity and human-object interaction from the sensor data without human intervention. We will first discuss EgoADL data collection setup and dataset. Our study was approved by the IRB before all the data collection.

Smartphone as an egocentric sensor hub: The EgoADL data collection app is implemented by JAVA in Android OS. It simultaneously collects 3 sensing modalities from the user’s smartphone. (i) *Audio capturing:* We record the monophonic sound at 48 kHz sampling rate through the smartphone’s bottom microphone. (ii) *Motion sensors:* We capture the 3-axis accelerometer signals at 200 Hz sampling rate, and log the per-sample timestamp for later uniform resampling. (iii) *Wi-Fi CSI:* We use the Nexmon CSI tool [92, 197] to extract the incoming

Wi-Fi packets' CSI. Our EgoADL prototype uses Nexus 5 for its compatibility with Nexmon [92, 197]. During the data collection, we employ a commodity 802.11ac Wi-Fi access point (AP) to transmit data at 400 packets/second. Due to packet losses, the receiving packet rate tends to be lower. Nevertheless, our data collection consistently maintains a minimum reception rate of 200 Wi-Fi packets/second, ensuring data quality.

Data collection procedure and setup: For our data collection, we recruited 30 participants, comprising 8 females and 22 males with an average age of 25.9. The participants had an average age of 25.9 years. Among them, 10 participants (3 females and 7 males) consented to provide ground-truth labels for our study. One notable limitation of our current dataset is the underrepresentation of female and elderly participants. Fig. 5.3 shows the detailed demographics. We clearly communicated the data collection objectives to the participants. Each participant was instructed to record data over a week, aiming for at least 5 hours in total. To guarantee a diverse and sufficient collection of ADL, we advised them to record during routine activities, excluding stationary periods such as working at a desk or sleeping. We did not impose any specific ADL types or scripted activities.

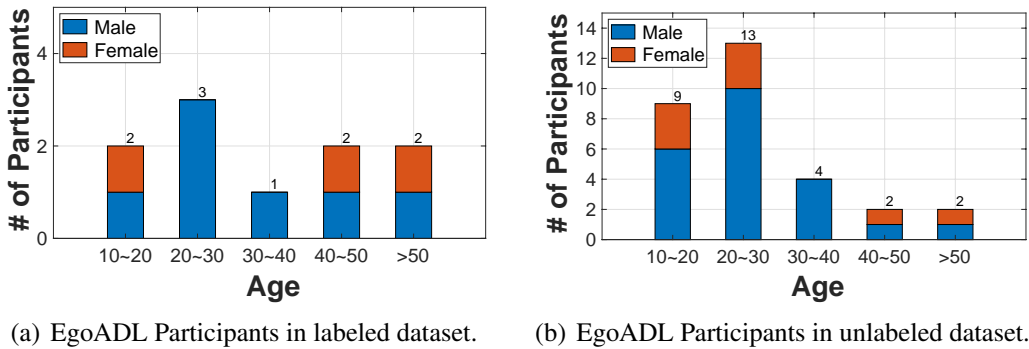


Figure 5.3. EgoADL demographics of participants.

Participants were provided with necessary devices, including a Wi-Fi AP and a smartphone equipped with the EgoADL app, along with instructions for setting up the devices. We asked participants to deploy the Wi-Fi AP at an *arbitrary location* in their home and they are

required to simply put the smartphone into their left/right trouser pocket, freely performing daily routines as the data collection app runs in the background. Participants are also allowed to take the smartphone out of the pocket and use the smartphone freely as usual. Upon reaching a user-specified time limit (typically 10 to 30 minutes), the app plays a notification sound and saves the sensor data. For participants who agreed to provide ground-truth labels (7 males and 3 females with an average age of 28.5, Fig. 5.3(a)), the procedure was identical, with an additional step of wearing a head-mounted GoPro camera. This camera captured egocentric audio and video at a resolution of 2560×1440 , 30 FPS, and with a linear field of view.

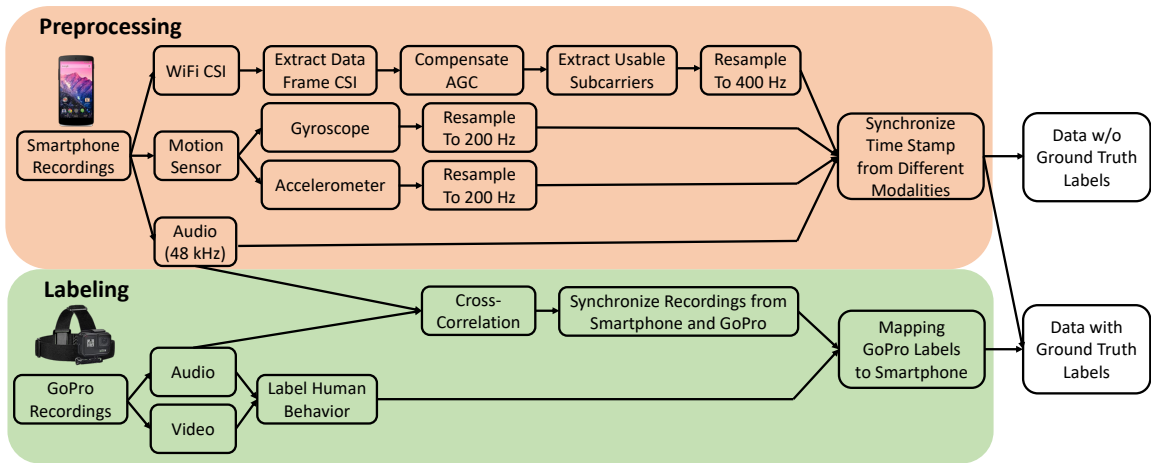


Figure 5.4. Implementation pipeline of the EgoADL preprocessing and labeling.

Data preprocessing: Fig. 5.4 summarizes the preprocessing pipeline. To mitigate the impact of unstable sample timing on Commercial Off-The-Shelf (COTS) smartphones, we first resample the motion sensor data uniformly to 200 Hz. For the Wi-Fi CSI data, we normalize the per-subcarrier magnitude by the corresponding automatic gain control (AGC) values to mitigate the AGC artifacts, and then resample the CSI sequence to 400 Hz. Specifically, we preprocess the Wi-Fi CSI data as follows: 1). Discard the frames without < 80 MHz bandwidth, and only keep the data frames with 80 MHz; 2). Compensate the automatic gain control (AGC) to guarantee the stable amplitude of Wi-Fi CSI signals in the time domain; 3). Discard the subcarriers without Wi-Fi CSI; 4). Resample the Wi-Fi CSI uniformly to 400 Hz sampling rate. Afterwards, we

synchronize the three sensing modalities based on their sampling timestamp. Afterwards, we synchronize the three sensing modalities based on their sampling timestamp.

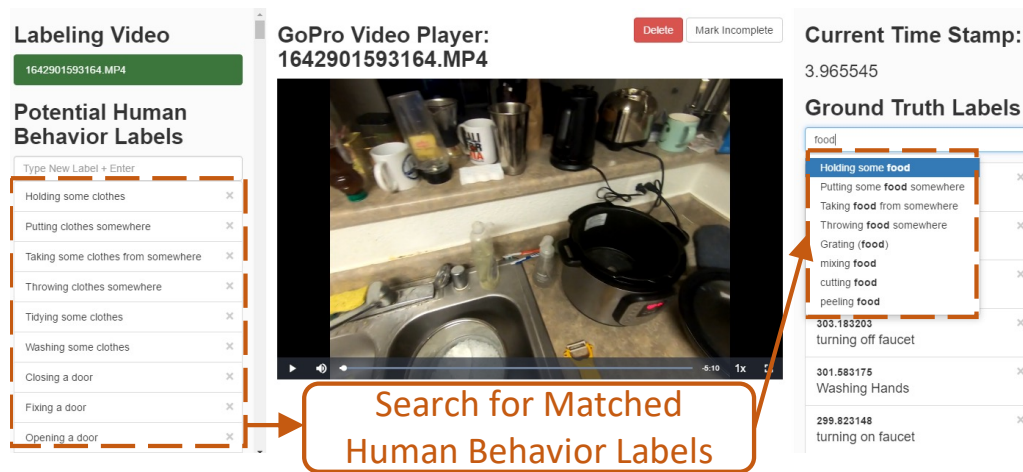


Figure 5.5. EgoADL labeling tool

Data labeling: Each ground truth label needs to specify the ADL, *i.e.*, an ambulatory action or human-object interaction event, along with the start and end timestamps. For instance, Fig. 5.2 shows 1-min labeled data containing a sequence of ADL. As shown in Fig. 5.5, we design a labeling tool to allow playback of the GoPro video/audio recordings and annotating the data using a set of ADL labels created by existing state-of-the-art video/audio based ADL sensing systems [203, 89, 22, 127]. To ensure accurate labeling, the annotators are exactly the data collectors, compensating for any limitations in egocentric video field of view (FoV) by using their memory of the events. They are equipped to segment and label the video footage, either using the provided predefined ADL labels or by adding new ones as they identify them. This open-ended approach to data labeling and annotation has yielded a comprehensive dictionary encompassing 1,000 words specifically related to ADL. The maximum allowable duration for each labeled segment is 10 seconds, aligning with the typical duration of discrete ADL observed in our studies. We further conduct the user studies with annotators and they empirically separate the ADL into *state-based* and *event-based* ADL [60]. *State-based ADL* usually last more than 5 s each and often periodically and continuously, like “walking” and “chopping meat”, *etc.* *Event-based ADL*

are one-short, like “opening the door” and “sitting down in chair”, *etc.*

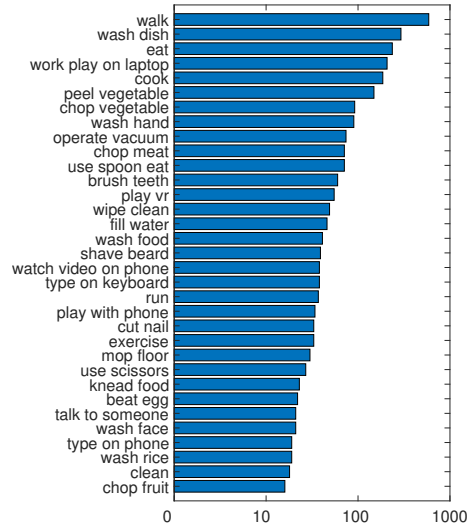


Figure 5.6. State-based human behaviors in EgoADL dataset.

Dataset Scale: Following the data labeling process, we streamlined the dataset by reducing the representation of longer-duration event-based ADL, particularly those that span over 10 consecutive minutes. This process yielded an effectively condensed dataset with an average duration of approximately 2 hours per participant. For the data without labels, we selectively refined the data by eliminating segments that lacked significant variation across all three modalities. Finally, we organized the data into three datasets. *(i) Labeled dataset.* The labeled dataset contains 20 hours of records from 10 users across 7 homes, area ranging from 600 to 2000 ft² with a variety of layouts. It comprises 7,000 ADL samples, including 221 types of ADL involving 70 actions and 91 objects. A detailed list is in Fig. 5.6 and Fig. 5.7, respectively. This dataset serves as a baseline for preliminary experimentation and few-shot fine tuning. *(ii) Balanced dataset with labels.* Remarkably, the uncontrolled user activities manifest an imbalanced long-tail distribution— more than half of the ADL in EgoADL are infrequent and only have less than 15 samples. To establish a baseline supervised learning model, we select a subset from (i), which involves 105 ADL each with 15 to 25 samples (2,500 samples in total),

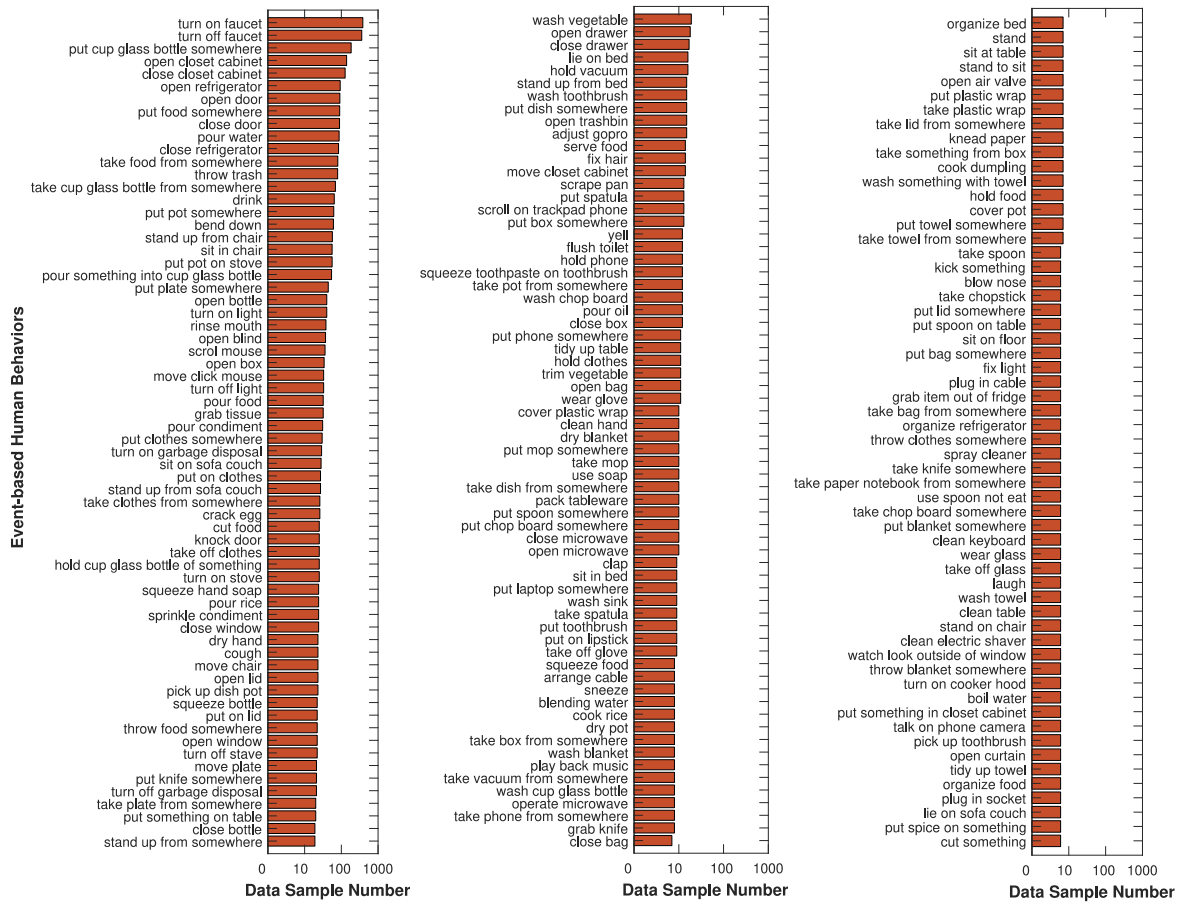


Figure 5.7. Event-based human behaviors in EgoADL dataset.

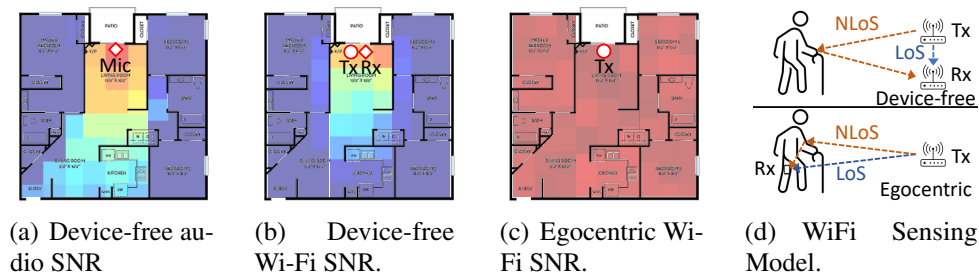


Figure 5.8. Egocentric v.s. Device-free Sensing SNR for audio and Wi-Fi. “User squeezing plastic bottle” sound and “Sit down in chair” motion are used as benchmarks sound event and human activity for audio SNR and Wi-Fi CSI SNR measurements. The heatmap in (a) represents the SNR as we vary sound source locations while fixing the mic. The heatmap in (b) and (c) represents the Wi-Fi CSI sensing SNR of specific location.

referred to as a *balanced dataset*. (iii) *Unlabeled data*. To facilitate the SSL (Sec. 5.6), we collected more than 100 hours of unlabeled egocentric data from 30 users in 20 homes.

5.4 Preliminary Study

In this section, we will discuss the advantages of EgoADL design choices, *i.e.*, egocentric sensing and sensing modalities selection.

5.4.1 Advantages of Egocentric Sensing

To better understand the advantages of the egocentric sensing, we conduct controlled experiments in a 1400 ft² apartment, and compare EgoADL against the conventional device-free setup [264] which captures users’ ADL through off-body non-visual sensors, *e.g.*, voice assistant [126, 23, 265] and Wi-Fi AP [247, 110]. We only examine the audio and Wi-Fi CSI modalities here since motion sensors are already widely used in egocentric setup [272, 269], which characterize a specific body part motion without any interference from other users.

Sensing Space Coverage: To control the audio sensing setup, we use a loudspeaker to replay a benchmark sound of “user squeezing plastic bottle” at a constant 68 dBA SPL, to emulate the corresponding user activity. For Wi-Fi CSI sensing, we use “sit down in chair” as the benchmark activity. In the device-free scenario, we vary the sound source location and the

location of human activity while fixing the sensor hub at a specific location (\diamond in Fig. 5.8(a) and Fig. 5.8(b)). In the egocentric scenario, the users put the sensor hub, *i.e.*, smartphone, in their trouser pocket. As shown in Fig. 5.8(a), in the device-free setup, the microphone is sensitive to wall blockage, and can only sense the activity sound at single-room coverage. Meanwhile, Fig. 5.8(b) shows that, the signal strength of device-free Wi-Fi sensing drops dramatically as the user moves away from the Tx/Rx or behind the wall. This is because it relies on the NLoS signals bouncing off the target user's body, which suffers from severe attenuation, diffraction, and scattering effects. *In contrast, in the egocentric setup, the sensor hub accompanies the user, so it achieves whole-home coverage with consistently high SNR (> 25 dB) for both audio and Wi-Fi CSI.*

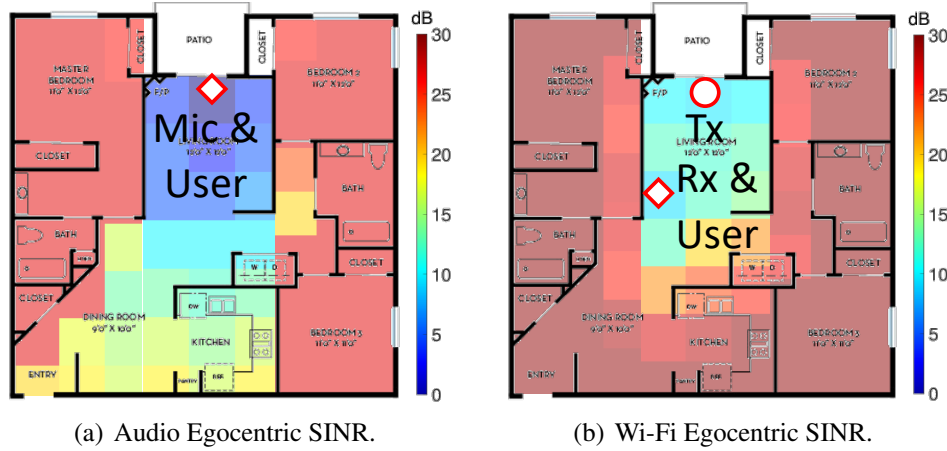


Figure 5.9. Egocentric SINR for the same benchmark sound event and human activity as in Fig. 5.8. The targeted user with egocentric sensor hub is fixed at the location of \diamond . The heatmap in (a) and (b) represents the SINR of audio and Wi-Fi CSI of the targeted user when there is an interfering subject performing the benchmark activity at each location.

Resilience to interference: Since the device-free setup achieves low whole-home SNR even without the interference source, we only examine the resilience to interference under the egocentric setup. Here the targeted user stays at a fixed location, while another (interfering) user performs the benchmark activity at arbitrary locations. We measure the SINR, where the desired signal power equals to the variance of egocentric signals caused by the targeted user's

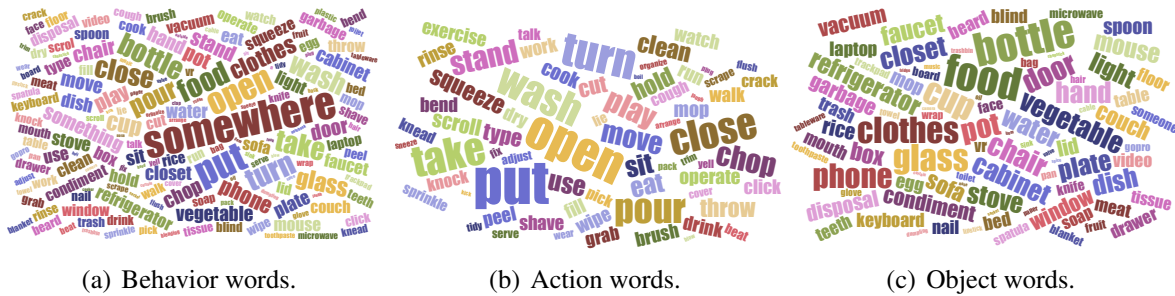


Figure 5.10. EgoADL dataset labels word cloud.

activities, whereas the interference is that from the interfering user. As shown in Fig. 5.9, for both audio and Wi-Fi CSI sensing, *the presence of an interfering user will noticeably impact the egocentric SINR (from 25 dB to 10 dB) only when it is in close proximity (< 2 m) of the target user or blocking the LoS path between the Tx and Rx for Wi-Fi CSI sensing.*

To sum up, compared to conventional device-free setup, egocentric sensing significantly improves the operating range and anti-interference ability for both Wi-Fi CSI and audio sensor. Although close-by interferers may still impact the SINR, we anticipate the motion sensor along with deep modality fusion can neutralize such impacts. Thus, in EgoADL, we do not restrict the presence of interference—all the data are collected in daily living settings with multiple coresidents.

We summarize the insights as follows:

i). Sensing Space Coverage: The signal strength of device-free sensing suffers from severe attenuation, diffraction, and scattering effects and drops dramatically as the user moves away from the Tx/Rx or behind the wall. In comparison, in the egocentric setup, the sensor hub accompanies the user, so it achieves whole-home coverage with consistent signal quality for both audio, Wi-Fi CSI and motion sensor.

ii). Resilience to interference: Wi-Fi CSI and audio suffer from interference since they can not distinguish the motion from targeted subject. In contrast, the egocentric sensing setup makes the presence of an interfering user impact significant only when it is in close proximity of

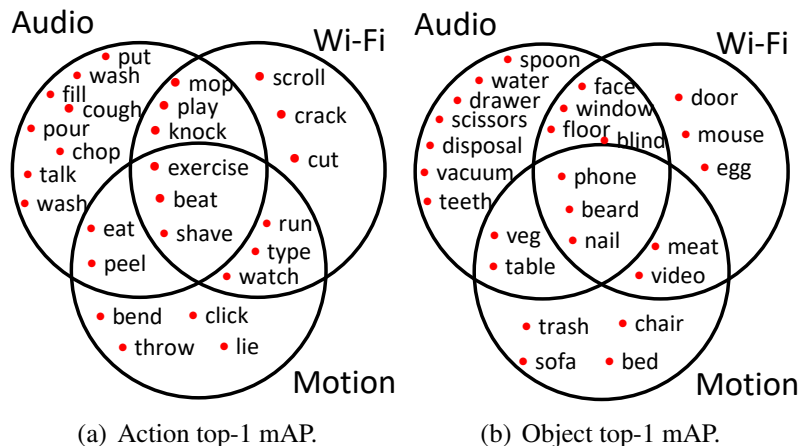


Figure 5.11. Venn diagram visualizes the advantages of each modality. For each action/object class, if a modality achieves comparable top-1 mAP (within 15%) with the best modality, we plot this class in the intersection of two circles representing these two modalities. Since the results are trained on single-modality data, there may be some overfitting in this visualization (e.g., Wi-Fi is good at recognizing “mouse” and “egg”).

the target user.

5.4.2 Sensing Modality Selection

There are mainly two reasons why EgoADL combines 3 modalities, *i.e.*, the motion sensor signals, wireless sensing signals and audio, to sense users’ ADLs.

i). Availability on existing commodity devices: These 3 non-visual sensors are widely equipped on mobile/wearable devices, including smartphones and smartwatches, which can be easily repurposed as an egocentric sensor hub to log ADL. In particular, the Wi-Fi CSI can be collected on such devices [197] but is heavily underutilized as a potential sensing modality.

ii). Complementary advantages of each modality: To understand the *complementary advantages* of different modalities, we conduct experiments to recognize the ADL by using each single modality data and the DNN model proposed in Sec 5.5. We use the balanced dataset with labels (Sec. 5.3), with a 7 : 1.5 : 1.5 split among training, validation, and testing set. Fig. 5.11 visualizes the *action* and *object* categories with > 60% mAP for at least one single modality. We summarize our insights as follows:

- In-pocket motion sensor easily captures ambulatory actions of the leg, *e.g.*, “sitting in the chair”, “lying on the bed”, *etc.*, but cannot easily capture whole-body motion or object interaction.

- Wireless sensing signals, *i.e.*, Wi-Fi CSI, can recognize certain full-body motion and interactions with ambient environment, like “opening the door”, “opening the window”, *etc.*, but it falls short in discriminating fine-grained activities.

- Audio sensing can easily recognize ADL with unique sound events, like “coughing”, “operating vacuum”, “brushing teeth”, *etc.*, but can hardly identify those with weak or similar sounds.

EgoADL aims to approach near-vision sensing resolution by synergizing the complementary advantages of the non-visual modalities.

5.5 EgoADL Supervised Learning

5.5.1 Problem Formulation

Most of the traditional DNN-based ADL analysis models are designed to perform *classification* [60, 202]. The key limitation is that they have to prescribe a known set of ADL, which falls short of extensibility when new ADL of interest emerge. Besides, such models only classify the ADL as integer IDs which do not fully utilize the label semantics from the natural language level [285].

In contrast, we propose a solution that formulates the problem as a sequence-to-sequence (seq-2-seq) task. Our approach encodes multi-modal sensory features and decodes them as the label name semantics using a sequence of words, rather than simple classification labels. This enables us to capture more nuanced and precise information about ADL. Specifically, our goal is to translate synchronized signals $\mathbf{x} = \{\mathbf{x}^{(a)}, \mathbf{x}^{(c)}, \mathbf{x}^{(m)}\}$, where $\mathbf{x}^{(a)}$, $\mathbf{x}^{(c)}$, and $\mathbf{x}^{(m)}$ correspond to audio recordings, Wi-Fi CSI, and motion sensor signals, respectively, into semantic labels \mathbf{y} for ADL in the form of a sequence of words.

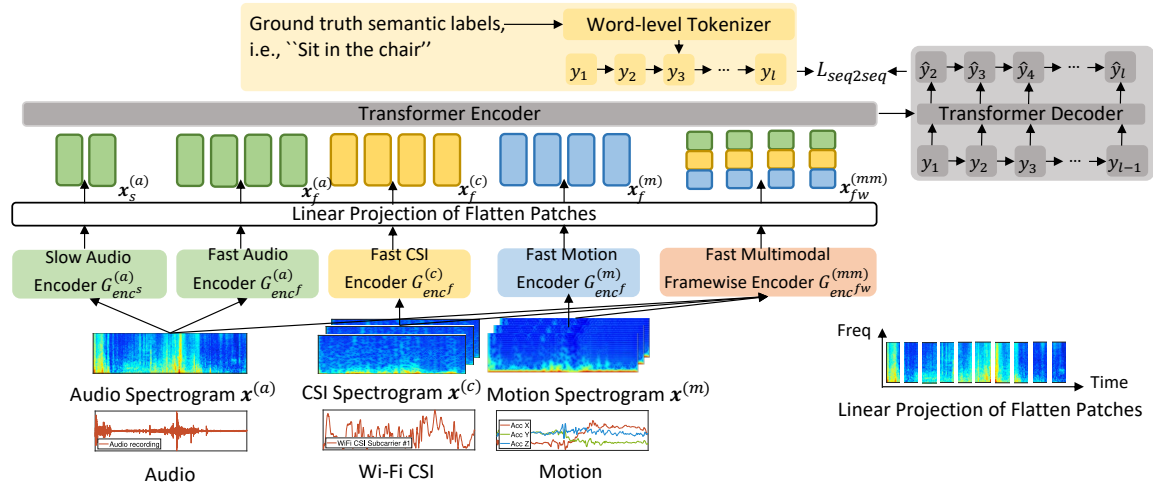


Figure 5.12. EgoADL Multi-Modal Frame-Wise Slow-Fast (MMFWSF) transformer design with modality-specific encoders and a transformer-based seq-2-seq model for translating sensory feature into natural language.

5.5.2 Input and Output Design

As shown in Fig. 5.12, we first discuss the input feature design.

Audio Recordings: We extract the T-F log mel spectrograms with a sampling rate of 32 kHz, Hamming window size of 31.25 ms, hop size of 10 ms and 64 mel filter banks [122].

Wi-Fi CSI: We first normalize each CSI subcarrier by mean-std normalization, and then extract the Doppler spectrogram applying STFT on the time-domain sequence of CSI values across subcarriers. The STFT uses a Hamming window size of 200 ms, a hop size of 10 ms and FFT size of 128 at 400 Hz sampling rate. The window size is much larger than audio because the Doppler feature caused by human motion exhibits lower frequency.

Motion Sensor: We utilize the linear acceleration (excluding the effect of gravitational force) as the raw input data [232], apply mean-std Z-normalization [224, 99], and then extract the motion sensor spectrogram via STFT on the time domain output of each sensor channel [232, 272, 273]. Such a preprocessing pipeline, as recommended by existing studies in human activity recognition using motion sensors [232, 224, 99, 272, 273], not only enriches the time-frequency domain representation but also helps to reduce the impact of variations in device orientation. Note that we use the *same hop size* of 10 ms for all 3 modalities to ensure *feature*

alignment in the time domain. For each modality, the input of the DNN model is a sequence of frequency spectrogram, $\mathbf{x} = \{x_1, x_2, \dots, x_t\}$, where t is the length of the spectrogram in the time domain. For each timestamp i , $\mathbf{x}_i = \{x_i^{(a)}, x_i^{(c)}, x_i^{(m)}\}$ represent the audio log-mel spectrogram, Wi-Fi CSI Doppler spectrogram, and motion sensor spectrogram, respectively.

EgoADL outputs a natural language text description of the ADL. As shown in Fig. 5.12, the output can be represented as a sequence of tokens. Unlike conventional ASR which uses the subword-level or character-level tokenizer corresponding to the phonological units, we adopt a *word-level tokenizer*, corresponding to the basic unit in ADL semantics. For example, for the “chop vegetables” activity, we expect EgoADL to recognize the action “chop” and the object “vegetables”, and generate a sequence of tokens “chop vegetables”. Our tokenizer dictionary contains 1,000 frequently-used words for describing ADL from EgoADL dataset labeling (see the data labeling discussion in Sec. 5.3). Finally, the output sequence is $\mathbf{Y} = \{y_1, y_2, \dots, y_l\}$, where l denotes the number of words in the output text.

5.5.3 MMFWSF Transformer

Design Principles: Next, we introduce our Multi-Modal Frame-Wise Slow-Fast (MMFWSF) transformer. Compared to existing methods that fuse traditional modalities such as audio, video, and text [78], the multi-modal fusion in EgoADL possesses the following unique properties which we can leverage:

- Audio, Wi-Fi CSI, and motion sensor data all have complementary advantages when it comes to capturing *fast-changing motion* [76], including both one-shot event-based ADL such as “sit down” and “stand up”, and periodical state-based ADL such as “walking” and “clapping”.
- In contrast to Wi-Fi CSI and motion sensor data, which mainly capture *fast-changing motion*, audio data has a distinct advantage in capturing *continuous scene sounds* with specific frequencies, such as “operating a vacuum,” “shaving bread,” and “brushing teeth”.
- Another essential aspect of our approach is the use of *framewise alignment* between multiple modalities. Since no single modality can fully capture all aspects of ADL, there may

always exist modality missing for different behaviors. For instance, in Fig. 5.2, the "Drinking" behavior is only captured by Wi-Fi CSI. However, the missing modality can provide additional supervision, helping us determine whether a specific modality can capture a specific behavior and what the cross-modal cues are at the frame level.

MMFWSF Transformer design: We design our multi-modal fusion and transformer-based seq-to-seq model to fully utilize the aforementioned insights.

Table 5.2. EgoADL CNN-based encoders parameters (C: Channel).

	Slow-pathway	Fast-pathway			Fast-pathway (Frame-wise)		
Modality	Audio	Audio	Acc	CSI	Audio	Acc	CSI
Spectrogram	(t, 64) C:1	(t, 64) C:1	(t, 64) C:3	(t, 64) C:208	(t, 64) C:1	(t, 64) C:3	(t, 64) C:208
Stride	(8,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)
CNN	3*3, C:120	3*3, C:30	3*3, C:30	3*3, C:30	3*3, C:10	3*3, C:10	3*3, C:10
Activation	BatchNorm2d+LeakyReLU + Dropout(0.1)						
Stride	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)
CNN	3*3, C:240	3*3, C:60	3*3, C:60	3*3, C:60	3*3, C:20	3*3, C:20	3*3, C:20
Activation	BatchNorm2d+LeakyReLU + Dropout(0.1)						
Stride	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
CNN	3*3, C:480	3*3, C:120	3*3, C:120	3*3, C:120	3*3, C:40	3*3, C:40	3*3, C:40
Concat	/	/	/	/	Frame-wise Concat		
Concat	Modal-wise Concat						

First, to leverage the complementary advantages of each modality for capturing *fast-changing motion*, we design *fast pathway* CNN-based encoder $G_{enc^f}^{(a)}$, $G_{enc^f}^{(c)}$ and $G_{enc^f}^{(m)}$ for audio, Wi-Fi CSI and motion sensor, respectively, to achieve a fine feature representation along the temporal dimension. The basic idea is to design CNN encoders with a small temporal stride of τ , resulting in the length of frequency spectrogram t/τ , to guarantee high temporal resolution. The default value is $\tau = 4$ in our experiments. Tab. 5.2 shows the design of different CNN-based encoder parameters. To make sure all the representations after encoders can fit into the transformer, we intentionally enforce the feature representation of a single frame to be the same

size. Therefore, the ‘‘Slow-pathway’’ encoders will have more channels than the ‘‘Fast-pathway’’ encoders while the ‘‘Modalwise’’ encoders will have more channels than ‘‘Framewise’’ encoders. Our transformer network architecture is based on [234]. It comprises 12 encoder layers and 6 decoder layers. Positional encoding is employed to capture temporal dynamics in sensory time-series data. Within the multi-head attention mechanism, we have configured 8 heads, and the dimensionality of the feedforward network is set at 3072 as default. Finally, with the *fast pathway* CNN-based encoder, the feature representations of audio ($j = a$), Wi-Fi CSI ($j = c$) and motion sensor ($j = m$) are

$$\mathbf{x}_f^{(j)} = \{x_{1f}^{(j)}, x_{2f}^{(j)}, \dots, x_{t/\tau_f}^{(j)}\} = G_{enc^f}^{(j)}(\mathbf{x}^{(j)}) \quad (5.1)$$

where $x_{if}^{(j)} \in \mathbb{R}^{C \times F}$ ($i = 1 \sim t/\tau$), C is the number of channels, and F is the number of frequency bins after *fast pathway* CNN-based encoder. The input of the transformer is $\mathbf{x}_f^{(mm)} = (\mathbf{x}_f^{(a)}, \mathbf{x}_f^{(c)}, \mathbf{x}_f^{(s)})$.

Second, we design an additional audio *slow pathway* with a CNN-based encoder $G_{enc^s}^{(a)}$ to learn the feature representation for *continuous scene sounds*. Basically, we use a large temporal stride of $\alpha\tau$, where $\alpha > 1$ to focus on learning frequency semantics [117]. We set $\alpha = 16$ as default. And the feature representation after the audio *slow pathway* CNN-based encoder is

$$\mathbf{x}_s^{(a)} = \{x_{1s}^{(a)}, x_{2s}^{(a)}, \dots, x_{\frac{t}{\alpha\tau}}^{(a)}\} = G_{enc^s}^{(a)}(\mathbf{x}^{(a)}) \quad (5.2)$$

where $x_{is}^{(a)} \in \mathbb{R}^{C \times F}$ ($i = 1 \sim \frac{t}{\alpha\tau}$).

Third, to learn the feature representation of *framewise alignment* between multiple modalities, we propose to further fuse the multi-modal sensory data at the frame level. Note that to make sure the final input sequence of feature representation can be the input of the transformer-based seq-2-seq model, we need to make sure that each feature representation is of the same size. Therefore, we train another 3 *fast pathway* CNN-based encoders $G_{enc^{fw}}^{(a)}$, $G_{enc^{fw}}^{(c)}$ and $G_{enc^{fw}}^{(m)}$ for each modality with $C/3$ channels, so that the feature representation of each

modality $x_{i_{fw}}^{(j)} \in \mathbb{R}^{\frac{C}{3} \times F}$, where $i = 1 \sim t/\tau$. Finally, the feature representation of *frame-wise* fusion is

$$x_{i_{fw}}^{(mm)} = \text{concat}(G_{enc_{fw}}^{(a)}(\mathbf{x}^{(a)}), G_{enc_{fw}}^{(c)}(\mathbf{x}^{(c)}), G_{enc_{fw}}^{(m)}(\mathbf{x}^{(m)}), \text{dim} = i) \quad (5.3)$$

where $x_{i_{fw}}^{(mm)} \in \mathbb{R}^{C \times F}$, and $\mathbf{x}_{fw}^{(mm)} = \{x_{1_{fw}}^{(mm)}, x_{2_{fw}}^{(mm)}, \dots, x_{t/\tau_{fw}}^{(mm)}\}$.

Note that to make sure all these representations can fit into the transformer, we intentionally enforce the feature representation of a single frame as $\mathbb{R}^{C \times F}$. Thus, inspired by vision transformer [67] and audio spectrogram transformer [84], our MMFWSF transformer model can concatenate the sequence of $\mathbf{x}_s^{(a)}, \mathbf{x}_f^{(mm)}, \mathbf{x}_{fw}^{(mm)}$ along the temporal dimension and then use the linear projection of flatten patches along the channel and frequency dimension to fit into the transformer model as shown in Fig. 5.12. Our transformer model contains 12 encoder layers and 6 decoder layers.

5.5.4 Training Strategy

Training loss design: We use the seq-to-seq loss based on the autoregressive decoder, where the previous output is fed back into the input, to decode the input sequence to the output token sequence. In the testing phase, the predicted word label \hat{y}_i and the hidden state h_i of the decoder at step i can be updated as $\hat{y}_i, h_i = \text{Decoder}(h_{i-1}, \hat{y}_{i-1}, c_i)$, where c_i is the context vector generate by the encoder. In the training phase, we use teacher forcing methods to train the model, which means $\hat{y}_i, h_i = \text{Decoder}(h_{i-1}, y_{i-1}, c_i)$, where y_{i-1} is the last token of the ground truth label. The objective is to minimize the corresponding cross entropy loss l_{seq2seq} .

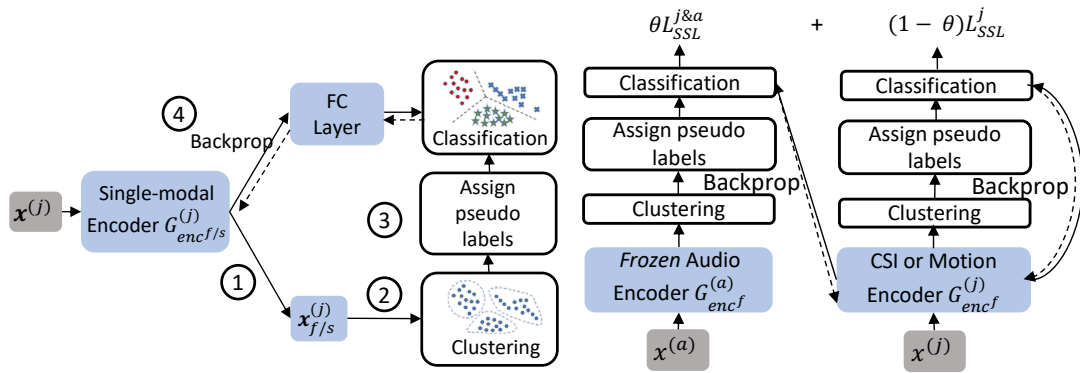
Beam search during testing: In the testing phase, we use beam search, a widely adopted method in NLP and ASR [236], to search for the top- K candidate sequences. For each step, we predict the K most promising next tokens, and then feed these K alternatives into the decoder to select the best K hypothesis at the next step iteratively.

5.6 EgoADL Self-supervised Learning

We have conducted experiments on the supervised model (Sec. 5.5) and identified limitations and potential pathways towards a self-supervised EgoADL model design. We found that while the supervised model performed well on a balanced labeled dataset, it struggled with *overfitting* and *lack of generalization and extensibility*, especially for infrequent ADL, due to limited ground truth labels. Scaling up the dataset requires an enormous amount of labeling effort, especially for non-visual sensors. On the other hand, in contrast to vision-based setups that require users to wear multiple devices (*e.g.*, head-mounted or chest-mounted cameras [60, 89]), collecting large-scale unlabeled data is much easier with EgoADL, as it is non-intrusive and only requires users to carry a smartphone in pocket. We harness this unique advantage through an SSL model that can improve (i) accuracy, (ii) extensibility for few-shot ADL with limited labels, and (iii) generalization across different ADL, users, and environments. *Our SSL model trains more generic encoders (see Fig. 5.12) by leveraging intrinsic supervisory signals within unlabeled data, and by distilling knowledge of human behavioral logic from external audio datasets.*

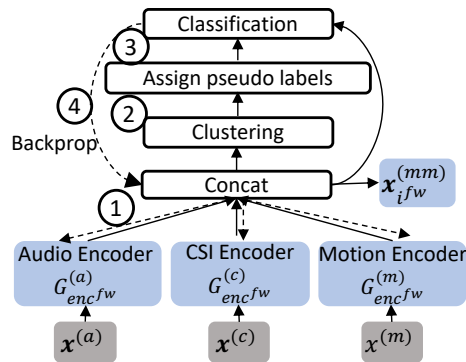
5.6.1 Single-modal Self-Supervised Deep Clustering

We first introduce a self-supervised clustering method to learn the single-modal encoders from unlabeled data. Inspired by vision-based SSL [48], our deep clustering method takes the input feature $\mathbf{x}^{(j)}$ from a single modality (j) as input. For each epoch, the training procedure of the deep clustering method follows the steps ①–④ in Fig. 5.13(a). First, the single-modal encoder generates the feature representation $G_{enc^{(j)}}(\mathbf{x}^{(j)})$. Second, an unsupervised clustering method is applied to all the feature representations from EgoADL’s unlabeled training data. Then, we assign *pseudo labels*, *i.e.*, the cluster index of each resulting cluster, to the corresponding data. Finally, we append fully-connected layers to the encoders to classify the feature representations to their corresponding pseudo labels and use the back propagation training algorithm to update the parameters in the encoders. The trained single-modal feature representations will be used in



(a) Single-modal SSL. EgoADL uses the deep cluster- (b) Single-modal deep clustering via cross-modal ing to assign pseudo labels for unlabeled training data self-supervision. EgoADL utilizes the pseudo la- to train the encoders $G_{enc^f/s}^{(j)}$, where “j” here represents bels generated by pretrained audio encoder $G_{enc^a}^{(a)}$, to audio, motion sensor or Wi-Fi CSI.

to train the encoders $G_{enc^f}^{(j)}$, where “j” represents motion sensor or Wi-Fi CSI.



(c) Cross-modal deep clustering. EgoADL fuses the modalities and learns their corre- spondence by training the concatenated rep- resentations of the 3 modalities encoders ($G_{enc^a}^{(a)}$, $G_{enc^c}^{(c)}$ and $G_{enc^m}^{(m)}$)

Figure 5.13. EgoADL SSL methods.

our later cross-modal SSL stage.

In our implementation, we use the entire 100-hour unlabeled EgoADL dataset to train the encoders for each modality separately, *i.e.*, $G_{enc_f}^{(a)}$, $G_{enc_s}^{(a)}$, $G_{enc_f}^{(c)}$ and $G_{enc_f}^{(m)}$. We choose K-means and 3 fully-connected layers with ReLU activation functions as the default clustering and classification method, respectively. To optimize the number of clusters k for K-means, we conducted the end-to-end experiments by varying k on a logarithmic scale during the hyperparameter tuning phase. These experiments, conducted within the context of the EgoADL dataset, which encompasses 221 distinct ADL, indicate that a k value within the range of 10^2 to 10^3 yielded near-optimal performance. Consequently, we selected $k = 200$ as our default configuration.

5.6.2 Cross-Modal Self-Supervised Deep Clustering

Second, we also leverage cross-modal self-supervisory to enhance the single-modal deep clustering. More specifically, we leverage the external audio dataset, *i.e.*, AudioSet [82], a massive dataset with more than 5,000 hours of labeled audio recordings for 527 event classes from YouTube videos, to first train generic audio encoders, *i.e.*, $G_{enc_f}^{(a)}$ and $G_{enc_s}^{(a)}$ [122]. Then, we use the audio pseudo labels from the same frame to train the Wi-Fi CSI $G_{enc_f}^{(c)}$ and motion sensor encoders $G_{enc_f}^{(m)}$. The use of audio pseudo labels offers several benefits. First, such labels from the pre-trained audio feature embeddings help prevent the Wi-Fi CSI and motion sensor models from overfitting to particular user characteristics, such as Wi-Fi transmission locations and sensor orientations. Second, they harness the broad sensing capabilities of audio, capturing both the event-based sound and continuous scene sounds for comprehensive supervision (see Sec. 5.5.3). Lastly, these labels can accelerate the training of deep clustering models for Wi-Fi CSI and motion sensor data. Fig. 5.13(b) shows the training procedure of this mechanism. $L_{SSL}^{j\&a}$ represents the loss when we use the audio pseudo labels to classify the Wi-Fi CSI or motion sensor feature representations. The final loss $L_{SSL}^{j\&a} = \theta L_{SSL}^{j\&a} + (1 - \theta)L_{SSL}^j$. Our evaluation results show that without using the pre-trained audio feature embeddings will result in reduction in both accuracy and generalization performance (see Sec. 5.8.2).

After training the single-modal encoders for each of the 3 modalities, we employ cross-modal deep clustering to fuse the modalities and learn their *correspondence*. As shown in Fig. 5.13(c), we concatenate the representations of the 3 modalities encoders ($G_{encfw}^{(a)}$, $G_{encfw}^{(c)}$ and $G_{encfw}^{(m)}$) from the same frame to perform the training of the cross-modal deep clustering. The training procedure of the cross-modal deep clustering follows the same steps as the single-modal case as discussed in Sec. 5.6.1. After finishing cross-modal deep clustering training, the resulting fast multimodal framewise encoder generates the multi-modal feature representation $\mathbf{x}_{fw}^{(mm)}$. In contrast to training single-modal deep clustering and directly concatenating the feature representations of 3 modalities, cross-modal deep clustering tries to automatically learn the co-occurring features from different modalities, which helps the DNN model understand the correspondence from different modalities.

Finally, after the SSL training, we use the cross-modal SSL models to replace the supervised encoders (see Fig. 5.12). We then use the small labeled EgoADL dataset to train the entire model end to end.

5.7 Knowledge Distillation from Natural Language Labels

Given that the output of EgoADL materializes in the form of natural language text, it opens up opportunities for leveraging the inherent semantics of natural language labels to enhance performance even further. In this section, we embark on two approaches. First, we introduce a label refinement mechanism aimed at ensuring that the granularity of labels remains congruent with the capabilities of the sensors. Subsequently, we put forth the idea of utilizing pre-existing natural language text to cultivate contextual reasoning for sequences of ADLs.

5.7.1 Label Refinement

To understand the limits of non-visual sensors, we compare EgoADL with egocentric vision-based methods (see Sec. 5.8.4). While most ADL show reasonable accuracy, we observed that several ADL had significantly lower accuracy. This is because we annotated and labeled the

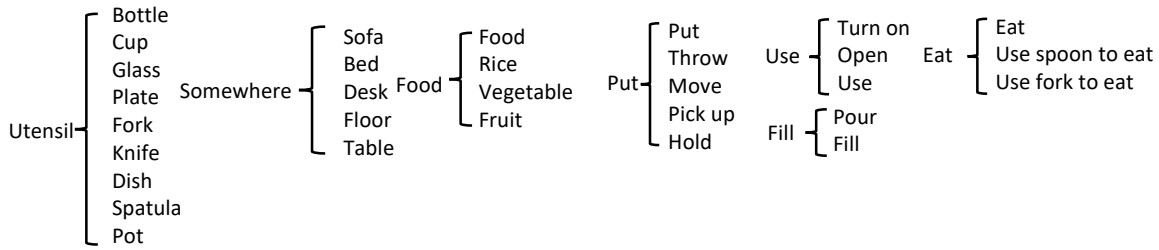


Figure 5.14. Representative label refinement.

EgoADL dataset by manually observing the egocentric video and audio. Nevertheless, as we discussed in the previous sections, the non-visual sensors (audio, motion sensor, and wireless sensing) have limited resolution compared to visual sensors, which may have impacted the labeling accuracy. To better understand the limits of EgoADL, we propose to refine the labeling by merging ADL that are difficult to distinguish using EgoADL non-visual sensors. Our label refinement involves three steps: *i*) ranking ADL based on mean average precision (mAP), *ii*) visualizing the confusion matrix of ADL with less than ϵ , and *iii*) merging actions or objects in ADL based on both the confusion matrix and our knowledge. We have empirically set ϵ to 30%, predicated on the observation that EgoADL’s overall system mAP is approximately 60%, and its discriminative power is significantly reduced for ADL with an mAP below 30%—a scenario applicable to roughly 30/105 of the ADL. We summarize the representative label refinements in Fig. 5.14. The refined labels consist of 75 frequently-used ADL, 38 object interactions, 41 actions, 29 state-based ADL, and 46 event-based ADL (see Fig. 5.17). When employing these labels to assess the performance of EgoADL, the achieved results notably surpass those attained from labels derived from egocentric video and audio sources. This disparity is attributed to the fact that the former set of labels encapsulates the intrinsic capacities of the sensors. Note that our current method requires to refine the labels manually. To further understand the boundary of EgoADL, we need to design an automatic label refinement solution for each modality and multimodal fusion, as well as fine-tuning the hyperparameter ϵ . This will be left as our future work.

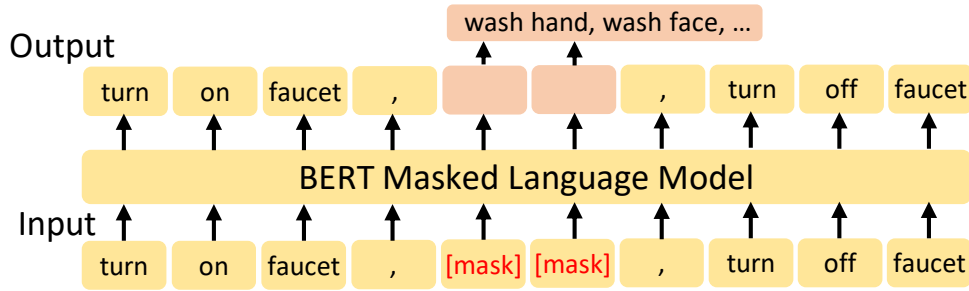


Figure 5.15. BERT-based EgoLM design, which learn the contextual information.

5.7.2 Distilling Contextual Information from Text

The above models only process the short segments of sensor data for single human behavior (with each segment varying in length but not exceeding 10 seconds, as detailed in Sec. 5.3). Longer segments of input may provide contextual information to boost the performance. But it will dramatically increase model complexity, and lead to severe overfitting due to the limited multi-modal dataset. In EgoADL, we harness existing natural language text to learn the contextual reasoning instead.

Inspired by the language model in NLP [64], our idea is to learn a “contextual language model” for ADL, referred to as *EgoLM*. Unlike traditional NLP which calculates the probability distribution over sequences of words, EgoLM outputs the probability of a given sequence of ADL via natural language text. As shown in Fig. 5.15, EgoLM is fine tuned from the celebrated Bidirectional Encoder Representations from Transformers (BERT), a transformer-based NLP model pre-trained by Google [64]. In the training phase, EgoLM takes the text description of a sequence of 30 s ADL as input, and uses a comma to mark the end of the last ADL. During the training phase of EgoLM, we experimented with various masking strategies, including masking a word-level token or an entire ADL, and then we utilize the contextual information from surrounding ADLs to predict the masked elements. Unlike the traditional BERT approach of word-level masking, we choose to mask an entire ADL (as shown in Fig. 5.15), the strategy that resulted in the most optimal performance (see Sec. 5.8.5).

To make EgoLM more general, we collect the training text corpus from not only the

EgoADL dataset but also existing video-based domestic ADL datasets, including Charades [203], CharadesEgo [201], EPIC-KITCHENS [60] and EGTEA-GAZE [137]. We extract the text corresponding to the sequence of ADL within a 30 s period from these datasets, which typically contains 3 ~ 8 ADL. Note that most of the datasets segment the ADL in 10 s units. However, the state-based ADL (see Sec. 5.3) typically last for more than 10 s, and the same ADL labels may appear consecutively. We thus merge these ADL to prevent replication of state-based ADL in the sequence. Overall, our EgoLM text corpus has 3,000 and 34,000 sequences of ADL from EgoADL dataset and 4 external datasets. We first use the whole EgoLM text corpus to train the original BERT model [64], and then fine tune it with the EgoADL dataset.

EgoLM can be applied to any EgoADL models, that we discussed previously, in the testing phase. We first use EgoADL model to generate the potential prediction of each single ADL via beam search, and save the score of the loss function from the EgoADL DNN model. And then, we apply a second round of beam search to combine the EgoADL loss with the language model loss $l_{\text{all}} = \gamma l_{\text{EgoADL}} + (1 - \gamma) l_{\text{EgoLM}}$, where l_{EgoADL} learns the information from the raw data in the current segment, and l_{EgoLM} learns the contextual information in a long period (30 s) before current segment. Finally, we select the ADL with the lowest score of l_{all} to output the top- K prediction.

5.8 Implementation and Experimental Evaluation

5.8.1 EgoADL Implementation and Evaluation Metrics

DNN Implementation: The EgoADL DNN model is implemented in PyTorch. For training, the self-supervised feature embedding DNN models for 3 modalities are first trained separately using single-modal SSL and then jointly using cross-modal SSL, as discussed in Sec 5.6. Next, we freeze these models and train the end-to-end seq2seq model as discussed in Sec. 5.12. We use the Adam optimizer with a $1e - 4$ initial learning rate followed by annealing. The current EgoADL implementation has 421.6 M parameters in total.

Evaluation Metrics: We evaluate EgoADL using classwise mAP metrics and captioning metrics which are adopted in egocentric video-based ADL recognition and captioning [61]. We measure the mAP for the aforementioned “action” and “object” categories, along with “state-based ADL” and “event-based ADL” (Sec. 5.3), and an “overall ADL” (aka. “ADL”) which is the superset of the 4 categories. Our EgoADL dataset contains 35 state-based and 186 event-based ADL classes, the former typically have more labeled data samples because state-based ADL last longer. We use two captioning metrics to measure the similarity between predicted and reference captions [59], *i.e.*, BLEU and SPICE, which are based on n-gram overlapping and scene graph similarity, respectively.

5.8.2 Micro Benchmark Analysis of EgoADL Supervised Learning Model

We conduct an ablation study to compare the EgoADL design across different modality fusions. As shown in Tab. 5.3, we evaluate EgoADL in 7 settings by using a single modality and combining multiple modalities by using supervised learning models. 5 different methods are evaluated: (i) “Fast-only” for single modality, (ii) “Modalwise” where the multi-modal features are fused along the modality dimension (iii) “Framewise” where the multi-modal features are fused along the frame dimension, (iv) “MMFW” where the multi-modal features are fused along both modality and frame dimensions without audio slow pathway (see Sec. 5.5.3), (v) “MMFWSF” represents to “MMFW” with audio slow pathway (Sec. 5.5.3). For a fair comparison, all the methods are trained using a balanced dataset with 2,500 labeled samples. We employed an 8 : 2 split between training and validation with 5-fold cross validation. For testing, we use the remaining unbalanced 2,800 samples. It is important to note that this unbalanced testing set does not skew our final results, as all outcomes are reported on a classwise basis (average across different ADL). The distribution of samples from all 10 users is balanced across the training, validation, and testing sets.

Performance gain due to multiple modalities: As shown in Tab. 5.3, “Audio” is the

most informative modality among these three modalities. Compared to audio-only solution, the multi-modal fusion achieves an overall 13.5% and 16.6% improvement for top-1 and top-5 overall ADL mAP, respectively. Fig. 5.18 demonstrates the performance gain introduced by the EgoADL multi-modal fusion design. With the motion sensor located at users’ trouser pocket, EgoADL is able to recognize more actions and objects related to lower body, including “Sit in a chair”, “Bend down”, “Mop the floor”, etc. Wi-Fi CSI further characterizes the full-body motions with large environment changes. ADL, like “Open/close the door”, “Open/close the window”, etc., are easily recognized through EgoADL’s multi-modal fusion. Besides, we also find that, compared to the “Audio” which is effective in identifying *event-based ADL*, both “Motion” and “Wi-Fi” have advantages in recognizing *state-based ADL*, which may involve periodic motions. As shown in Fig. 5.18, the multi-modal fusion only incurs slightly lower accuracy for few ADL, like “Dry hand”, “Grab tissue”, etc.

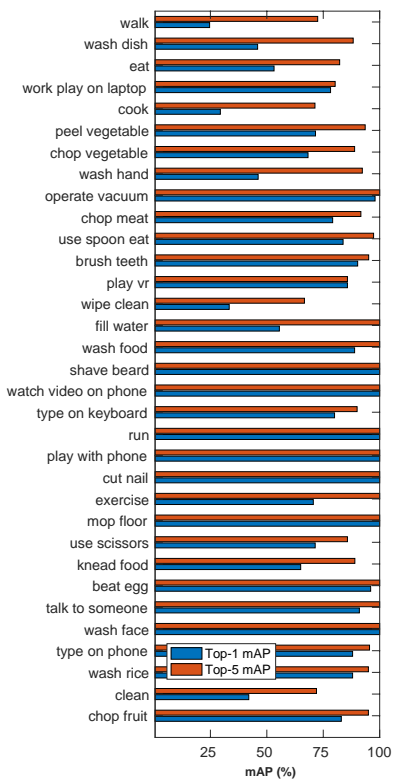
Performance gain due to MMFWSF fusion design: As shown in Tab. 5.3, compared to traditional modalwise/ framewise fusion algorithms, our MMFWSF fusion achieves better performance for all kinds of ADL. By utilizing the complementary modalities of different modalities, EgoADL further pushes the limits to achieve 55.3% and 78.1% top-1 and top-5 overall ADL mAP. Further, the 95% confidence intervals for the mAP across all categories are within $\pm 2.5\%$, indicating the consistent performance of EgoADL.

5.8.3 Accuracy, Generalization and Extensibility of EgoADL SSL Model

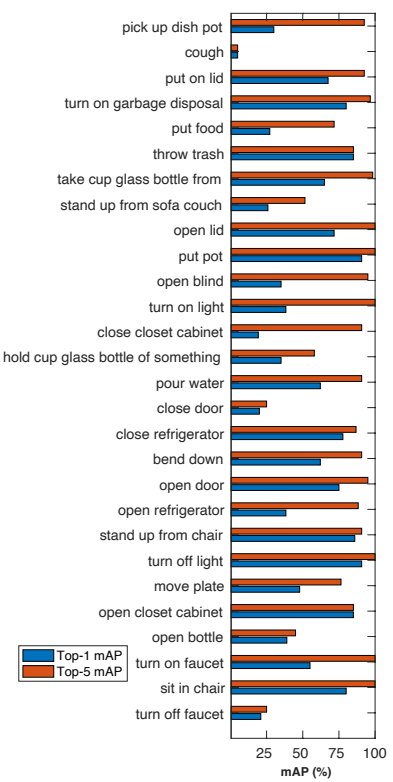
In this section, we evaluate the EgoADL models using various SSL training methods (Sec. 5.6), focusing on improvements in accuracy, generalization, and extensibility. The EgoADL models are trained by 3 different training approaches: *i*). “W/o” for without SSL, *ii*). “SM” for single-modal SSL (Sec. 5.6.1), *iii*). “CM” for cross-modal SSL (Sec. 5.6.2). Initially, models are pretrained using 100 hours of unlabeled data from 20 additional subjects. This is followed by fine-tuning the models using a balanced dataset comprising 2,500 samples for training, and an unbalanced dataset with 2,800 samples for testing, with the same 5-fold cross validation setup

Table 5.3. EgoADL micro benchmark. We benchmark 5 categories of ADLs. “ADL”, “A”, “O”, “S”, “E” represent to “Overall ADL”, “Action”, “Object”, “State” and “Event”, respectively (see Sec. 5.3). “Refine” represents the case where the non-ambiguous labels for non-visual sensors, as discussed in Sec. 5.8.4. The number below the mAP result is the 95% confidence interval for the mAP.

Modal	Methods	Top-1 mAP (%)					Top-5 mAP					Captioning	
		ADL	A	O	S	E	ADL	A	O	S	E	BLEU	SPICE
Audio	Fast-only	44.1	56.0	54.3	71.3	31.7	62.6	75.3	67.1	82.4	53.5	0.42	0.40
	Fast-only	25.9	37.0	36.1	49.7	15.0	51.6	71.9	57.8	80.4	38.3	0.29	0.28
	Fast-only	23.0	32.8	30.0	41.6	14.5	44.5	53.9	51.2	61.6	36.6	0.28	0.23
Audio + Motion	Modalwise	50.7	61.1	61.0	73.0	40.5	67.2	80.8	81.7	90.5	56.5	0.48	0.46
	Framewise	46.4	59.1	55.8	71.7	34.8	67.1	79.2	77.1	87.0	57.9	0.46	0.44
	MMFW	51.5	63.8	61.9	73.8	41.3	67.4	80.7	78.3	89.4	57.3	0.48	0.45
	MMFWSF	52.1	64.0	62.0	75.0	41.6	67.8	81.7	78.5	90.2	57.5	0.49	0.45
Audio + Wi-Fi	Modalwise	47.4	58.5	59.2	71.3	36.4	67.8	78.5	81.3	90.5	57.3	0.46	0.43
	Framewise	47.9	57.8	56.3	72.8	36.5	64.0	74.5	78.6	86.5	53.7	0.44	0.43
	MMFW	48.6	62.6	62.9	73.1	37.4	70.1	82.3	81.6	90.2	60.8	0.49	0.46
	MMFWSF	49.2	62.9	63.5	74.5	37.5	71.0	82.5	81.9	90.5	62.1	0.49	0.46
Wi-Fi + Motion	Modalwise	40.2	51.9	46.9	66.2	28.2	64.4	78.1	72.1	84.2	55.2	0.43	0.41
	Framewise	41.1	50.6	46.9	66.8	29.3	64.6	78.9	72.6	84.5	55.5	0.43	0.41
	MMFW	41.3	53.1	48.0	66.8	29.5	65.0	78.2	73.1	83.9	56.1	0.43	0.41
	Modalwise	51.5	63.4	62.7	73.8	41.2	76.1	85.2	86.9	90.8	68.3	0.52	0.51
Audio+ Motion+ Wi-Fi	Framewise	53.8	66.7	64.1	76.7	43.2	74.7	86.5	87.1	90.6	67.4	0.53	0.52
	MMFW	54.6	67.6	66.2	76.0	44.8	76.6	86.6	88.0	90.7	70.1	0.54	0.52
	MMFWSF	± 2.1	± 1.8	± 1.7	± 1.5	± 2.8	± 1.9	± 1.2	± 1.5	± 1.2	± 2.5	± 0.02	± 0.02
		55.3	68.1	65.8	77.0	45.3	78.1	87.8	88.4	92.5	71.5	0.56	0.52
	± 2.0	± 1.5	± 1.9	± 1.3	± 2.4	± 1.2	± 0.9	± 1.1	± 1.0	± 1.4	± 0.02	± 0.01	
Refine	68.2	70.4	72.2	83.2	55.9	87.3	90.1	92.3	94.8	82.7	0.65	0.63	
	± 1.5	± 1.6	± 1.4	± 0.8	± 2.0	± 1.3	± 1.9	± 1.4	± 1.0	± 2.2	± 0.01	± 0.01	



(a) State-based human behaviors



(b) Event-based human behaviors

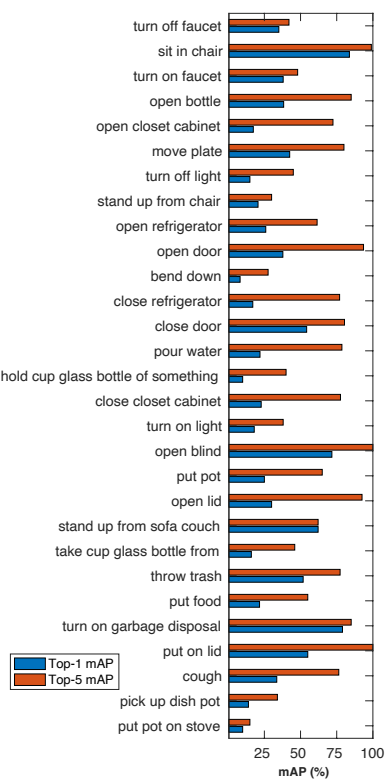
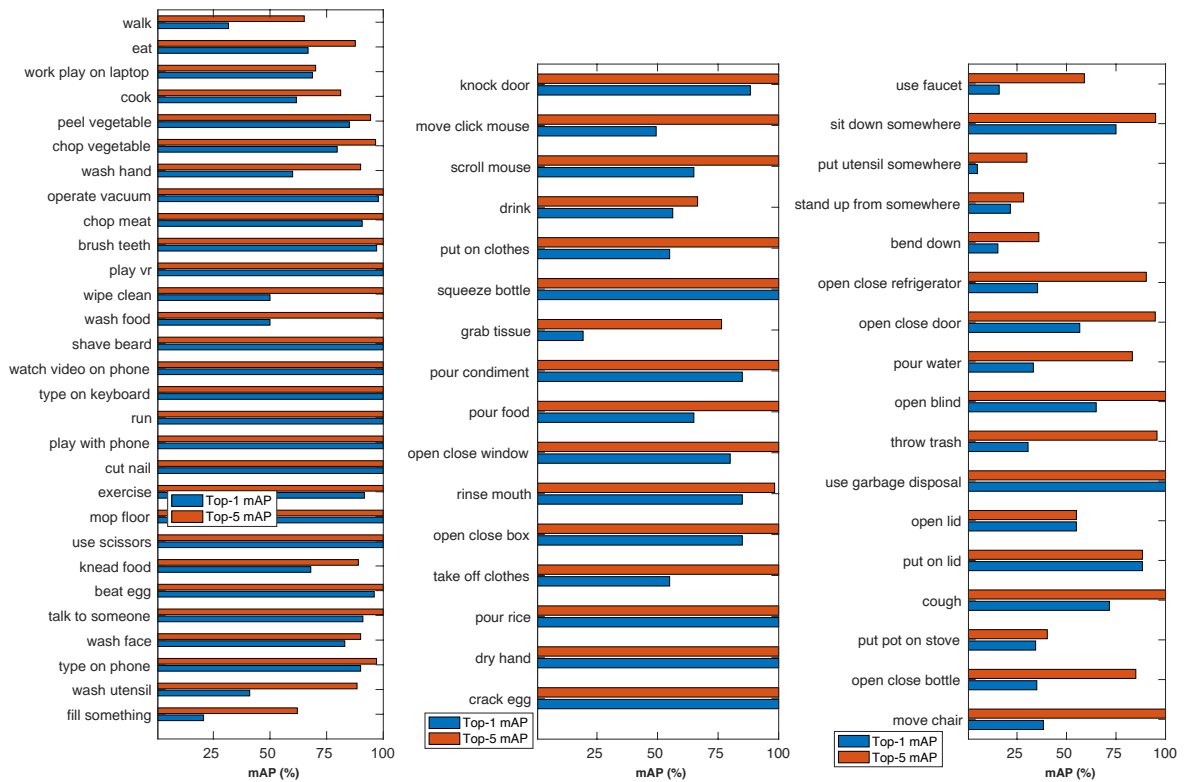


Figure 5.16. EgoADL classwise mAP



(a) State-based human behaviors

(b) Event-based human behaviors

Figure 5.17. EgoADL classwise mAP with label refinement

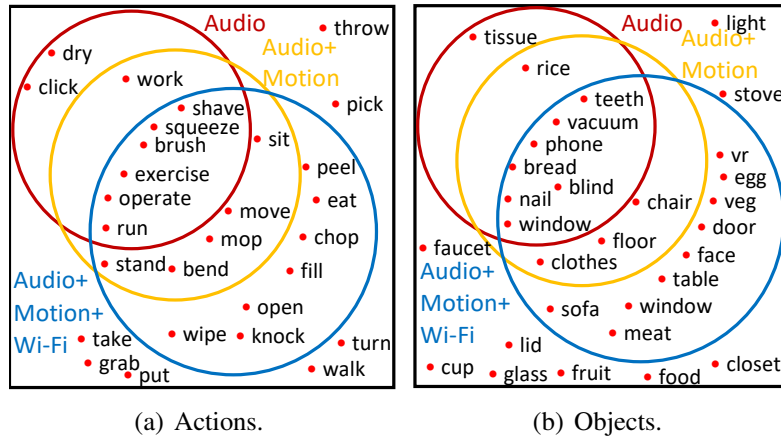


Figure 5.18. Venn diagram visualizes the advantages and limitations of EgoADL multi-modal fusion. Actions/ objects with $> 80\%$ top-1 mAP are in the circle for different modality fusions. described in the previous section.

Accuracy gain due to SSL: Table 5.6 shows that leveraging SSL into EgoADL yields a marginal Top-1 mAP enhancement across various ADL categories when compared to the non-SSL EgoADL models. Notably, the adoption of cross-modal SSL (Sec. 5.6.2) allows EgoADL to capitalize on audio pseudo labels derived from pre-trained audio feature embeddings, culminating in a further 2.9% mAP increment for the overall ADL.

Generalization gain due to SSL: For generalization evaluation, the models are trained and tested using a leave-one-out cross-validation approach, to evaluate on unseen users (UU) and unseen environments (UE). Table 5.5 shows the top-1 mAP. The baseline model shows poor generalization, with 35.3%, 27.7% top-1 “ADL” mAP and large 95% confidence interval for UU and UU+UE respectively. In contrast, with the “Cross-modal SSL” design, EgoADL achieves 47.7% and 47.1% in the same metrics with a smaller 95% confidence interval. Additionally, when we fine-tune the EgoADL model with 200 labeled samples for a specific user (“Personalized fine tuning” in Tab. 5.5), the model reaches 77.1% and 91.5% top-1 mAP for “ADL” and “state” respectively.

Extensibility gain due to SSL: For extensibility assessment, we focus on 116 event-based “tail” ADL, which refers to the ADL with less than 15 data samples in our labeled dataset.

Table 5.4. Accuracy of EgoADL SSL. “W/o”, “SM”, “CM” represent to “W/o SSL”, “Single-modal SSL”, “Cross-modal SSL”, respectively. “ADL”, “A”, “O”, “S”, “E” represent to “Overall ADL”, “Action”, “Object”, “State” and “Event”, respectively. The number below the mAP result is the 95% confidence interval for the mAP.

	Top-1 (%)					Top-5 (%)				
	ADL	A	O	S	E	ADL	A	O	S	E
W/o	55.3 ± 2.0	68.1 ± 1.5	65.8 ± 1.9	77.0 ± 1.3	45.3 ± 2.4	78.1 ± 1.2	87.8 ± 0.9	88.4 ± 1.1	92.5 ± 1.0	71.5 ± 1.4
SM	56.3 ± 1.3	67.5 ± 1.0	66.9 ± 1.2	77.5 ± 0.9	46.6 ± 1.6	78.5 ± 0.8	87.1 ± 1.0	89.5 ± 1.3	93.0 ± 0.7	71.8 ± 1.3
CM	59.2 ± 1.0	68.5 ± 1.2	68.9 ± 1.4	79.5 ± 0.8	49.8 ± 1.1	79.8 ± 1.0	87.6 ± 1.1	90.6 ± 1.2	92.8 ± 0.6	73.8 ± 1.3

Training and validation utilize a balanced set of 2,500 labeled samples, supplemented by 1,000 “tail” ADL samples. In total, 3,500 out of 7,000 labeled samples form the training set while the remaining 3,500 unbalanced labeled samples form the test set. As shown in Tab. 5.6, our SSL approach yields substantial increases in mAP for these “tail” classes: 39.8% top-1 and 62.3% top-5. Further fine-tuning with an additional 3 samples per “tail” class significantly boosts these metrics to 60.2% top-1 and 74.1% top-5.

5.8.4 Evaluating the Limits of Non-Visual Sensors

To understand the limits of non-visual sensors, we compare EgoADL with egocentric vision-based methods. We reimplemented the SOTA methods, *i.e.*, Ego-exo [135], on two egocentric vision datasets, *i.e.*, Charades-Ego [201] and GTEA-GAZE [157]. The ADL sets differ across datasets. For a fair comparison, we only consider the classes which have the same labels in both the egocentric vision and EgoADL datasets (see detailed ADL sets in Fig. 5.19). For EgoADL, we use the model trained in the previous section. For Ego-exo [135], we employed models fine-tuned with the Charades-Ego [201] and GTEA-GAZE [157] datasets. The models are evaluated not only on the egocentric video dataset but also on the egocentric video data from the EgoADL dataset, which is originally used for labeling. For a fair comparison, all evaluations are conducted in the “UU + UE” scenario.

Table 5.5. Generalization of EgoADL SSL. “UU” and “UE” represent to “Unseen user” and “Unseen environment”. “W/o”, “SM”, “CM” represent to “W/o SSL”, “Single-modal SSL”, “Cross-modal SSL”, respectively. “P” represents to “Personalized fine tuning”. “ADL”, “A”, “O”, “S”, “E” represent to “Overall ADL”, “Action”, “Object”, “State” and “Event”, respectively. The number below the mAP result are the 95% confidence interval for the mAP.

	UU (Top-1 mAP %)					UU + UE (Top-1 mAP %)				
	ADL	A	O	S	E	ADL	A	O	S	E
W/o	35.3 ± 6.3	46.3 ± 5.5	36.6 ± 6.9	48.5 ± 3.3	29.3 ± 7.4	27.7 ± 6.6	36.6 ± 5.8	34.5 ± 7.0	41.3 ± 4.1	21.5 ± 8.0
SM	41.9 ± 4.1	52.6 ± 3.7	54.5 ± 4.0	60.3 ± 2.8	33.4 ± 4.8	42.8 ± 4.4	48.5 ± 3.4	52.1 ± 4.0	65.5 ± 3.5	32.2 ± 5.0
CM	47.7 ± 3.5	56.1 ± 3.3	62.5 ± 4.0	69.7 ± 2.9	37.6 ± 4.5	47.1 ± 3.8	55.8 ± 3.2	60.9 ± 3.9	68.5 ± 2.0	37.2 ± 4.0
P	77.3 ± 2.3	88.2 ± 2.1	87.3 ± 1.8	93.0 ± 1.3	70.1 ± 2.7	77.1 ± 2.5	87.9 ± 2.0	87.5 ± 2.4	91.5 ± 1.9	70.5 ± 3.0

Table 5.6. Extensibility of EgoADL SSL. “W/o”, “SM”, “CM” represent to “W/o SSL”, “Single-modal SSL”, “Cross-modal SSL”, respectively. “P” represents to “Personalized fine tuning”. “E”, “A”, “O” represent to “Event”, “Action”, “Object”.

	Tail Classes					
	Top-1 (%)			Top-5 (%)		
	E	A	O	E	A	O
W/o	22.8	38.5	41.2	43.2	50.2	53.5
SM	35.6	51.9	55.0	58.2	62.3	65.5
CM	39.8	53.5	58.3	62.3	70.9	72.3
P	60.2	69.3	71.2	74.1	80.2	83.1

Tab. 5.7 shows the results compared with egocentric vision. We found that testing on the egocentric vision data collected by ourselves (see “EgoADL vision” in Tab. 5.7) has much worse performance than testing on existing egocentric vision datasets. This is because that egocentric vision datasets require the users to adjust the camera FoV or use specialized cameras with a larger FoV to capture the subject hand and interaction objects. In contrast, our collected egocentric vision data will only be used by data labeling. Therefore, we use a commodity camera with limited FoV. Part of our egocentric vision data is not able to capture the hand motion and

Table 5.7. Comparison between EgoADL and egocentric vision. “ADL”, “A”, “O”, “S”, “E” represent to “Overall ADL”, “Action”, “Object”, “State” and “Event”, respectively. “EgoADL vision” means that we evaluate egocentric vision models using the testing video data collected by ourselves, which is originally used for labeling.

	# of ADL	Top-1 mAP (%)				
		ADL	A	O	S	E
EgoADL v.s. Charades-Ego v.s. EgoADL vision	40 (105)	48.8	56.3	57.5	46.4	50.4
EgoADL v.s. EGTEA-GAZE v.s. EgoADL vision	40 (157)	39.9	41.2	34.5	25.0	46.3
	40 (105)	31.1	40.5	31.3	21.1	35.3
	40 (105)	47.3	54.8	50.0	65.3	48.3
	40 (106)	46.9	57.9	56.4	67.3	42.2
	40 (105)	40.0	42.6	41.8	40.3	39.8

interaction object of the subject. This does not affect the data labeling as users can remember what they are doing and label the ADLs based on their memory. However, directly using such data will unfairly degrade the performance of other datasets. We also evaluate egocentric vision using their dataset and compare the results with EgoADL. *We found that EgoADL achieves comparable performance with vision-based methods for the overlapped classes between the egocentric vision datasets and the EgoADL datasets.* The performance of both highly depends on the label type and granularity (Tab. 5.7), because they both have unique advantages and limitations, which we summarize as follows:

Adv: EgoADL shows remarkable advantages in recognizing state-based ADL with unique motion patterns or sound events. It achieves a top-1 “state” mAP improvement of 21.4% over Charades-Ego [201], as most state-based ADL cannot be entirely captured by egocentric vision with limited field of view.

Limit1: EgoADL is limited in recognizing ambiguous actions, i.e., actions that are similar to non-visual sensors but can be described by natural language in different ways. For example, human actions, *i.e.*, “grab”, “put”, “take”, “hold”, “pick”, “throw”, all involve humans using their hands to fetch something. Our classwise detailed experiments in Fig. 5.18(a) and Fig. 5.16 show that EgoADL can only achieve < 30% mAP on average to distinguish these few actions. Further

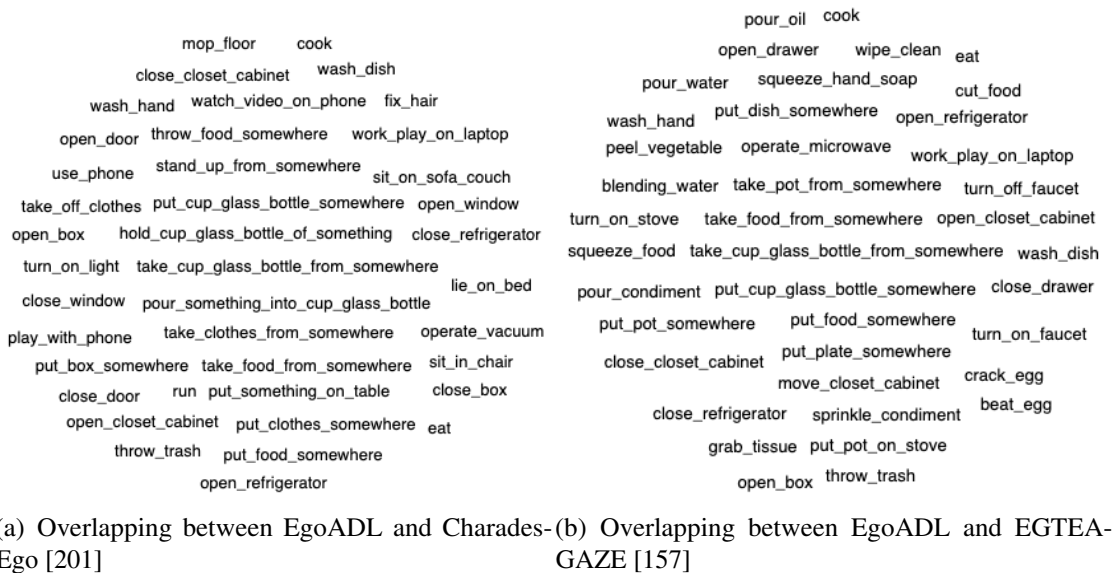


Figure 5.19. Overlapped ADL labels between EgoADL and egocentric vision

such actions are not only hard to recognize for non-visual sensors, but also for vision-based methods without detailed contextual information [157].

Limit2: Without vision information, EgoADL is limited to recognize detailed objects. Although audio can capture the specific sound of human-object interaction to distinguish different objects, without vision information, non-visual sensors can only recognize the object with coarse granularity. For instance, when subjects are chopping something in the kitchen, the vision-based methods will be able to recognize the detailed type of objects, like “carrot”, “potato”, “watermelon”, “beef”, *etc.* However, EgoADL can only recognize the “chop” action but not the type of objects. In EgoADL, we do not label the objects with such granularity. Thus, in Tab. 5.7, EgoADL even achieves higher performance for “object” mAP as it can recognize many objects outside camera FoV.

5.8.5 Evaluation on Knowledge Distillation from Natural Language

Label refinement for non-visual sensors: We follow the steps discussed in Sec. 5.7.1. The refined labels consist of 75 frequently-used ADL, 38 object interactions, 41 actions, 29 state-based ADL, and 46 event-based ADL. As shown in Tab. 5.3 and Fig. 5.17, with the label

Table 5.8. Performance gain due to EgoLM. “ADL”, “A”, “O”, “S”, “E” represent to “Overall ADL”, “Action”, “Object”, “State” and “Event”, respectively.

Methods	Top-1 mAP (%)			Top-5 mAP (%)						
	ADL	A	O	S	E	ADL	A	O	S	E
EgoADL w/o LM	68.2	70.4	72.2	83.2	55.9	87.3	90.1	92.3	94.8	82.7
EgoADL w/ LM, Word Masking	69.9	70.4	72.2	83.8	59.6	88.5	90.9	92.9	94.6	83.2
EgoADL w/ LM, ADL Masking	72.5	75.3	76.5	85.9	65.8	90.8	92.1	93.4	94.5	83.9

refinement, EgoADL achieves an overall mAP of 68.2%, with 83.2% and 55.9% for state-based and event-based ADL, respectively. This is significantly higher than the labels obtained from egocentric video and audio.

Performance gain due to EgoLM: To evaluate the performance gain due to EgoLM, we need to use continuous testing samples in the time domain since EgoLM takes the prediction text of EgoADL in a long period (30 s) as the input to learn the contextual information. So we use 24 continuous recordings, each lasting 5-min, as the EgoLM testing dataset (2 hours in total). Tab. 5.8 summarizes the results. We evaluate the EgoLM with two different masking strategies, masking *i*). a word-level token, and *ii*). an entire ADL. Tab. 5.8 indicates that masking an entire ADL led to a notably improved performance compared to merely masking word-level tokens. This outcome suggests that masking complete ADL is more effective in enabling the EgoLM model to grasp the contextual relationships integral to ADL. EgoLM gains an additional 4.3% (from 68.2% to 72.5%) top-1 overall ADL mAP for EgoADL, matching the intuition that EgoLM can better understand the contextual information when the original model performance is sufficiently high. Besides, EgoLM is proficient in enhancing the mAP of event-based ADL which tend to have more contextual information.

5.8.6 Energy Consumption

In this section, we evaluate the energy consumption associated with the sensing capabilities of EgoADL, while a detailed discussion on computational resource consumption is provided in Sec. 5.9. We conduct a preliminary profiling of the EgoADL sensor data capturing app by using Android’s native battery usage measurement. During the measurement, EgoADL collects the audio recordings, Wi-Fi CSI and motion sensor signals in the background with the display off. We found that *EgoADL only consumes less than 60 mAh per hour on a Nexus 5 smartphone*. That means EgoADL can work on a Nexus 5 with 2300 mAh battery for about 7.6 days if it continuously records the multi-modal sensing data for 5 hours a day. All the 3 sensor modalities are significantly more energy efficient than a camera (more than 600 mAh per hour) [185],

making it a promising potential in practical scenarios.

5.9 Discussion and Limitations

Privacy consideration: EgoADL requires capturing the egocentric audio signals, which may inadvertently include users' daily conversations. However, thanks to the EgoADL DNN design, we can separate the audio branch from the whole model, and calculate the audio feature embedding on-device without uploading raw audio data to the edge/ cloud devices to protect users' privacy. To this end, we can employ the model designed to be deployed on smartphones or edge device [196] as the basic feature embedding network. Another potential solution is to selectively anonymize or mask the speech data [182] from audio recording. These privacy enhancement mechanisms are left for our future exploration.

Generalization of EgoADL dataset: Due to the availability of Wi-Fi sensing, one of the limitations of EgoADL is that the dataset is only collected by a single type of commodity smartphone (*i.e.*, Nexus 5). However, it will not significantly affect the generalizability of the dataset because of the following reasons: *i*). There is no limitation imposed on the placement of Wi-Fi AP nor on the manner in which users carry the smartphones in their trouser pockets when collecting the data. Therefore, this leads to greater variability in the data than the type of device used, owing to the variability in Wi-Fi AP locations, which can vary by several meters, and the differences in smartphone Wi-Fi antenna positions, which can vary by several centimeters. *ii*). We focus on the Wi-Fi CSI Doppler shift induced by human motion and environment factors. Given that the Wi-Fi signal's wavelength at a frequency of 5 GHz is approximately 6 cm, the resultant Doppler shift predominantly reflects motions with displacements on the order of tens of centimeters. Therefore, such features are not significantly influenced by variations between different smartphone models. Another limitation of EgoADL dataset is the demographics of participants (see Sec. 3.8.1). We hope that, by open-sourcing EgoADL, we can encourage a broader spectrum of participants and researchers to contribute to the EgoADL data collection,

thereby enhancing the demographic diversity of our dataset.

Potential Missing Data: Generally, smartphones are capable of continuously capturing both audio and motion sensor data with minimal data loss. However, due to packet losses, the receiving packet rate of Wi-Fi CSI tends to be lower, especially when there is significant distance between the subject and Wi-Fi AP. In our data collection within apartments up to 2000 ft², and where the distance between the Wi-Fi AP and the smartphone is impeded by fewer than two solid walls, we observed negligible loss of Wi-Fi packets. Conversely, in scenarios where the distance exceeds 15 meters or involves more than three solid walls, we noted a notable decrease in packet reception, resulting in a lower packet rate for Wi-Fi CSI. To ensure the data quality, we ensured a minimum packet reception rate of 200 packets per second on smartphones during data collection, corresponding to a 50 Hz Doppler shift akin to daily human motion maximum speed (about 3 m/s). In practical scenarios, if reception rates drop below this threshold, we can alternatively use EgoADL models that operate without Wi-Fi CSI data requirement.

System resource consumption of EgoADL: EgoADL focuses on improve the performance of ADL sensing performance, rather than optimizing the system resource usage. Currently, the computational resource requirement is relatively high. We notice that most of the parameters (275.5 M) are contributed by the self-supervised feature embedding VGG-like DNN models (Sec. 5.6). We plan to replace them using more efficient DNN models, like MobileNet [196], without losing significant accuracy. Further, a full-fledged implementation of EgoADL needs to carefully split the on-device vs. in-cloud processing, and strikes a balance between computation and communication energy cost. This is left for our future work.

Applicability for EgoADL device: Currently, smartphones serve as the device for EgoADL, primarily chosen for their availability to capture Wi-Fi CSI. However, it is noted that smartphone is not always carried by users, particularly among the elderly population. We recognize that wearable devices, such as smartwatches, may present a more suitable option for EgoADL. One promising direction to explore in future work is to include a more diverse set of wireless sensors, *i.e.*, low-cost ultrasound sonar, UWB radar or mmWave radar, on wearable

devices [249].

Integrating Large Language Model (LLM) into ADL sensing: EgoADL fine-tunes a language model, *i.e.*, BERT [64], to extract and distill contextual information pertaining to human behaviors, as detailed in Section 5.7.2. While the current implementation deals with computational complexities by fine-tuning a more manageable model size, the approach can be scaled to accommodate the fine-tuning of a larger language model in the future. Moreover, given that large language models are designed for general natural language processing tasks, it may be feasible for EgoADL to bypass fine-tuning altogether. Instead, EgoADL could provide its generated sequence of words and the corresponding probability distribution and organize them as the input prompt directly to the LLM. This would allow the LLM to employ its robust contextual capabilities to refine and correct the word sequence autonomously. We posit that EgoADL paves the way for a novel integration of sensory data with natural language processing. Moving forward, our research will explore to leverage LLMs to enhance the perception and understanding of ADL through different sensing technologies.

5.10 Conclusion

This paper presents the first study that uses a commodity smartphone as an egocentric multi-modal sensor hub to recognize unrestricted user behaviors in free living environment. Although the absolute sensing accuracy of the proposed EgoADL system still leaves room for improvement, its performance is already comparable to state-of-the-art egocentric vision-based solutions. EgoADL verifies several promising mechanisms, such as a joint design of self-supervised single-modal and multi-modal clustering, and context distillation from generic data, which can overcome the fundamental barriers—particularly the generalization and labeling—in ubiquitous sensor-based behavior analysis. Our EgoADL dataset will be released as open source to promote research in both ubiquitous computing and machine learning.

5.11 Acknowledgments

This chapter contains material from “Multimodal Daily-life Logging in Free-living Environments Using Non-Visual Egocentric Sensors on a Smartphone”, by Ke Sun, Chuyu Xia, Xinyu Zhang, Hao Chen, and Charlie Zhang, which appears in the Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), Volume 8, Issue 1, 2024 [214]. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Conclusion and Future Work

Acoustic sensors and actuators, such as loudspeakers and microphones, are some of the most common components in consumer electronic devices and play a crucial role in enabling ambient intelligence. Traditionally, these components have been used for sound-related tasks like voice-user interfaces, sound playback, and event detection. However, with the increasing demand for consumer electronics to provide more advanced, user-friendly, and powerful ambient intelligence, there is a growing need to unlock the full potential of these components for cross-modality sensing applications. By repurposing acoustic sensors and actuators, devices can achieve intelligent, cost-effective, human-centric, and trustworthy ambient intelligence, while reducing costs, energy consumption, and computational overhead.

6.1 Dissertation Conclusion

In this dissertation, I argue that advanced signal processing techniques and deep learning models can be used to repurpose acoustic components for cross-modality sensing applications, achieving resolutions comparable to dedicated sensors. By comprehensively understanding the strengths and limitations of acoustic sensors and actuators, I design end-to-end cross-modal sensing systems that span from the application level to the hardware level. This includes hardware design, sensor placement optimization, advanced signal processing techniques, deep neural network architectures, and overall system optimization.

First, I investigate the speech privacy issue in ubiquitous acoustic devices using cross-modality sensing. In MicShield, I design a speech privacy protection system that functions as a companion to voice assistants. It automatically safeguards speech privacy by selectively jamming unintended private speech while allowing legitimate voice commands to pass through. I propose a novel speech processing pipeline which leverages the framewise likelihood to detect the onset of a wake words, thus realizing selective jamming. I employ ultrasound-based inaudible voice generation, producing sound that is imperceptible to humans but detectable by microphones, thereby ensuring no impact on the core functionalities of the device. Our evaluation confirms that the 3D-printed shield effectively protects speech privacy for various voice assistants, including those with microphone arrays. In StealthyIMU, I expose a new threat that enables a zero-permission app to steal private information from voice user interface (VUI) responses on a smartphone. This attack exploits a side channel, where a motion sensor “overhears” low-frequency vibrations from the co-located loudspeaker. Our case studies demonstrate that StealthyIMU can accurately extract sensitive, permission-protected information—such as contacts, search history, calendar events, home addresses, and GPS routes—from widely used VUIs like Google Assistant and Google Maps. Additionally, I propose effective defense mechanisms to help VUI vendors mitigate this vulnerability. These two works highlight that speech privacy can be compromised not only by always-on microphones but also by cross-modal side-channel sensors. The risk becomes even more significant when contextual information is combined with speech leakage. Therefore, addressing these privacy threats while preserving the functionality of ubiquitous acoustic devices is a critical challenge moving forward.

Second, I repropose the acoustic sensors and actuators as new sensing modality for cross-modal sensing applications by developing multi-modal signal processing and deep learning algorithms. I tackle the open challenge of speech processing, i.e., the single-channel, audio-only speech separation and enhancement problem, by using a single pair of microphone and loudspeaker. I propose UltraSE, which leverages ultrasound sensing as a complementary modality to capture the speaker’s articulatory gestures and separate the desired speaker’s voice from

interference and noise. I design a multi-modal multi-domain DNN framework for single-channel speech enhancement which fuses the ultrasound and speech features, and simultaneously improves speech intelligibility and quality. I further introduce a cGAN-based cross-modal training model which effectively captures the correlation between ultrasound and speech for multi-modal denoising. In EgoADL, I introduce the first egocentric human activities of daily living (ADL) sensing system that uses an in-pocket smartphone as a multi-modal sensor hub to capture body motion, interactions with the physical environment and daily objects using non-visual sensors (audio, wireless sensing, and motion sensors). I designed multi-modal frame-wise slow-fast encoders to learn feature representations that capture the complementary strengths of various sensing modalities. Additionally, I adapt a transformer-based sequence-to-sequence model to decode time-series sensor signals into sequences of words representing ADLs. To address the challenge of limited labeled data and enhance generalization, we introduced a self-supervised learning framework that extracts intrinsic supervisory signals from multi-modal sensor data. These works demonstrate how external sensors and modalities can be leveraged to unlock the potential of traditional acoustic sensors, enabling novel sensing applications with enhanced resolution and improved generalization.

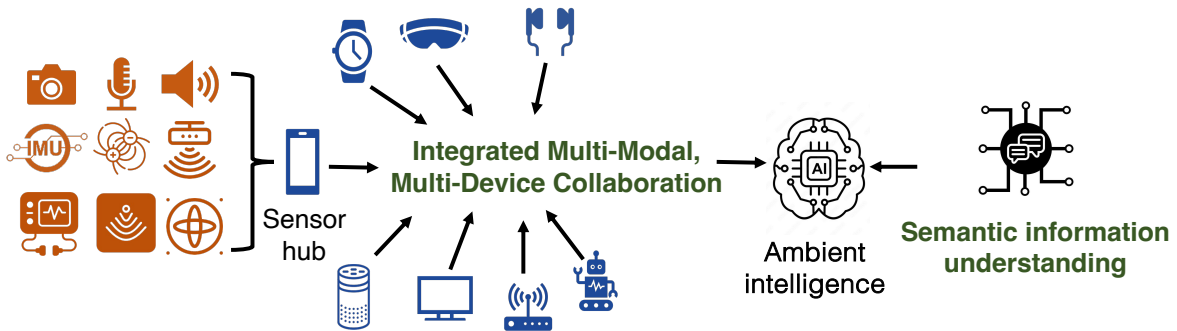


Figure 6.1. Ambient Intelligence Vision

6.2 Future Work

My previous work has demonstrated the potential of repurposing acoustic components in ubiquitous consumer electronic devices for cross-modality sensing. Building on this foundation, I aim to explore new research challenges centered on designing multi-device and multi-modal systems that enable intelligent, cost-effective, human-centric, and trustworthy mobile, wearable, and IoT solutions. Below, I outline four key research directions that I will pursue in the near future.

6.2.1 Multi-device Multi-modal Sensing on Ubiquitous Acoustic Devices

In this dissertation, my primary focus has been exploring the potential of acoustic sensors and actuators for cross-modality sensing. A natural extension of this work is to explore multi-device and multi-modal scenarios. As shown in Fig. 6.1, modern personal smart devices increasingly act as sensor hubs, integrating a variety of sensors and actuators. My goal is to unlock the potential of these distributed sensor hubs to realize the vision of ambient intelligence. For instance, I am working on extending the EgoADL framework to a multi-device setup for comprehensive daily life logging. By integrating data from personal wearable devices such as smartphones, smartwatches, and smart earbuds—each attached to different parts of the body—these systems will have a more holistic understanding of body motion, human-object interactions, and how the body reacts to different environments and scenarios.

6.2.2 Resource-efficient Cross-Modality Sensing on Ubiquitous Acoustic Devices

One of the primary challenges in deploying the multi-modal sensing systems developed in this dissertation is the limited resources available on mobile, wearable, and IoT devices. These devices are constrained by their small form factor, limiting computational power, battery life, and communication bandwidth. For example, multi-modal sensing deep neural network models are computationally intensive and difficult to deploy on resource-constrained platforms like

smartwatches and earbuds. Achieving the vision of 24/7 ambient intelligence becomes even more challenging due to energy consumption constraints.

To address these limitations, my future research will focus on developing resource-efficient cross-modality sensing techniques for ubiquitous acoustic devices. By gaining a deeper understanding of the capabilities of different sensors and actuators, I aim to design systems that dynamically manage sensor usage, communication resources, and computational models. This approach will enable intelligent sensing applications to be deployed on resource-constrained platforms, making 24/7 ambient intelligence more feasible and efficient.

6.2.3 Enabling Human-AI-Sensor Interaction for Acoustic Sensing and Privacy Protection

As sensors on personal devices grow more sophisticated, their ability to capture vast amounts of data is rapidly expanding. This creates a powerful sensing environment but also introduces significant privacy challenges, as demonstrated by the MicShield and StealthyIMU. Cross-modal side channels inadvertently capture sensitive information, blurring the line between sensor functionality and privacy protection. Currently, sensor privacy models on smart devices are limited to binary ON/OFF permission settings, which do not adapt to changing user preferences or situational contexts.

In response to this gap, my future research will focus on creating adaptive, user-centric privacy systems that build up sensor privacy knowledge bases and leverage large language models to enable dynamic control over sensor data. This Human-AI-Sensor interaction framework will allow users to manage their data privacy in real-time, based on personal preferences and the evolving capabilities of sensors. By advancing this approach, I aim to provide a flexible, comprehensive solution for safeguarding personal data without sacrificing the functionality of modern sensing systems.

6.2.4 Decoding Semantic Boundaries for Cross-Modality Sensing

Looking forward, I plan to shift focus from simply sensing physical metrics to interpreting and understanding their semantic meaning. One of the key questions I aim to address is: “How can ambient intelligence leverage sensing results to meaningfully enhance user experiences?” Given the varying granularity of different sensors, relying on a single sensor type often limits the depth of insight that can be achieved.

To overcome this, my research will bridge multi-device, multi-modal sensor data with natural language understanding, decoding the semantic boundaries from various sensors or through sensor fusion across multiple devices. EgoADL has taken a pioneering step in this direction by using language models to understand human daily life through multi-modal sensory data from smartphones. Moving forward, I will develop a reinforcement learning framework that integrates Large Language Models (LLMs) and multi-modal foundation AI to better interpret the granularity and potential of multi-device, multi-modal sensor fusion, unlocking new opportunities for intelligent, human-centered applications.

Bibliography

- [1] 20 helpful amazon echo voice commands for you to try. <https://www.popsoci.com/20-amazon-echo-voice-commands/>.
- [2] Alexa privacy and data handling overview. <https://d1.awsstatic.com/product-marketing/A4B/White%20Paper%20-%20Alexa%20Privacy%20and%20Data%20Handling%20Overview.pdf>.
- [3] Amazon.com: Echo dot (3rd gen) - smart speaker with alexa - charcoal: Amazon devices. <https://www.amazon.com/Echo-Dot/dp/B07FZ8S74R>.
- [4] Filterless 3w class-d stereo audio amplifier (datasheet). <https://www.diodes.com/assets/Datasheets/PAM8403.pdf>.
- [5] Ina 219 zero-drift, bidirectional current/power monitor with i2c interface. <http://www.ti.com/lit/ds/symlink/ina219.pdf>.
- [6] International standard iec 61672:2003. International Electrotechnical Commission, 2003.
- [7] Noise and hearing loss prevention. <https://www.asha.org/public/hearing/Noise-and-Hearing-Loss-Prevention/>.
- [8] Pimoroni pHAT DAC24-bit/192khz sound card. <https://shop.pimoroni.com/products/phat-dac>.
- [9] Environmental health criteria – ultrasound, 1982. <https://apps.who.int/iris/bitstream/handle/10665/37263/9241540826-eng.pdf?sequence=1&isAllowed=y>.
- [10] Amazon Polly, 2019. <https://aws.amazon.com/polly/>.
- [11] Amazon Transcribe, 2019. <https://aws.amazon.com/transcribe/>.
- [12] CMUSphinx, 2019. <https://cmusphinx.github.io/>.
- [13] Google Cloud Speech-to-Text, 2019. <https://cloud.google.com/speech-to-text/>.
- [14] Google Cloud Text-to-Speech, 2019. <https://cloud.google.com/text-to-speech/>.
- [15] IBM Text-to-Speech, 2019. <https://www.ibm.com/cloud/watson-text-to-speech>.

- [16] The respeaker 6 mic array for raspberry pi, 2019. <https://respeaker.io>.
- [17] The CMU Pronouncing Dictionary, 2019. <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>.
- [18] Theano, 2019. <https://github.com/Theano/Theano>.
- [19] Best smartphones for audio, 2020. <https://www.soundguys.com/best-smartphones-for-audio-16373>.
- [20] Pytorch Mobile, 2020. <https://pytorch.org/mobile/home/>.
- [21] Android Profiler, 2022. <https://developer.android.com/studio/profile>.
- [22] EPIC-KITCHENS-100- 2021 Challenges Report, 2022. <https://epic-kitchens.github.io/Reports/EPIC-KITCHENS-Challenges-2021-Report.pdf>.
- [23] Rebecca Adaimi, Howard Yong, and Edison Thomaz. Ok google, what am i doing? acoustic activity recognition bounded by conversational assistant interactions. *Proceedings of ACM IMMUT*, 2021.
- [24] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. My lips are concealed: Audio-visual speech enhancement through obstructions. In *Proceedings of Interspeech*, 2019.
- [25] Yuvraj Agarwal and Malcolm Hall. Protectmyprivacy: Detecting and mitigating privacy leaks on ios devices using crowdsourcing. In *Proceedings of ACM MobiSys*, 2013.
- [26] Muhammad Ejaz Ahmed, Il-Youp Kwak, Jun Ho Huh, Iljoo Kim, Taekkyung Oh, and Hyoungshick Kim. Void: A fast and light voice liveness detection system. In *Proceedings of USENIX Security Symposium*, 2018.
- [27] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *Proceedings of NeurIPS*, 2021.
- [28] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *Proceedings of NeurIPS*, 2020.
- [29] Amazon. Alexa, echo devices, and your privacy, amazon help & customer service. <https://www.amazon.com/gp/help/customer/display.html?nodeId=GVP69FUJ48X9DK8V>.
- [30] Amazon. Google home mini. <https://www.amazon.com/gp/help/customer/display.html?nodeId=202201630>.
- [31] Amazon.com. Amazon echo. <https://www.amazon.com/echo/>.
- [32] S Abhishek Anand and Nitesh Saxena. Speechless: Analyzing the threat to speech privacy from smartphone motion sensors. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2018.

- [33] S Abhishek Anand, Chen Wang, Jian Liu, Nitesh Saxena, and Yingying Chen. Spearphone: A speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers. *arXiv preprint arXiv:1907.05972*, 2019.
- [34] S Abhishek Anand, Chen Wang, Jian Liu, Nitesh Saxena, and Yingying Chen. Spearphone: a lightweight speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers. In *Proceedings of ACM WiSec*, 2021.
- [35] Amir Anhari. Alexa dataset - build voice-first applications. <https://www.kaggle.com/aanhari/alexa-dataset>.
- [36] Russakovskii Artem. Google is permanently nerfing all home minis because mine spied on everything i said 24/7. <https://www.androidpolice.com/2017/10/10/google-nerfing-home-minis-mine-spied-everything-said-247/#1>.
- [37] autoblog. How to Monitor Your RPM Gauge to Get the Best Performance Out of Your Car, 2016. <https://www.autoblog.com/2016/04/14/how-to-monitor-your-rpm-gauge-to-get-the-best-performance-out-of/>.
- [38] Adam J Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M Smith. Practicality of accelerometer side channels on smartphones. In *Proceedings of ACSAC*, 2012.
- [39] Zhongjie Ba, Tianhang Zheng, Xinyu Zhang, Zhan Qin, Baochun Li, Xue Liu, and Kui Ren. Learning-based practical smartphone eavesdropping with built-in accelerometer. In *Proceedings of NDSS*, 2020.
- [40] Tom Barker, Tuomas Virtanen, and Olivier Delhomme. Ultrasound-coupled semi-supervised nonnegative matrix factorisation for speech enhancement. In *Proceedings of IEEE ICASSP*, 2014.
- [41] Sarnab Bhattacharya, Rebecca Adaimi, and Edison Thomaz. Leveraging sound and wrist motion to detect activities of daily living with commodity smartwatches. *Proceedings of ACM IMMUT*, 2022.
- [42] Logan Blue, Luis Vargas, and Patrick Traynor. Hello, is it me you're looking for? differentiating between human and electronic speakers for voice interface security. In *Proceedings of ACM WiSec*, 2018.
- [43] Igor Bobriakov. Comparison of top 10 speech processing APIs. <https://medium.com/activewizards-machine-learning-company/comparison-of-top-10-speech-processing-apis-2293de1d337f>.
- [44] Steven Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on acoustics, speech, and signal processing*, 1979.
- [45] Catherine P Browman and Louis Goldstein. Articulatory gestures as phonological units. *Phonology*, 1989.

- [46] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *Proceedings of USENIX Security Symposium*, 2016.
- [47] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *Proceedings of IEEE Security and Privacy Workshops (SPW)*, 2018.
- [48] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of ECCV*, 2018.
- [49] Alejandro Cartas, Jordi Luque, Petia Radeva, Carlos Segura, and Mariella Dimiccoli. Seeing and hearing egocentric actions: How much can we learn? In *Proceedings of IEEE/CVF ICCV Workshop*, 2019.
- [50] Antoine Caubrière, Sahar Ghannay, Natalia Tomashenko, Renato De Mori, Antoine Laurent, Emmanuel Morin, and Yannick Estève. Error analysis applied to end-to-end spoken language understanding. In *Proceedings of IEEE ICASSP*, 2020.
- [51] Brian Chen, Andrew Rouditchenko, Kevin Duarte, Hilde Kuehne, Samuel Thomas, Angie Boggust, Rameswar Panda, Brian Kingsbury, Rogerio Feris, David Harwath, et al. Multimodal clustering networks for self-supervised learning from unlabeled videos. In *Proceedings of ICCV*, 2021.
- [52] Fuming Chen, Sheng Li, Yang Zhang, and Jianqi Wang. Detection of the vibration signal from human vocal folds using a 94-ghz millimeter-wave radar. *Sensors*, 2017.
- [53] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 2021.
- [54] Wei-Han Chen and Kannan Srinivasan. Acoustic eavesdropping from passive vibrations via mmwave signals. In *Proceedings of IEEE GLOBECOM*, 2022.
- [55] Yuxin Chen, Huiying Li, Steven Nagels, Zhijing Li, Pedro Lopes, Ben Y Zhao, and Haitao Zheng. Wearable microphone jamming. In *Proceedings of ACM CHI*, 2020.
- [56] Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *Proceedings of IEEE CVPR*, 2017.
- [57] J. Clark and P. C. van Oorschot. Sok: Ssl and https: Revisiting past challenges and evaluating certificate trust model enhancements. In *Proceedings of IEEE Symposium on Security and Privacy*, 2013.
- [58] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv:1805.10190*, 2018.

- [59] Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. Learning to evaluate image captioning. In *Proceedings of IEEE/CVF CVPR*, 2018.
- [60] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, and Will Price. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of ECCV*, 2018.
- [61] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE TPAMI*, 2021.
- [62] Johnson Dave. How to save battery on your samsung galaxy s10 in 4 simple ways. <https://www.businessinsider.com/how-to-save-battery-on-samsung-galaxy-s10>.
- [63] Abe Davis, Michael Rubinstein, Neal Wadhwa, Gautham J Mysore, Fredo Durand, and William T Freeman. The visual microphone: Passive recovery of sound from video. *Proceedings of ACM SigGraph*, 2014.
- [64] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, 2019.
- [65] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*, 2014.
- [66] Mariella Dimiccoli, Juan Marín, and Edison Thomaz. Mitigating bystander privacy concerns in egocentric activity recognition with deep learning and intentional image degradation. *Proceedings of ACM IMWUT*, 2018.
- [67] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [68] John D’Errico. Surface fitting using gridfit. *MathWorks file exchange*, 643, 2005. <https://www.mathworks.com/matlabcentral/fileexchange/8998-surface-fitting-using-gridfit>.
- [69] Yariv Ephraim and David Malah. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on acoustics, speech, and signal processing*, 1984.
- [70] Yariv Ephraim and Harry L Van Trees. A signal subspace approach for speech enhancement. *IEEE Transactions on speech and audio processing*, 1995.
- [71] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. In *Proceedings of ACM SIGGRAPH*, 2018.

- [72] Hakan Erdogan, John R Hershey, Shinji Watanabe, and Jonathan Le Roux. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *Proceedings of IEEE ICASSP*, 2015.
- [73] Hakan Erdogan, John R Hershey, Shinji Watanabe, Michael I Mandel, and Jonathan Le Roux. Improved mvdr beamforming using single-channel mask prediction networks. In *Proceedings of Interspeech*, 2016.
- [74] Lijie Fan, Tianhong Li, Yuan Yuan, and Dina Katabi. In-home daily-life captioning using radio signals. In *Proceedings of ECCV*, 2020.
- [75] Long Fan, Lei Xie, Xinran Lu, Yi Li, Chuyu Wang, and Sanglu Lu. mmmic: Multi-modal speech recognition based on mmwave radar. In *Proceedings of IEEE INFOCOM*, 2023.
- [76] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of IEEE/CVF ICCV*, 2019.
- [77] Aviv Gabbay, Asaph Shamir, and Shmuel Peleg. Visual speech enhancement. In *Proceedings of Interspeech*, 2018.
- [78] Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. Multi-modal transformer for video retrieval. In *Proceedings of ECCV*, 2020.
- [79] Chuhan Gao, Yilong Li, and Xinyu Zhang. Livetag: Sensing human-object interaction through passive chipless wifi tags. In *Proceedings of USENIX NSDI*, 2018.
- [80] Ming Gao, Yajie Liu, Yike Chen, Yimin Li, Zhongjie Ba, Xian Xu, and Jinsong Han. Inertiar: Automatic and device-independent imu-based eavesdropping on smartphones. In *Proceedings of IEEE INFOCOM*, 2022.
- [81] John S Garofolo et al. Darpa timit acoustic-phonetic speech database. *National Institute of Standards and Technology (NIST)*, 15:29–50, 1988.
- [82] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proceedings of IEEE ICASSP*, 2017.
- [83] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. Gripsense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of ACM UIST*, 2012.
- [84] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *Proceedings of Interspeech*, 2021.
- [85] Google. Google home. https://store.google.com/product/google_home.
- [86] Google. Google home mini. https://store.google.com/product/google_home_mini.

- [87] Google. More about data security and privacy on devices that work with assistant. <https://support.google.com/googlenest/answer/7072285?hl=en>.
- [88] Google. Android permission API reference, 2019. <https://developer.android.com/reference/android/Manifest.permission/>.
- [89] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of IEEE/CVF CVPR*, 2022.
- [90] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings of IEEE International Joint Conference on Neural Networks*, 2005.
- [91] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984.
- [92] Francesco Gringoli, Matthias Schulz, Jakob Link, and Matthias Hollick. Free your csi: A channel state information extraction platform for modern wi-fi chipsets. In *Proceedings of ACM WiNTECH*, 2019.
- [93] François Grondin and François Michaud. Lightweight and optimized sound source localization and tracking methods for open and closed microphone array configurations. *Robotics and Autonomous Systems*, 2019.
- [94] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of IEEE CVPR*, 2006.
- [95] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 2008.
- [96] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. Accomplice: Location inference using accelerometers on smartphones. In *Proceedings of IEEE COMSNETS*, 2012.
- [97] Albert Haque, Arnold Milstein, and Li Fei-Fei. Illuminating the dark spaces of healthcare with ambient intelligence. *Nature*, 2020.
- [98] Harish Haresamudram, Irfan Essa, and Thomas Plötz. Contrastive predictive coding for human activity recognition. *Proceedings of ACM IMWUT*, 2021.
- [99] Harish Haresamudram, Irfan Essa, and Thomas Plötz. Assessing the state of self-supervised human activity recognition using wearables. *Proceedings of ACM IMWUT*, 2022.
- [100] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of IEEE ICCV*, 2015.

- [101] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE CVPR*, 2016.
- [102] Yitao He, Junyu Bian, Xinyu Tong, Zihui Qian, Wei Zhu, Xiaohua Tian, and Xinbing Wang. Canceling Inaudible Voice Commands Against Voice Control Systems. In *Proceedings of ACM MobiCom*, 2019.
- [103] John R Hershey and Michael Casey. Audio-visual sound separation via hidden markov models. In *Proceedings of NeurIPS*, 2002.
- [104] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *Proceedings of IEEE ICASSP*, 2016.
- [105] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Proceedings of NeurIPS Deep Learning Workshop*, 2014.
- [106] Guoning Hu and DeLiang Wang. Monaural speech segregation based on pitch tracking and amplitude modulation. *IEEE Transactions on Neural Networks*, 2004.
- [107] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of IEEE CVPR*, 2018.
- [108] Jingyu Hua, Zhenyu Shen, and Sheng Zhong. We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones. *IEEE Transactions on Information Forensics and Security*, 2016.
- [109] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of IEEE CVPR*, 2017.
- [110] Wenjun Jiang, Chenglin Miao, Fenglong Ma, Shuochao Yao, Yaqing Wang, Ye Yuan, Hongfei Xue, Chen Song, Xin Ma, and Dimitrios Koutsonikolas. Towards environment independent device free human activity recognition. In *Proceedings of ACM MobiCom*, 2018.
- [111] Wenjun Jiang, Hongfei Xue, Chenglin Miao, Shiyang Wang, Sen Lin, Chong Tian, Srinivasan Murali, Haochen Hu, Zhi Sun, and Lu Su. Towards 3d human pose construction using wifi. In *Proceedings of ACM MobiCom*, 2020.
- [112] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE TPAMI*, 2020.
- [113] John D. Cutnell, Kenneth W. Johnson, David Young, Shane Stadler. *Physics*. Wiley, 11 edition.
- [114] Don H Johnson and Dan E Dudgeon. *Array signal processing: concepts and techniques*. PTR Prentice Hall Englewood Cliffs, 1993.

- [115] Bjørn Karmann. Project Alias, 2019. <https://www.instructables.com/id/Project-Alias/>.
- [116] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *Proceedings of IEEE/CVF ICCV*, 2019.
- [117] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Slow-fast auditory streams for audio recognition. In *Proceedings of IEEE ICASSP*, 2021.
- [118] Heather Kelly. How to make sure your amazon echo doesn't send secret recordings, 5 2018. <https://money.cnn.com/2018/05/25/technology/amazon-alexa-stop-recording/index.html>.
- [119] Rinat Khusainov, Djamel Azzi, Ifeyinwa E Achumba, and Sebastian D Bersch. Real-time human ambulation, activity, and physiological monitoring: Taxonomy of issues, techniques, applications, challenges and limitations. *Sensors*, 2013.
- [120] Tomi Kinnunen, Evgenia Chernenko, Marko Tuononen, Pasi Fränti, and Haizhou Li. Voice activity detection using mfcc features and support vector machine. In *Int. Conf. on Speech and Computer (SPECOM07), Moscow, Russia*, volume 2, pages 556–561, 2007.
- [121] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.
- [122] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM TASLP*, 2020.
- [123] Taku Kudo and John Richardson. In *Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing*, 2018.
- [124] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. Skill squatting attacks on amazon alexa. In *Proceedings of USENIX Security Symposium*, 2018.
- [125] Andrew Kwong, Wenyuan Xu, and Kevin Fu. Hard drive of hearing: Disks that eavesdrop with a synthesized microphone. In *Proceedings of IEEE S&P*, 2019.
- [126] Gierad Laput, Karan Ahuja, Mayank Goel, and Chris Harrison. Ubioustics: Plug-and-play acoustic activity recognition. In *Proceedings of ACM UIST*, 2018.
- [127] Gierad Laput and Chris Harrison. Sensing fine-grained hand activity with smartwatches. In *Proceedings of ACM CHI*, 2019.
- [128] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R Hershey. Sdr-half-baked or well done? In *Proceedings of IEEE ICASSP*, 2019.
- [129] Ki-Seung Lee. Speech enhancement using ultrasonic doppler sonar. *Speech Communication*, 2019.

- [130] Yeonjoon Lee, Yue Zhao, Jiutian Zeng, Kwangwuk Lee, Nan Zhang, Faysal Hossain Shezan, Yuan Tian, Kai Chen, and XiaoFeng Wang. Using sonar for liveness detection to protect smart speakers against remote attackers. In *Proceedings of ACM IMWUT (UbiComp)*, 2020.
- [131] Yeonjoon Lee, Yue Zhao, Jiutian Zeng, Kwangwuk Lee, Nan Zhang, Faysal Hossain Shezan, Yuan Tian, Kai Chen, and XiaoFeng Wang. Using sonar for liveness detection to protect smart speakers against remote attackers. *Proceedings of ACM IMWUT (UbiComp)*, 2020.
- [132] Chris Lewis and Steve Pickavance. Implementing quality of service over cisco mpls vpns. *Selecting MPLS VPN Services*, 2006.
- [133] Ping Li, Zhenlin An, Lei Yang, and Panlong Yang. Towards physical-layer vibration sensing with rfids. In *Proceedings of IEEE INFOCOM*, 2019.
- [134] Qiang Li, John A Stankovic, Mark A Hanson, Adam T Barth, John Lach, and Gang Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *Proceedings of IEEE International Workshop on Wearable and Implantable Body Sensor Networks*, 2009.
- [135] Yanghao Li, Tushar Nagarajan, Bo Xiong, and Kristen Grauman. Ego-exo: Transferring visual representations from third-person to first-person videos. In *Proceedings of IEEE/CVF CVPR*, 2021.
- [136] Yin Li, Miao Liu, and Jame Rehg. In the eye of the beholder: Gaze and actions in first person video. *IEEE TPAMI*, 2021.
- [137] Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proceedings of ECCV*, 2018.
- [138] Qianru Liao, Yongzhi Huang, Yandao Huang, Yuheng Zhong, Huitong Jin, and Kaishun Wu. Magear: Eavesdropping via audio recovery using magnetic side channel. In *Proceedings of ACM MobiSys*, 2022.
- [139] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. When good becomes evil: Keystroke inference with smartwatch. In *Proceedings of ACM CCS*, 2015.
- [140] Philipos C Loizou. *Speech enhancement: theory and practice*. CRC press, 2013.
- [141] Yan Long, Pirouz Naghavi, Blas Kojusner, Kevin Butler, Sara Rampazzi, and Kevin Fu. Side eye: Characterizing the limits of pov acoustic eavesdropping from smartphone cameras with rolling shutters and movable lenses. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2023.
- [142] Chris Xiaoxuan Lu, Bowen Du, Hongkai Wen, Sen Wang, Andrew Markham, Ivan Martinovic, Yiran Shen, and Niki Trigoni. Snoopy: Sniffing your smartwatch passwords via deep sequence learning. *Proceedings of ACM IMWUT (UbiComp)*, 2018.

- [143] Li Lu, Jiadi Yu, Yingying Chen, Hongbo Liu, Yanmin Zhu, Yunfei Liu, and Minglu Li. Lippass: Lip reading-based user authentication on smartphones leveraging acoustic signals. In *Proceedings of IEEE INFOCOM*, 2018.
- [144] Li Lu, Jiadi Yu, Yingying Chen, and Yan Wang. Vocallock: Sensing vocal tract for passphrase-independent user authentication leveraging acoustic signals on smartphones. *Proceedings of ACM IMWUT (UbiComp)*, 2020.
- [145] Yi Luo and Nima Mesgarani. Tasnet: time-domain audio separation network for real-time, single-channel speech separation. In *Proceedings of IEEE ICASSP*, 2018.
- [146] Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 2019.
- [147] Haojie Ma, Zhijie Zhang, Wenzhong Li, and Sanglu Lu. Unsupervised human activity representation learning with multi-task deep clustering. *Proceedings of ACM IMWUT*, 2021.
- [148] Wenguang Mao, Jian He, and Lili Qiu. Cat: high-precision acoustic motion tracking. In *Proceedings of ACM MobiCom*, 2016.
- [149] Wenguang Mao, Mei Wang, Wei Sun, Lili Qiu, Swadhin Pradhan, and Yi-Chao Chen. Rnn-based room scale hand motion tracking. In *Proceedings of ACM MobiCom*, 2019.
- [150] Wenguang Mao, Zaiwei Zhang, Lili Qiu, Jian He, Yuchen Cui, and Sangki Yun. Indoor follow me drone. In *Proceedings of ACM MobiSys*, 2017.
- [151] Héctor A Cordourier Maruri, Paulo Lopez-Meyer, Jonathan Huang, Willem Marco Beltman, Lama Nachman, and Hong Lu. V-speech: Noise-robust speech capturing glasses using vibration sensors. *Proceedings of ACM IMWUT*, 2018.
- [152] Matthias R Mehl, Simine Vazire, Nairán Ramírez-Esparza, Richard B Slatcher, and James W Pennebaker. Are women really more talkative than men? *Science*, 2007.
- [153] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *Proceedings of USENIX Security Symposium (USENIX Security)*, 2014.
- [154] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. Powerspy: Location tracking using mobile device power analysis. In *Proceedings of USENIX Security*, 2015.
- [155] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 1956.
- [156] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: your finger taps have fingerprints. In *Proceedings of ACM MobiSys*, 2012.

- [157] Kyle Min and Jason J Corso. Integrating human gaze into attention for egocentric activity recognition. In *Proceedings of IEEE WACV*, 2021.
- [158] Kazunori Miura. Ultrasonic directive speaker. *Elektor Magazine*, 2011.
- [159] Brian B Monson, Eric J Hunter, Andrew J Lotto, and Brad H Story. The perceptual significance of high-frequency energy in the human voice. *Frontiers in psychology*, 2014.
- [160] James Morra. Ai chip brings always-on alexa to battery-powered devices. <https://www.electronicdesign.com/technologies/embedded-revolution/article/21808470/ai-chip-brings-alwayson-alexa-to-batterypowered-devices>.
- [161] Arsalan Mosenia, Xiaoliang Dai, Prateek Mittal, and Niraj K Jha. Pinme: Tracking a smartphone user around the world. *IEEE Transactions on Multi-Scale Computing Systems*, 2017.
- [162] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: a large-scale speaker identification dataset. *Proceedings of Interspeech*, 2017.
- [163] Katsuyuki Nakamura, Hiroki Ohashi, and Mitsuhiro Okada. Sensor-augmented egocentric-video captioning with dynamic modal attention. In *Proceedings of ACM MM*, 2021.
- [164] Katsuyuki Nakamura, Serena Yeung, Alexandre Alahi, and Li Fei-Fei. Jointly learning energy expenditures and activities using egocentric multimodal signals. In *Proceedings of IEEE CVPR*, 2017.
- [165] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. Fingerio: Using active sonar for fine-grained finger tracking. In *Proceedings of ACM CHI*, 2016.
- [166] Sashank Narain, Triet D Vo-Huu, Kenneth Block, and Guevara Noubir. Inferring user routes and locations using zero-permission mobile sensors. In *2016 IEEE S&P*, pages 397–413. IEEE, 2016.
- [167] Arun Narayanan and DeLiang Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *Proceedings of IEEE ICASSP*, 2013.
- [168] Ben Nassi, Yaron Pirutin, Tomer Galor, Yuval Elovici, and Boris Zadov. Glowworm attack: Optical tempest sound recovery via a device’s power indicator led. In *Proceedings of ACM CCS*, 2021.
- [169] Ben Nassi, Yaron Pirutin, Adi Shamir, Yuval Elovici, and Boris Zadov. Lamphone: Real-time passive sound recovery from light bulb vibrations. 2022.
- [170] Zhaoheng Ni and Michael I Mandel. Mask-dependent phase estimation for monaural speaker separation. In *Proceedings of IEEE ICASSP*, 2020.
- [171] John Oglesby. What’s in a number? moving beyond the equal error rate. *Speech communication*, 1995.

- [172] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda. Attentive statistics pooling for deep speaker embedding. In *Proceedings of Interspeech*, 2018.
- [173] Xiaomin Ouyang, Xian Shuai, Jiayu Zhou, Ivy Wang Shi, Zhiyuan Xie, Guoliang Xing, and Jianwei Huang. Cosmo: contrastive fusion learning with small data for multimodal human activity recognition. In *Proceedings of ACM MobiCom*, 2022.
- [174] Zhiheng Ouyang, Hongjiang Yu, Wei-Ping Zhu, and Benoit Champagne. A fully convolutional neural network for complex spectrogram processing in speech enhancement. In *Proceedings of IEEE ICASSP*, 2019.
- [175] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of ACM Workshop on Mobile Computing Systems & Applications*, 2012.
- [176] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proceedings of IEEE ICASSP*, 2015.
- [177] Ashutosh Pandey and DeLiang Wang. Tcnn: Temporal convolutional neural network for real-time speech enhancement in the time domain. In *Proceedings of IEEE ICASSP*, 2019.
- [178] Se Rim Park and Jinwon Lee. A fully convolutional neural network for speech enhancement. In *Proceedings of Interspeech*, 2017.
- [179] Santiago Pascual, Antonio Bonafonte, and Joan Serra. Segan: Speech enhancement generative adversarial network. In *Proceedings of IEEE ICASSP*, 2018.
- [180] Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan. Beepbeep: a high accuracy acoustic ranging system using COTS mobile devices. In *Proceedings of ACM SenSys*, 2007.
- [181] Jianwei Qian, Haohua Du, Jiahui Hou, Linlin Chen, Taeho Jung, and Xiang-Yang Li. Hide-behind: Enjoy voice input with voiceprint unclonability and anonymity. In *Proceedings of ACM SenSys*, 2018.
- [182] Jianwei Qian, Haohua Du, Jiahui Hou, Linlin Chen, Taeho Jung, and Xiang-Yang Li. Hide-behind: Enjoy voice input with voiceprint unclonability and anonymity. In *Proceedings of ACM SenSys*, 2018.
- [183] Kun Qian, Chenshu Wu, Fu Xiao, Yue Zheng, Yi Zhang, Zheng Yang, and Yunhao Liu. Acousticcardiogram: Monitoring heartbeats using acoustic signals on smart devices. In *IEEE INFOCOM 2018-IEEE conference on computer communications*, pages 1574–1582. IEEE, 2018.
- [184] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

- [185] Swaminathan Vasanth Rajaraman, Matti Siekkinen, and Mohammad A Hoque. Energy consumption anatomy of live video streaming from a smartphone. In *Proceedings of IEEE PIMRC*, 2014.
- [186] Ashwin Rao, Justine Sherry, Arnaud Legout, Arvind Krishnamurthy, Walid Dabbous, and David Choffnes. Meddle: Middleboxes for increased transparency and control of mobile traffic. 2012.
- [187] ITU-T Recommendation. Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *Rec. ITU-T P. 862*, 2001.
- [188] Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. In *Proceedings of IEEE ICASSP*, 2018.
- [189] Bertrand Rivet, Wenwu Wang, Syed Mohsen Naqvi, and Jonathon A Chambers. Audiovisual speech source separation: An overview of key methodologies. *IEEE Signal Processing Magazine*, 2014.
- [190] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *Proceedings of IEEE ICASSP*, 2001.
- [191] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. Backdoor: Making microphones hear inaudible sounds. In *Proceedings of ACM MobiSys*. ACM, 2017.
- [192] Nirupam Roy and Romit Roy Choudhury. Listening through a vibration motor. In *Proceedings of ACM MobiSys*, 2016.
- [193] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. Inaudible voice commands: The long-range attack and defense. In *Proceedings of Usenix NSDI*, 2018.
- [194] Nirupam Roy, He Wang, and Romit Roy Choudhury. I am a smartphone and i can tell my user's walking direction. In *Proceedings of MobiSys*, 2014.
- [195] Sriram Sami, Yimin Dai, Sean Rui Xiang Tan, Nirupam Roy, and Jun Han. Spying with your robot vacuum cleaner: eavesdropping via lidar sensors. In *Proceedings of ACM SenSys*, 2020.
- [196] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of IEEE CVPR*, 2018.
- [197] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. Nexmon: The c-based firmware patching framework, 2017. <https://nexmon.org>.

- [198] Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. *Proceedings of NAACL*, 2019.
- [199] Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio. Towards end-to-end spoken language understanding. In *Proceedings of IEEE ICASSP*, 2018.
- [200] Cong Shi, Xiangyu Xu, Tianfang Zhang, Payton Walker, Yi Wu, Jian Liu, Nitesh Saxena, Yingying Chen, and Jiadi Yu. Face-mic: inferring live speech and speaker identity via subtle facial dynamics captured by ar/vr motion sensors. In *Proceedings of ACM MobiCom*, 2021.
- [201] Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Actor and observer: Joint modeling of first and third-person videos. In *Proceedings of CVPR*, 2018.
- [202] Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626*, 2018.
- [203] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Proceedings of ECCV*, 2016.
- [204] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. Deep neural network embeddings for text-independent speaker verification. In *Proceedings of Interspeech*, 2017.
- [205] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *Proceedings of IEEE ICASSP*, 2018.
- [206] Meet H Soni, Neil Shah, and Hemant A Patil. Time-frequency masking-based speech enhancement using generative adversarial network. In *Proceedings of IEEE ICASSP*, 2018.
- [207] Gaurav Srivastava, Kunal Bhuwalka, Swarup Kumar Sahoo, Saksham Chitkara, Kevin Ku, Matt Fredrikson, Jason Hong, and Yuvraj Agarwal. Privacyproxy: Leveraging crowdsourcing and in situ traffic analysis to detect and mitigate information leakage, 2017.
- [208] Greg Sterling. Alexa devices maintain 70% market share in u.s. according to survey. <https://marketingland.com/alexa-devices-maintain-70-market-share-in-u-s-according-to-survey-265180>.
- [209] Weigao Su, Daibo Liu, Taiyuan Zhang, and Hongbo Jiang. Towards device independent eavesdropping on telephone conversations with built-in accelerometer. *Proceedings of ACM IMWUT (UbiComp)*, 2022.

- [210] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands: Laser-based audio injection attacks on voice-controllable systems. 2019.
- [211] Ke Sun, Chen Chen, and Xinyu Zhang. “alexa, stop spying on me!” speech privacy protection against voice assistants. In *Proceedings of ACM SenSys*, 2020.
- [212] Ke Sun, Wei Wang, Alex X. Liu, and Haipeng Dai. Depth aware finger tapping on virtual displays. In *Proceedings of ACM MobiSys*, 2018.
- [213] Ke Sun, Chunyu Xia, Songlin Xu, and Xinyu Zhang. Stealthyimu: Stealing permission-protected private information from smartphone voice assistant using zero-permission sensors. Network and Distributed System Security Symposium (NDSS), 2023.
- [214] Ke Sun, Chunyu Xia, Xinyu Zhang, Hao Chen, and Charlie Jianzhong Zhang. Multimodal daily-life logging in free-living environment using non-visual egocentric sensors on a smartphone. *Proceedings of ACM IMWUT (UbiComp)*, 2024.
- [215] Ke Sun and Xinyu Zhang. Ultrase: single-channel speech enhancement using ultrasound. In *Proceedings of ACM MobiCom*, 2021.
- [216] Ke Sun, Ting Zhao, Wei Wang, and Lei Xie. Vskin: Sensing touch gestures on surfaces of mobile devices using acoustic signals. In *Proceedings of ACM MobiCom*, 2018.
- [217] Zehua Sun, Qihong Ke, Hossein Rahmani, Mohammed Bennamoun, Gang Wang, and Jun Liu. Human action recognition from various data modalities: A review. *IEEE TPAMI*, 2022.
- [218] Sanjib Sur, Teng Wei, and Xinyu Zhang. Autodirective Audio Capturing through a Synchronized Smartphone Array. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2014.
- [219] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proceedings of NeurIPS*, 2014.
- [220] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *Proceedings of IEEE ICASSP*, 2010.
- [221] Naoya Takahashi, Sudarsanam Parthasaarathy, Nabarun Goswami, and Yuki Mitsufuji. Recursive speech separation for unknown number of speakers. In *Proceedings of Interspeech*, 2019.
- [222] Jiayao Tan, Cam-Tu Nguyen, and Xiaoliang Wang. Silenttalk: Lip reading through ultrasonic sensing on mobile phones. In *Proceedings of IEEE INFOCOM*, 2017.
- [223] Jiayao Tan, Xiaoliang Wang, Cam-Tu Nguyen, and Yu Shi. Silentkey: A new authentication framework through ultrasonic-based lip reading. *Proceedings of ACM IMWUT (UbiComp)*, 2018.

- [224] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, Soren Brage, Nick Wareham, and Cecilia Mascolo. Selfhar: Improving human activity recognition through self-training with unlabeled data. *Proceedings of ACM IMWUT*, 2021.
- [225] Chuanxin Tang, Chong Luo, Zhiyuan Zhao, Wenxuan Xie, and Wenjun Zeng. Joint time-frequency and time domain learning for speech enhancement. In *Proceedings of IJCAI*, 2020.
- [226] Kristin J Teplansky, Brian Y Tsang, and Jun Wang. Tongue and lip motion patterns in voiced, whispered, and silent vowel production. In *Proceedings of ASSTA ICPHS*.
- [227] H Tijdeman. On the propagation of sound waves in cylindrical tubes. *Journal of Sound and Vibration*, 1975.
- [228] Ingo R Titze and Daniel W Martin. Principles of voice production, 1998.
- [229] Yu-Chih Tung and Kang G Shin. Exploiting sound masking for audio privacy in smartphones. In *Proceedings of ACM AsiaCCS*, 2019.
- [230] Gokhan Tur and Renato De Mori. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.
- [231] Nicolas Turpault, Romain Serizel, Ankit Parag Shah, and Justin Salamon. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. 2019.
- [232] Yunus Emre Ustev, Ozlem Durmaz Incel, and Cem Ersoy. User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal. In *Proceedings of ACM UbiComp*, 2013.
- [233] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *9th USENIX Workshop on Offensive Technologies*, 2015.
- [234] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NeurIPS*, 2017.
- [235] Nishant Verma. *Mobile Test Automation With Appium*. Packt Publishing Ltd, 2017.
- [236] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. In *Proceedings of AAAI*, 2016.
- [237] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 2006.

- [238] Chao Wang, Feng Lin, Zhongjie Ba, Fan Zhang, Wenyao Xu, and Kui Ren. Wavesdropper: Through-wall word detection of human speech via commercial mmwave devices. *Proceedings of ACM IMWUT*, 2022.
- [239] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. Friend or foe? your wearable devices reveal your personal pin. In *Proceedings of ACM AsiaCCS*, 2016.
- [240] Chuyu Wang, Lei Xie, Yuancan Lin, Wei Wang, Yingying Chen, Yanling Bu, Kai Zhang, and Sanglu Lu. Thru-the-wall eavesdropping on loudspeakers via rfid by capturing sub-mm level vibration. *Proceedings of ACM IMWUT (UbiComp)*, 2020.
- [241] DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2018.
- [242] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *Proceedings of ACM MobiCom*, 2015.
- [243] Lei Wang, Wei Li, Ke Sun, Fusang Zhang, Tao Gu, Chenren Xu, and Daqing Zhang. Loear: Push the range limit of acoustic sensing for vital sign monitoring. *Proceedings of ACM IMWUT (UbiComp)*, 2022.
- [244] Quan Wang, Hannah Muckenhirn, Kevin Wilson, Prashant Sridhar, Zelin Wu, John Hershey, Rif A Saurous, Ron J Weiss, Ye Jia, and Ignacio Lopez Moreno. Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking. In *Proceedings of Interspeech*, 2019.
- [245] Shiyang Wang, Xingchen Wang, Wenjun Jiang, Chenglin Miao, Qiming Cao, Haoyu Wang, Ke Sun, Hongfei Xue, and Lu Su. Towards smartphone-based 3d hand pose reconstruction using acoustic signals. *ACM Transactions on Sensor Networks*, 2024.
- [246] Siqi Wang, Anuj Pathania, and Tulika Mitra. Neural network inference on mobile socs. *IEEE Design & Test*, 2020.
- [247] Wei Wang, Alex X Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. Understanding and modeling of wifi signal based human activity recognition. In *Proceedings of ACM MobiCom*, 2015.
- [248] Wei Wang, Alex X. Liu, and Ke Sun. Device-free gesture tracking using acoustic signals. In *Proceedings of ACM MobiCom*, pages 82–94, 2016.
- [249] Wei Wang, Alex X. Liu, and Ke Sun. Device-free gesture tracking using acoustic signals. In *Proceedings of ACM MobiCom*, 2016.
- [250] Xun Wang, Ke Sun, Ting Zhao, Wei Wang, and Qing Gu. Dynamic speed warping: Similarity-based one-shot learning for device-free gesture signals. In *Proceedings of IEEE INFOCOM*, 2020.

- [251] Yan Wang, Jian Liu, Yingying Chen, Marco Gruteser, Jie Yang, and Hongbo Liu. E-eyes: device-free location-oriented activity identification using fine-grained wifi signatures. In *Proceedings of ACM MobiCom*, 2014.
- [252] Yan Wang, Jian Liu, Yingying Chen, Marco Gruteser, Jie Yang, and Hongbo Liu. E-eyes: In-home device-free activity identification using fine-grained WiFi signatures. In *Proceedings of ACM MobiCom*, 2014.
- [253] Yuxuan Wang, Arun Narayanan, and DeLiang Wang. On training targets for supervised speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 2014.
- [254] Ziqi Wang, Zhe Chen, Akash Deep Singh, Luis Garcia, Jun Luo, and Mani B Srivastava. Uwhear: through-wall extraction and separation of audio vibrations using wireless signals. In *Proceedings of ACM SenSys*, 2020.
- [255] Teng Wei, Shu Wang, Anfu Zhou, and Xinyu Zhang. Acoustic eavesdropping through wireless vibrometry. In *Proceedings of ACM MobiCom*, 2015.
- [256] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 1993.
- [257] Gordon Wichern, Joe Antognini, Michael Flynn, Licheng Richard Zhu, Emmett McQuinn, Dwight Crow, Ethan Manilow, and Jonathan Le Roux. WHAM!: Extending Speech Separation to Noisy Environments. *CoRR*, abs/1907.01160, 2019.
- [258] J Widder and A Morcelli. Basic principles of mems microphones, 2016. <https://www.edn.com/basic-principles-of-mems-microphones/>.
- [259] Simon Wiesler, Alexander Richard, Ralf Schlüter, and Hermann Ney. Mean-normalized stochastic gradient for large-scale deep learning. In *Proceedings of IEEE ICASSP*, 2014.
- [260] Robert Williams. Study: Smart speaker ownership surges 36% to 53m US adults. <https://www.mobilemarketer.com/news/study-smart-speaker-ownership-surges-36-to-53m-us-adults/545717/>.
- [261] Donald S Williamson, Yuxuan Wang, and DeLiang Wang. Complex ratio masking for monaural speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 2015.
- [262] Donald S Williamson, Yuxuan Wang, and DeLiang Wang. Complex ratio masking for monaural speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 2015.
- [263] Zack Wittaker. Amazon says US government demands for customer data went up. <https://techcrunch.com/2019/08/01/amazon-prism-transparency-data/>.

- [264] Dan Wu, Daqing Zhang, Chenren Xu, Hao Wang, and Xiang Li. Device-free wifi human sensing: From pattern-based to model-based approaches. *IEEE Communications Magazine*, 2017.
- [265] Jason Wu, Chris Harrison, Jeffrey P Bigham, and Gierad Laput. Automated class discovery and one-shot interactions for acoustic activity recognition. In *Proceedings of ACM CHI*, 2020.
- [266] Yuan-Kuei Wu, Chao-I Tuan, Hung-yi Lee, and Yu Tsao. Saddle: Joint speech separation and denoising model based on multitask learning. *arXiv:2005.09966*, 2020.
- [267] A. D. Wyner. The wire-tap channel. *The Bell System Technical Journal*, 54(8):1355–1387, Oct 1975.
- [268] Chenhan Xu, Zhengxiong Li, Hanbin Zhang, Aditya Singh Rathore, Huining Li, Chen Song, Kun Wang, and Wenyao Xu. Waveear: Exploring a mmwave-based noise-resistant speech sensing for voice-user interface. In *Proceedings of ACM MobiSys*, 2019.
- [269] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications. In *Proceedings of ACM SenSys*, 2021.
- [270] Linfeng Xu, Qingbo Wu, Lili Pan, Fanman Meng, Hongliang Li, Chiyuan He, Hanxin Wang, Shaoxu Cheng, and Yu Dai. Towards continual egocentric activity recognition: A multi-modal egocentric activity dataset for continual learning. *arXiv preprint arXiv:2301.10931*, 2023.
- [271] Panlong Yang, Yuanhao Feng, Jie Xiong, Ziyang Chen, and Xiang-Yang Li. Rf-ear: Contactless multi-device vibration sensing and identification using cots rfid. In *Proceedings of IEEE INFOCOM*, 2020.
- [272] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of WWW*, 2017.
- [273] Shuochao Yao, Ailing Piao, Wenjun Jiang, Yiran Zhao, Huajie Shao, Shengzhong Liu, Dongxin Liu, Jinyang Li, Tianshi Wang, Shaohan Hu, et al. Stfnets: Learning sensing signals from the time-frequency perspective with short-time fourier neural networks. In *Proceedings of WWW*, 2019.
- [274] Dacheng Yin, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. Phasen: A phase-and-harmonics-aware speech enhancement network. In *Proceedings of AAAI*, 2020.
- [275] Soo Youn. Alexa is always listening — and so are amazon workers. <https://abcnews.go.com/Technology/alex-listening-amazon-workers/story?id=62331191>.

- [276] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *Proceedings of IEEE ICASSP*, 2017.
- [277] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, XiaoFeng Wang, and Carl A Gunter. Commandersong: A systematic approach for practical adversarial voice recognition. In *Proceedings of USENIX Security Symposium*, 2018.
- [278] Sangki Yun, Yi-Chao Chen, Huihuang Zheng, Lili Qiu, and Wenguang Mao. Strata: Fine-grained acoustic-based device-free tracking. In *Proceedings of ACM MobiSys*, 2017.
- [279] Muhammed Zahid Ozturk, Chenshu Wu, Beibei Wang, and KJ Liu. Radiomic: Sound sensing via mmwave signals. *arXiv e-prints*, 2021.
- [280] Mohammad Zeineldeen, Albert Zeyer, Wei Zhou, Thomas Ng, Ralf Schlüter, and Hermann Ney. A systematic comparison of grapheme-based vs. phoneme-based label units for encoder-decoder-attention models. *arXiv:2005.09336*, 2020.
- [281] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of ACM CCS*, 2017.
- [282] Li Zhang, Parth H Pathak, Muchen Wu, Yixin Zhao, and Prasant Mohapatra. Accelword: Energy efficient hotword detection through accelerometer. In *Proceedings of ACM MobiSys*, 2015.
- [283] Linghan Zhang, Sheng Tan, and Jie Yang. Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication. In *Proceedings of ACM CCS*, 2017.
- [284] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *Proceedings of IEEE Security and Privacy*, 2019.
- [285] Xiyuan Zhang, Ranak Roy Chowdhury, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. Modeling label semantics improves activity recognition. *arXiv preprint arXiv:2301.03462*, 2023.
- [286] Yang Zhang, Gierad Laput, and Chris Harrison. Vibrosight: Long-range vibrometry for smart environment sensing. In *Proceedings of ACM UIST*, pages 225–236, 2018.
- [287] Bing Zhou, Jay Lohokare, Ruipeng Gao, and Fan Ye. Echoprint: Two-factor authentication using acoustics and vision on smartphones. In *Proceedings of ACM MobiCom*, 2018.
- [288] Yinian Zhou, Awais Ahmad Siddiqi, Jia Zhang, Junchen Guo, Rui Xi, Meng Jin, Zhengang Zhai, and Yuan He. Voice recovery from human surroundings with millimeter wave radar. In *Proceedings of INFOCOM Workshops*, 2021.

- [289] Shilin Zhu, Chi Zhang, and Xinyu Zhang. Automating Visual Privacy Protection Using a Smart LED. In *Proceedings of ACM MobiCom*, 2017.