Scaling Machine Learning Systems Using Domain Adaptation

Akhil Mathur

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Computer Science
University College London

December 10, 2020

I, Akhil Mathur, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Machine-learned components, particularly those trained using deep learning methods, are becoming integral parts of modern intelligent systems, with applications including computer vision, speech processing, natural language processing and human activity recognition. As these machine learning (ML) systems scale to real-world settings, they will encounter scenarios where the distribution of the data in the real-world (i.e., the target domain) is different from the data on which they were trained (i.e., the source domain). This phenomenon, known as domain shift, can significantly degrade the performance of ML systems in new deployment scenarios.

In this thesis, we study the impact of domain shift caused by variations in system hardware, software and user preferences on the performance of ML systems. After quantifying the performance degradation of ML models in target domains due to the various types of domain shift, we propose unsupervised domain adaptation (uDA) algorithms that leverage *unlabeled data* collected in the target domain to improve the performance of the ML model. At its core, this thesis argues for the need to develop uDA solutions while adhering to practical scenarios in which ML systems will scale. More specifically, we consider four scenarios: (i) *opaque ML systems*, wherein parameters of the source prediction model are not made accessible in the target domain, (ii) *transparent ML systems*, wherein source model parameters are accessible and can be modified in the target domain, (iii) ML systems where source and target domains do not have identical label spaces, and (iv) *distributed ML systems*, wherein the source and target domains are geographically distributed, their datasets are private and cannot be exchanged using adaptation. We study the unique challenges and constraints of each scenario and propose novel uDA algorithms that outperform state-of-the-art baselines.

Impact Statement

In recent years, we have witnessed the exciting capabilities and opportunities that artificial intelligence (AI) can bring for end-users. For example, the Seeing AI app from Microsoft assists users with visual impairments in identifying people, text, or objects near them; the AI-based biometrics platform by Element Inc. aims to provide a digital identity to infants and children in the Global South. While such technologies are undoubtedly promising, it is important to ensure that they work-universally-for users around the world and not just be limited to a subset of the population in a specific part of the world, otherwise we risk creating a world of AI haves and have-nots, i.e., people who will gain advanced capabilities in their lives with AI and others who will not.

One of the hurdles in fulfilling this vision is the reliance of today's state-of-the-art AI algorithms, such as those based on deep learning methods, on large-scale labeled datasets. Moreover, a fundamental assumption behind these solutions is that the distribution (or characteristic) of the data will remain the same between training and deployment stages. Naturally, as AI technologies become more widespread, such divergences between training and test data are more likely to occur. For example, users in different parts of the world will have diversity in the devices they use (ranging from low-end phones to very expensive smartphones), their environment conditions (e.g., ambient noise levels, natural lighting), and their technology usage patterns—all of which are likely to introduce unique variability in the user data, thus making it different from the original training data.

This thesis contributes to addressing these challenges by proposing algorithms to *adapt* ML systems to new deployment settings, only using unlabeled data collected therein. In doing so, we place a special emphasis on ensuring that our proposed algorithms are grounded in the realistic scenarios in which ML systems will be deployed. As an example, ML systems often encounter private and sensitive user data (e.g., biometrics, health records) during deployment which cannot be shared

with other parties; hence, one of our proposed adaptation algorithms is designed to work without requiring access to raw data during deployment.

During this research, we collected two large-scale multi-microphone speech datasets which have been made publicly available to the academic community to facilitate research and development of robust speech systems. Finally, the algorithms developed in this thesis are being deployed in the industry to adapt AI systems operating on mobile and embedded devices to new deployment settings.

Acknowledgements

I want to thank my wonderful advisors, Nic Lane and Nadia Berthouze, for their support and motivation throughout my Ph.D. journey. I am deeply grateful to both of you for the countless hours of discussions on research problems, paper writing, and career directions. I am inspired by your passion for research, and the kindness and care you show towards the students is something I wish to emulate. Thank you!

The decision to pursue a Ph.D. alongside a full-time job was not easy for me. Looking back at it, I am glad I took this path. A lot of credit goes to Fahim Kawsar, who encouraged me to pursue the Ph.D. and has been a constant source of support and inspiration in my research career. *Dhonnyobaad*!

I consider myself lucky to have been mentored by some fantastic people in my career. My deepest gratitude to Neha Kumar for her unwavering support, positivity, and kindness over the years. I am also thankful to Matthew Kam, Ravin Balakrishnan, Sharad Jaiswal, Vishy Poosala, Bo Olofsson, and Markus Hofmann for their roles in shaping my research career.

I have been extremely fortunate to work with many outstanding people over the last few years. I thank my friends and colleagues at the University College London (Chongyang Wang, Tao Bi, Ahmed Alqaraawi, Temitayo Olugbade, Robert Smith) and CaMLSys (Daniel J. Beutel, Taner Topal, Titouan Parcollet, Xinchi Qiu) for the fantastic discussions and memories.

I am also blessed to have some exceptional colleagues and friends at Nokia Bell Labs over the years (Alessandro Montanari, Christina L-De-Menezes-Dietschi, Chulhong Min, Alberto Gil Ramos, Marc Van Den Broeck, Mo Alloulah, Shaoduo Gan, Sourav Bhattacharya, Tianlin Zhang, Utku Acer, Youngjae Chang), and thank them for all the research collaborations and adventures around Cambridge.

I would like to give a special shout-out to Anton Isopoussu for being a great collaborator and friend in this journey. I have learned a lot working with you and hope to

collaborate again in the future.

I am lucky to have a large and loving circle of family and friends back home in India. I thank you all for your love and awesomeness. A massive thanks to my in-laws (Ayush, Dishant, Mr. K.N. Mathur and Ms. Shubha Mathur) for your support and understanding.

I dedicate this thesis to the four pillars of my life: Anjali, Meenu, Daddy, and Mummy. I am fortunate to have you in my life. Without your love, affection and constant support, this thesis would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The only way that we can live, is if we grow. The only way that we can grow is if we change. The only way that we can change is if we learn. The only way we can learn is if we are exposed. And the only way that we can become exposed is if we throw ourselves out into the open. Do it. Throw yourself.

C. JoyBell C.

The field of machine learning has witnessed significant breakthroughs in the last decade driven by the availability of large-scale data, advancements in computational models that process this data to generate meaningful inferences, and the massive increase in computational power available to train these models. As a result, machine-learned components are increasingly being incorporated in the design of intelligent systems in various domains such as medical imaging [3] autonomous cars [4], conversational agents [5], smart buildings [6] and even smartphones and wearable devices [7, 8, 9]. As an example, consider the Automatic Brightness subsystem in the latest Android smartphones. Until recently, this sub-system used a rule-based classifier to adjust the screen brightness based on the ambient illuminance captured from the light sensor on the phone. However, system designers realized that this one-size-fits-all approach does not work for all users, and replaced the rule-based classifier with one trained using machine learning techniques that can adjust the screen brightness by learning a user's unique preferences [10]. On similar lines, Mehrotra et al. [9] have proposed intelligent notification systems for smartphones, which leverage machine learning techniques to learn a user's notification receptivity behavior and personalize the delivery of notifications.

As machine learning systems become pervasive, they will be deployed in diverse real-world scenarios, many of which the system designer might not have even considered at the time of training. For example, speech-based conversational systems

will have to interact with users with different accents; object recognition systems might be deployed in lighting conditions drastically different from the ones in which they were trained. In such diverse scenarios, the data collected during deployment (i.e., test data) would look quite different from the data on which the model was trained (i.e., training data), which in turn could degrade the performance of the machine learning system [11]. A good example of this practical challenge is the recently released ObjectNet dataset [12], which contains images of objects captured with different backgrounds, rotations, and viewing angles. When state-of-the-art object detection models trained on the widely used ImageNet [13] dataset were tested on ObjectNet, they suffered a dramatic drop of 40-45% in both top-1 and top-5 class recognition accuracy.

Hence, in order for machine learning systems to scale in the real-world successfully, it is essential that they are capable of evolving to new scenarios with minimal human supervision. Towards this goal, this thesis proposes a number of algorithms and system components for adapting machine learning systems to new domains, just using unlabeled data collected in them.

In the next section, we contextualize the contributions of this thesis in the big picture of scalability challenges for machine learning systems.

1.1 Challenges in Scaling Machine Learning Systems

The problem of designing scalable machine learning (ML) systems is multi-faceted. This section discusses some prominent challenges in scaling ML systems and highlights how the work in this thesis fits into the research landscape.

1.1.1 Overview

ML models are often trained on large-scale datasets to achieve state-of-the-art prediction performance. Hence, it is important to design *scalable and efficient training pipelines* to accelerate the training process. In this direction, substantial research has been done on developing hardware accelerators [14], distributing the training across multiple machines [15, 16, 17, 18, 19, 20], and even on novel data pipelines [21]. In many scenarios, training data is distributed geographically across multiple nodes and cannot be aggregated on a central server due to privacy constraints; *federated learning* [22, 23, 24] algorithms tackle this challenge by a combination of local training of ML models on the individual nodes and aggregating the parameter updates on a central server.

Once a model is trained, it has to be made available to downstream system components or end-user applications. ML models are often hosted on cloud infrastructure such as Amazon Elastic Compute Cloud or Google Cloud Platform, and offered as a service to user applications. In this case, it becomes important to design *scalable serving solutions* that can fulfill thousands of concurrent inference requests from the users. Model serving frameworks such as TensorFlow Serving [25], TorchServe [26], and MLflow [27] are some of the promising examples in this space. Further, for ML systems that are deployed on the edge, it is crucial that they can scale to a wide variety of devices with different resource constraints on memory and computation capabilities; in this direction, there have been promising efforts in the area of embedded deep learning in the last few years [28, 29, 30, 31].

Prior research has also highlighted the significant challenges in *engineering large-scale ML systems* [32, 33] including the management of model configurations and hyperparameters, keeping the data processing and feature extraction pipelines clean, dealing with hidden feedback loops and mitigating unexpected dependencies between different system components.

Finally, the issue of *data heterogeneity* in real-world scenarios is a major challenge to the scalability of ML systems. ML models trained using supervised learning techniques assume that at inference time, test data will be drawn from the same distribution as the training data. However, in practical settings, a number of factors could cause the test data distribution to diverge from the training distribution, which in turn could lead to significant performance degradation of ML models.

Dealing with heterogeneity or drift in the data between training and test stages requires a two-step solution. Firstly, we need to identify that data drift is indeed happening in the ML system. In this direction, there have been numerous interesting works that have highlighted the need for continuous monitoring of ML systems in deployment [34, 33, 35] both at the level of input data and model predictions. From an algorithm viewpoint, Rabanser et al. [36] proposed various techniques to detect dataset drift through a combination of dimensionality reduction and a two-sample-testing approach. There is also literature on anomaly detection [37] using techniques such as one-class Support Vector Machine [38] and random cut forest [39] that can be used to identify anomalous data samples during deployment stages. Other strategies proposed for detecting out-of-distribution (OOD) samples include applying a threshold on the softmax probabilities of the pre-trained model [40], pre-processing the training data with adversarial perturbations [41], or explicitly exposing the model to an outlier dataset [42] during training.

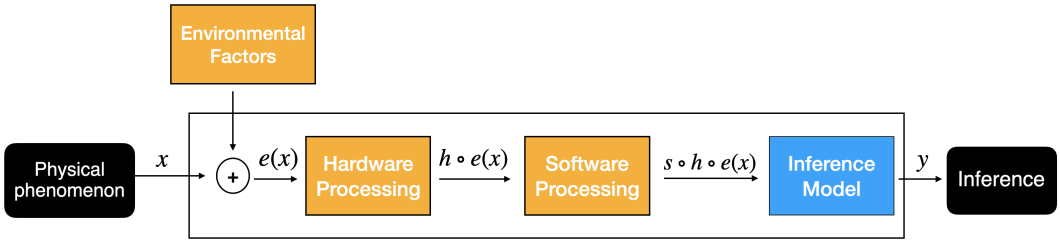


Figure 1.1: Generic pipeline for sensor data processing and inference.

Once a drift in the data is detected, the second step is to fix it to ensure that it does not degrade an ML system's performance during deployment. The work in this thesis aims to address this particular challenge in scaling ML systems. We consider different scenarios of deployment of ML systems, such as *opaque ML systems* (wherein the model parameters are not accessible and cannot be modified during deployment), *transparent ML systems* (wherein the model parameters can be modified during deployment), and *distributed ML systems*. Further, our investigations focus on scaling ML systems that operate on sensor data collected from microphones and inertial sensors, and encounter sensor-induced heterogeneities in the data.

1.1.2 Sensing Heterogeneities

We now give a brief overview of the causes of heterogeneities in sensor data and how this thesis aims to address them. Later in Chapter 2, we elaborate on these heterogeneities in the context of ML systems studied in this thesis.

There are a plethora of factors that can cause heterogeneity in sensor data in the real-world. Figure 1.1 shows a generic pipeline for collecting and processing sensor data. As we can see, before the sensor data reaches the inference model, it undergoes processing by a number of components, each of which could potentially lead to heterogeneity in the data. A physical analog signal $\mathbf{x}$ first encounters various environmental factors that can be a source of heterogeneity in the data. For example, perturbations in audio and image data can be respectively introduced by acoustic noise or lighting conditions in the environment. The perturbed signal denoted as $e(\mathbf{x})$ is then processed by the sensor hardware which converts it into a digital output $h \circ e(\mathbf{x})$. Thereafter, it is processed by domain-specific sampling and signal processing algorithms to obtain $s \circ h \circ e(\mathbf{x})$, before being passed to the ML inference model.

Prior research has demonstrated that each of the above components in the sensing pipeline can induce variability in the sensor data. For example, [43] highlighted that imperfections introduced in the analog circuitry of accelerometer sensors during the

manufacturing process could cause variations in their digital output. Interestingly, they found that these hardware-induced variations in accelerometer data are computationally so significant that they can be leveraged to fingerprint a smart device and identify it with over 99% accuracy in a pool of 100 devices. On similar lines, in a study across 52 smartphones, [44] showed that microphone sensors of these devices could introduce variability in the audio data, which are also reflected in the Mel-Frequency Cepstral Coefficient (MFCC) features and could be used to fingerprint the smartphone with an accuracy of 97%.

Moreover, the software stack on sensor devices can also be the source of heterogeneity in the data. For example, Stisen et al. [45] highlighted that run-time factors on smartphones, such as instantaneous I/O load or delays in timestamp attachment to sensor measurements by the operating system could lead to unstable sampling rates for accelerometer and gyroscope sensors. Similarly, manufacturers of multi-channel microphone arrays [46, 47] employ different (and often proprietary) noise reduction and beamforming algorithms to enhance the signal quality before it is passed onto the user applications. Prior research [48, 49] has shown that these algorithms are also sensitive to microphone parameters, such as gain mismatches between individual microphones and could add unintended biases in the audio data that is passed to the user applications. For example, through experiments on multi-channel speech recordings, [49] found that acoustic beamforming algorithms may even cause the performance of speech recognition models to degrade if channel parameters across microphones differ. Besides the hardware and software causes, variability in sensor data can also come from differences in sensing environments, user demographics, and user preferences regarding how they interact with the system.

As we describe in the next chapter, there have been research initiatives to individually address each of these sources of variability in sensor data, for example, by developing sensor calibration solutions [50, 51] or adaptive beamforming algorithms [49]. While such efforts are important, we argue that instead of developing such component-level solutions to counter individual sources of heterogeneity in sensor data, we can address the *composite effect of these heterogeneities* directly inside the training and inference pipelines of machine learning models. This viewpoint is essential because ML components are often treated as end-consumers of sensor data in real-world systems. They may not have low-level access to system components such as the sensor hardware or the signal processing algorithms used by the system designer. As such, it will be infeasible to apply any component-level solutions designed to address the individual sources of variability in the data.

1.1.3 Sensing Heterogeneities as Domain Shift

The combined effect of different types of sensing heterogeneities can be interpreted as a form of *domain shift* in the sensor data. As defined by Storkey [52, 11], domain shift is characterized by a change in the measurement system of data. Assuming that there is an underlying unchanging latent representation space in which the sensor sample $\mathbf{x}$ lies, we would ideally like to learn a relationship between $\mathbf{x}$ and an output class $\mathbf{y}$. However as shown in Figure 1.1, in practical ML systems instead of $\mathbf{x}$ we only observe $\mathbf{x}' = f(\mathbf{x}) = s \circ h \circ e(\mathbf{x})$, where f is a composite function that represents the accumulated effect of hardware, software and environment-related factors on sensor data.

More importantly, variations in the hardware, software pipelines, or the operating environment in real-world systems could cause the sensor data $\mathbf{x}'$ received by an ML model to change between training and deployment stages, even if the underlying physical phenomenon $\mathbf{x}$ is the same. This would, in effect, cause the distributions of training and deployment data to differ from each other, thereby invalidating a fundamental assumption of supervised learning algorithms that are used to train sensor-based ML models.

Towards the goal of scaling ML systems in the presence of sensing heterogeneities, we propose algorithms and system components built on the principle of domain adaptation that minimize the effect of domain shift in ML systems. More specifically, as collecting labeled data during deployment could be expensive or infeasible, we focus on unsupervised domain adaptation solutions that can adapt ML systems to new deployment scenarios only using unlabeled data from them.

1.2 Research Questions and Contributions

We now formulate four research questions that we seek to answer in this thesis. Each research question is accompanied by a conceptual diagram that summarizes our contribution. In the rest of the thesis, we use the term *source domain* to refer to the training stage of an ML model, and *target domain* to refer to its deployment stage.

RQ1. *What is the effect of sensing heterogeneities on the performance of state-of-the-art deep learning models operating on speech and inertial sensing data?*

In Chapter 3 (and also published in [53, 54, 55, 56]), we quantify the effect of two types of domain shift on sensor-based ML models. Firstly, to facilitate the

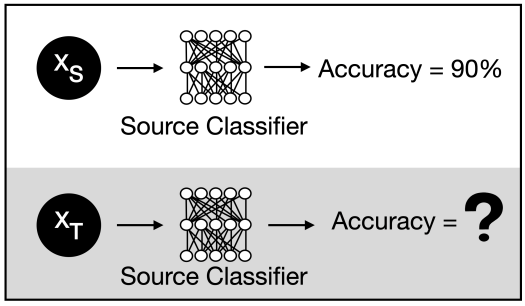


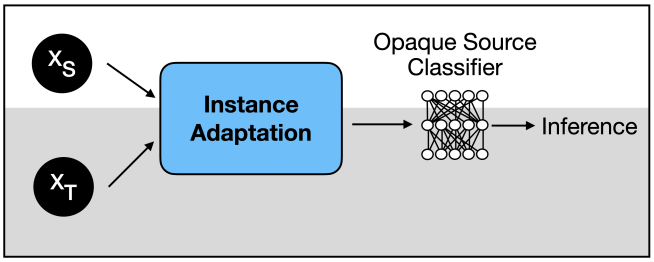
Figure 1.2: In Chapter 3, we quantify the effect of various domain shifts on sensor inference models. The parts with grey and white backgrounds represent the target and source domains respectively. X_S and X_T represent source and target datasets.

study of microphone-induced domain shift in deep learning-based speech models, we introduce a data collection methodology to collect large-scale speech datasets simultaneously from multiple microphones. Based on the proposed methodology, we collect two new speech datasets that contain domain shifts induced by hardware, software, and environmental factors. We then evaluate the combined effect of these domain shifts on the performance of Spoken Keyword Detection (SKD) and Automatic Speech Recognition (ASR) models. Secondly, in the context of human activity recognition (HAR) models, we study the domain shift in inertial sensor data caused by variations in sensor hardware, software, and user preferences. Our results demonstrate that domain shift induced by the above heterogeneities poses a major challenge to the scalability of speech and HAR models in real-world scenarios.

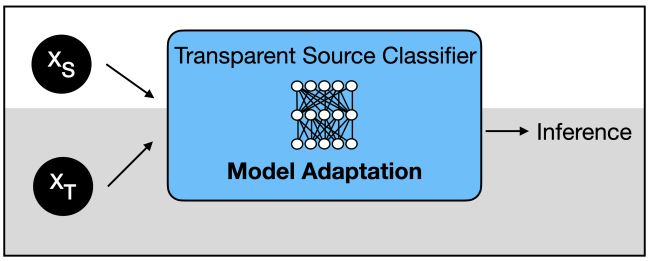
RQ2. *What types of algorithms can be developed to counter domain shift in sensor inference systems while adhering to practical system realities?*

Having quantified the impact of domain shift on various sensor inference models, we propose two solutions to counter them based on the principles of **unsupervised domain adaptation (uDA)**. Grounded in practical realities of ML systems, the solutions are as follows:

Solution 1. In Chapter 4 (and also published in [54]), we consider speech-based **opaque ML systems** where the parameters of the prediction model trained in the source domain are not accessible during the adaptation process. We propose an adaptation solution that operates on input data instances and learns a translation function between data from the source and target domains, using the principles of Cyclic Generative Adversarial Networks (CycleGANs). Once learned, this translation function could be applied as a pre-processing component in the speech inference pipeline to translate data samples from the target domain to the source domain,



(a) In Chapter 4, we propose an instance adaptation solution for opaque ML systems.



(b) In Chapter 5, we propose a model adaptation solution for transparent ML systems.

Figure 1.3: uDA Solutions are proposed in Chapters 4 and 5 to counter domain shift in speech and inertial data.

thereby reducing the domain shift and enhancing the speech inference model's accuracy in the target domain.

Solution 2. In Chapter 5 (and also published in [56]), we consider the case of **transparent ML systems** wherein model parameters are accessible and can be modified during the adaptation process. In the context of sensor-placement induced domain shifts in human activity recognition (HAR) systems, we propose an adversarial feature alignment algorithm that takes as input a neural network model trained on a labeled source domain, and adapts its parameters for use in a new unlabeled target domain. We show that this technique can outperform a number of baseline adaptation methods and improve the accuracy of HAR systems when they are deployed in new target domains.

RQ3. *How do we extend unsupervised domain adaptation algorithms to ML systems where the source and target domains do not share a common label space?*

Through the research conducted in Chapter 5, we uncovered a number of assumptions related to the source and target distributions that can impact the performance of uDA algorithms. In Chapter 6 (and also published in [57]), we focus on relaxing

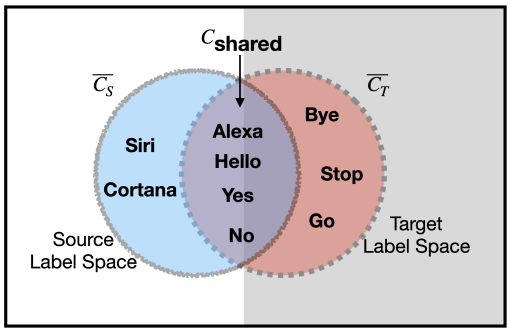


Figure 1.4: In Chapter 6, we extend uDA to scenarios of mismatched source and target label spaces. $C_{\text{shared}}, \overline{C_S}$ and $\overline{C_T}$ denote the shared and private label spaces in the source and target domains.

one key assumption made in uDA methods that the label spaces of source and target domains must be identical. This assumption could be easily violated in real-world scenarios due to the presence of outlier or private classes in either the source or the target domain. We first quantify the adverse effect of such private classes on the adaptation of speech classifiers, and then propose an end-to-end unsupervised adaptation solution to mitigate this problem.

RQ4. *What are the challenges in deploying uDA algorithms in distributed ML systems with multiple target domains?*

Our investigations on uDA algorithms in the previous chapters also uncovered that these algorithms are not designed to work in distributed ML systems, where source and target domains are geographically separated, and exchanging data between them could be expensive or may have privacy implications. Hence, in Chapter 7 and [58], we carefully study the bottlenecks of uDA algorithms related to privacy and convergence time in distributed settings, and present a distributed adaptation solution to address them.

Moreover, we show that as ML systems scale to multiple target domains, it becomes important to allow target domains to flexibly choose their adaptation collaborators (i.e., the source domain with which uDA is performed). In this direction, we present an Optimal Collaborator Selection algorithm, which could be used to find a collaborator for each target domain from a set of available candidates. Finally, we propose an end-to-end framework called *Framework for Realistic uDA (FRUDA)* that combines the collaborator selection algorithm and the distributed uDA training approach. We demonstrate how FRUDA can help scale various domain adaptation algorithms to distributed settings.

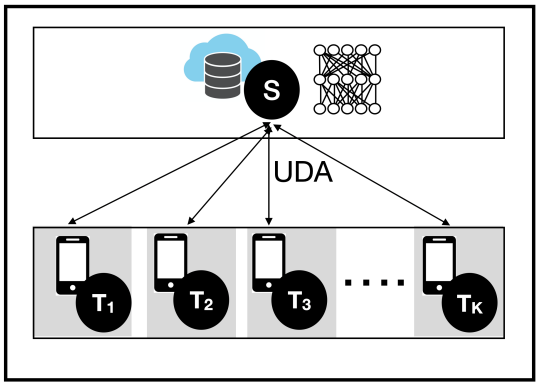


Figure 1.5: In Chapter 7, we extend uDA to distributed ML settings with multiple target domains.

1.3 Thesis Structure

The thesis is organised as follows. Chapter 2 provides background on various types of sensory inference tasks studied in this dissertation, along with a detailed discussion on unsupervised domain adaptation algorithms proposed in the literature. In Chapter 3, we study RQ1 and quantify the effect of various domain shifts in speech and inertial recognition models. Chapter 4 and Chapter 5 contribute to answering RQ2 by proposing instance-based and model-based domain adaptation approaches for opaque and transparent ML models respectively. Chapter 6 is devoted to answer RQ3 and offers solutions for the problem of label space mismatch which can cause accuracy degradation in domain adaptation. Chapter 7 then studies RQ4 and extends the previous adaptation algorithms to distributed settings. Finally, Chapter 8 concludes this thesis, outlines its limitations and highlights avenues for future work.

1.4 Research Outcomes

In this subsection, we list the publications and dataset artefacts that support the text in this thesis.

1.4.1 Peer-Reviewed Publications

Below are the peer-reviewed publications that came out of the work done in this thesis.

Conference and Journal Papers (* denotes equal contribution)

- **Mathur A.**, Berthouze N., Lane N.D. *Unsupervised Domain Adaptation Under Label Space Mismatch for Speech Classification*. Proceedings of Interspeech 2020.

- **Mathur A.**, Kawsar F., Berthouze N., Lane N.D. *LIBRI-ADAPT: A new speech dataset for unsupervised domain adaptation*. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020.
- Chang Y*., **Mathur A.***, Isopoussu A., Song J., Kawsar F. *A Systematic Study of Unsupervised Domain Adaptation for Robust Human-Activity Recognition*. Proceedings of the ACM on Interactive, Mobile,Wearable and Ubiquitous Technologies (IMWUT), 2020.
- **Mathur A.**, Isopoussu A., Kawsar F., Berthouze N., Lane N.D. *FlexAdapt: Flexible Cycle-Consistent Adversarial Domain Adaptation*. Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019
- **Mathur A.**, Isopoussu A., Kawsar F., Berthouze N., Lane N.D. *Mic2Mic: Using Cycle-Consistent Generative Adversarial Networks to Overcome Microphone Variability in Speech Systems*. Proceedings of the 18th ACM International Conference on Information Processing in Sensor Networks (IPSN), 2019.
- **Mathur A.**, Gan S., Isopoussu A., Kawsar F., Berthouze N., Lane N.D. *Scaling Unsupervised Domain Adaptation through Optimal Collaborator Selection and Lazy Discriminator Synchronization*. arXiv preprint.

Workshop Papers

- Gan S.*, **Mathur A.***, Isopoussu A., Berthouze N., Lane N.D., Kawsar F. *Distributed Asynchronous Domain Adaptation: Towards Making Domain Adaptation More Practical in Real-World Systems*. Workshop on Systems for ML at Thirty-third Annual Conference on Neural Information Processing Systems (NeurIPS), 2019.
- **Mathur A.**, Isopoussu A., Kawsar F., Smith R., Lane N.D. and Berthouze N. *On Robustness of Cloud Speech APIs: An Early Characterization*. Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing and Wearable Computers (UbiComp), 2018.
- **Mathur A.**, Isopoussu A., Berthouze N., Lane N.D., Kawsar F. *Unsupervised Domain Adaptation for Robust Sensory Systems*. Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing and Wearable Computers (UbiComp), 2019.

Book Chapter

- **Mathur A.**, Isopoussu A., Kawsar F., Smith R., Lane N.D. and Berthouze N. *Towards the Design and Evaluation of Robust Audio-Sensing Systems*. Book chapter in Human Activity Sensing, Springer 2019.

In addition to these works, I was also involved in the following publications that are not part of this thesis.

- **Mathur A.**, Zhang, T., Bhattacharya, S., Veličković, P., Joffe, L., Lane, N. D., Kawsar F, Lió, P. *A Deep Data Augmentation Training Method to Address Software and Hardware Heterogeneities in Wearable and Smartphone Sensing Devices*. Proceedings of the 17th ACM International Conference on Information Processing in Sensor Networks (IPSN), 2018.

- Beutel, D.J., Topal, T., **Mathur A.**, Qiu, X., Parcollet, T. and Lane, N.D. *Flower: A Friendly Federated Learning Research Framework.* arXiv preprint arXiv:2007.14390.
- Gong T, Ramos AG, Bhattacharya S, **Mathur A.**, Kawsar F. *AudiDoS: Real-Time Denial-of-Service Adversarial Attacks on Deep Audio Models*. Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019
- Min, C., Montanari, A., **Mathur A.**, Kawsar, F. *A closer look at quality-aware runtime assessment of sensing models in multi-device environments.* Proceedings of the 17th ACM Conference on Embedded Networked Sensor Systems (SenSys), 2019.

1.4.2 Datasets

We created a large-scale ASR dataset named Libri-Adapt containing examples of a number of real-world heterogeneities in speech data. This dataset paper has been published at IEEE ICASSP [53] and the dataset has been released to the academic community.

1.4.3 Recognition

My research contributions were recognized by the academic community in the following ways:

- I was selected as a member of the *ACM Future of Computing Academy (FCA)* in 2019. ACM FCA is a group of 36 young computing professionals selected from across the globe to address the emerging challenges in the field of computing.
- I was a finalist for the Gaetano Boriello Outstanding Student Award at ACM Ubicomp 2019. This award recognizes students making outstanding contribution in the field of Ubiquitous and Pervasive Computing.

Chapter 2

Background

In this chapter, we first provide background on sensor-based ML systems operating on inertial and speech data (§2.1). Next, in §2.2, we highlight the scalability challenges for these systems by reviewing the literature on sensor and user-induced heterogeneities in the data. The main objective of this review is to understand the existing approaches used to tackle sensing heterogeneities, and to identify gaps in the literature to inform the research questions of this thesis. In §2.3, we discuss that the composite effect of sensing heterogeneities could be interpreted as domain shift. §2.4 then presents a general review of unsupervised domain adaptation (uDA) literature, and explains how uDA techniques have been used to address sensing heterogeneities in speech and inertial sensing models.

Finally, §2.5 summarizes the key findings from the chapter and explain how they lead to our research questions. Later while presenting our technical contributions in Chapters 4-7, we also provide a more focused literature review related to the contents of each chapter.

2.1 Computational Models for Sensor Data

In this thesis, the scalability challenges to machine learning (ML) systems are primarily studied in the context of human sensing systems. Human sensing systems collect sensor data from personal devices owned by an end-user (e.g., a smartphone) or from devices in the user's environment (e.g., an Amazon Echo) and process them using ML models to infer user behavior and context. In particular, we focus on human sensing systems operating on speech and inertial sensor data.

As such, we first provide an overview of commonly used training and inference pipelines for speech and inertial sensing systems.[1]

[1]To show the broader applicability of some of the algorithms and systems developed in this dissertation, we also evaluate them with computer vision datasets in Chapter 7. Details about those vision datasets and visual recognition tasks are provided in Chapter 7.

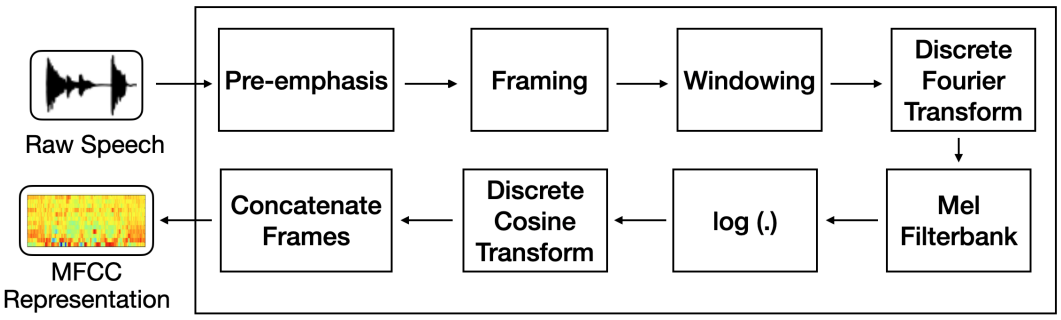


Figure 2.1: Pre-processing and feature extraction pipeline to obtain Mel Frequency Cepstral Coefficients (MFCCs) from raw speech data.

2.1.1 Audio and Speech Recognition

Speech has become a prominent modality to interact with our personal smart devices (e.g., smartphones) and near-body devices (e.g., Amazon Echo). In addition to the conventional use cases of speech processing such as automatic speech recognition (ASR), newer use-cases of audio and speech processing have emerged, such as inferring eating activities [59, 60], ambient conditions [61, 62], subjective user states [63], and productivity [64, 65]. Moreover, to facilitate speech-based interaction with personal digital assistants such as Siri, Alexa, or with voice assistants in smart cars, a significant emphasis has been placed on the task of Spoken Keyword Classification [66, 28] in recent years.

Feature Extraction. The field of speech processing has taken giant leaps in the last decade, primarily due to the availability of large labeled datasets and breakthroughs in deep learning techniques. The first step in deep learning-based speech modeling pipelines is to extract compact yet effective features to represent the speech data. One of the most widely used features for speech modeling are the Mel Frequency Cepstral Coefficients (MFCCs) [67], which are designed to take into account the unique aspects of human auditory processing and perception.

Figure 2.1 illustrates a common feature extraction pipeline for speech data. First, a pre-emphasis filter is applied on the raw time-domain speech signal to boost the amount of energy in high frequencies. Next, the speech waveform is assumed as quasi-stationary and is split into overlapping segments (or frames) of short time duration such as 25ms. After splitting the signal into frames, a windowing function such as the Hamming window is applied to each frame to reduce spectral leakage in Fast Fourier Transform (FFT) [68]. Thereafter, the frames are processed using Short-Time Fourier-Transform (STFT) to obtain a frequency-domain power spectrum. This power spectrum is evenly distributed in the frequency domain, however

human ear perceives sound in a non-linear fashion, by being more discriminative at lower frequencies and less discriminative at higher frequencies. To mimic this property of human ear, triangular filters on the Mel scale are applied to the power spectrum to obtain filter bank features. Filter bank features are also referred to as Mel-spectrograms. Finally, Discrete Cosine Transform (DCT) is applied to decorrelate the log filter bank features and yield a compressed representation of the filter banks in the form of cepstral coefficients, also known as MFCCs. The MFCC coefficients from all the frames are then stacked and often mean-normalized to form a two-dimensional (2D) representation of the speech signal.

Training and Inference Pipelines. Different deep neural network architectures are used to process the 2D MFCC feature representation depending on the modeling task at hand. For speech classification tasks such as Spoken Keyword Classification and Emotion Recognition where the goal is to assign a class or a label (e.g., Happy, Sad) to a given speech segment, convolutional neural networks are widely employed to learn local features in the data. As an example, Badshah et al. [69] proposed an architecture with three 2D convolution layers followed by three fully-connected layers to learn emotions in human speech. Other works [70] have leveraged temporal convolutional layers for local feature learning and also explored the use of a Long-Short Term Memory (LSTM) layer to learn [71] long-term dependencies from the learned local features. The training of these neural network models is done by optimizing a categorical cross-entropy loss between the model outputs and ground truth labels. At inference time, the test speech signal is again processed to extract MFCC features, which are then fed to the trained neural network to compute task-specific inferences. The performance of the model is determined by computing metrics such as accuracy, precision, recall or the F_1 score on a test set.

On the other hand, state-of-the-art Automatic Speech Recognition (ASR) models directly operate on top of the mel-spectrograms extracted from the speech data, and employ a series of convolution layers and bi-directional LSTM layers to extract local and global features, which are followed by 1-2 fully connected layers. These models are trained by optimizing the Connectionist Temporal Classification (CTC) loss [72]. At inference time, mel-spectrograms of speech are fed into the neural network, which outputs a probability distribution over characters in the vocabulary at each time step. Thereafter, a decoder is employed to convert the probability distributions to word transcripts, either using greedy search or beam search algorithms [73]. A widely used metric to evaluate the performance of ASR systems is the word error rate (WER), which is defined as the minimum edit distance between the transcripts generated by the ASR model and the ground truth transcripts.

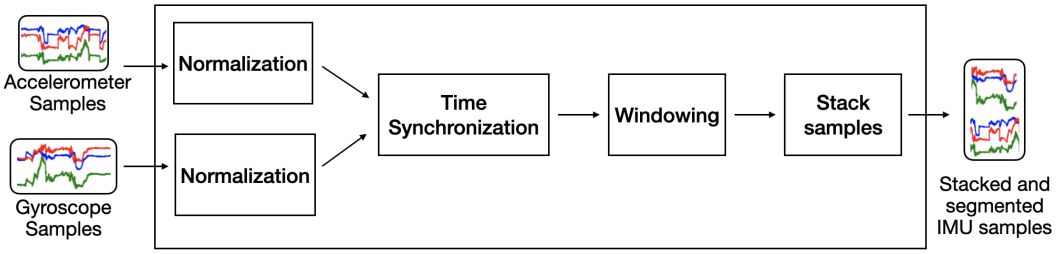


Figure 2.2: Pre-processing pipeline for data from the inertial measurement unit.

2.1.2 Human Activity Recognition

Modern smart devices such as smartphones and smartwatches are equipped with inertial measurement units (IMUs), which commonly consist of an accelerometer and a gyroscope sensor. In addition, newer class of IMU-equipped devices are also emerging such as earables [74], smart glasses [75] and neck pendants [76]. The data collected from the IMU sensors of these devices could be used for recognizing human activities, such as locomotion (e.g., walking, running), posture (e.g., standing, sitting), or composite activities (e.g., commuting, cleaning).

Feature Extraction. Initial approaches for human activity recognition (HAR) using data from the IMUs relied on extracting hand-crafted statistical features such as mean, variance, energy, spectral entropy, features derived from the empirical cumulative distribution function (ECDF) and FFT coefficients. These features were subsequently fed into shallow classifiers such as Random Forests or Support Vector Machines to map them to the activity labels [77, 78].

In the last five years, deep learning-based methods have emerged to automatically extract task-dependent features from raw accelerometer and gyroscope data and process them using deep neural networks for activity recognition. Figure 2.2 shows a commonly used pre-processing pipeline for IMU data. In general, the proposed methods first obtain accelerometer and gyroscope data from an IMU sensor placed on a specific position on the human body and then normalize the data using standard techniques such as mean normalization, min-max normalization or Z-score normalization. Thereafter, the data is segmented into smaller time windows, typically between 1-5 seconds, depending on the expected duration of the physical activities represented in the data. Inside each time window, acceleration readings from the three axes of the accelerometer and angular velocity measurements from the gyroscope are stacked on top of each other to form a 2D representation of the time-series IMU data. For instance, considering a sampling rate of 50Hz and 3-second long time windows, each window is represented by a 2D array of dimensions 150x6.

Training and Inference Pipelines. For training HAR models on IMU data, convolutional neural networks (CNNs) with one-dimensional (1D) convolution kernels have been used in the literature [79, 80, 81]. For example, Hammerla et al. [79] used four 1D convolutional layers, each followed by a pooling layer for feature extraction. The extracted features are then fed to a classifier made of one or multiple fully-connected layers. Both the feature extractor and classifier are trained by optimizing the categorical cross-entropy loss. Other works have also employed Recurrent neural networks with Gated Recurrent Units (GRUs) or Long Short-Term Memory (LSTM) units for training HAR classifiers [82, 83, 79].

2.2 Scalability Challenges for Inference Models

In this section, we describe the challenges in scaling speech and inertial inference models in practical settings.

2.2.1 Challenges in Scaling Speech Models

The main challenges in scaling speech models to new scenarios are listed below.

Speaker and Environment Heterogeneity. Variations in speaker characteristics (e.g., accents, pronunciations) pose a major challenge for scaling speech models to new users. This challenge has led to extensive research in the speech community to develop models robust to speaker variations. To this end, a popular approach is to extract speaker-specific features such as i-vectors [84] and incorporate them in the training pipeline. Initially proposed in the context of speaker verification systems, i-vectors aim to encapsulate the unique characteristics of a speaker's identity (such as accent, pitch) in a low-dimensional fixed-length representation. By concatenating i-vectors with other speech features such as the MFCCs during training, [85, 86, 87] have shown that more robust speech recognition models can be trained.

Another significant challenge for speech models comes from the acoustic environment in which the speech is recorded. Quite expectedly, the signal-to-noise (SNR) ratio decreases in noisy acoustic environments (e.g., in heavy traffic, inside a pub) which in turn degrades the performance of speech models. For example, Chon et al. [88] found that Gaussian Mixture Models (GMM) and Random Forest models trained for ambient sound classification suffer from poor precision and recall when deployed in unconstrained environments. Similar findings on the adverse impact of diverse acoustic environments were found for speaker turn-taking detection models [64] and acoustic stress detection models [89]. In order to counter acoustic

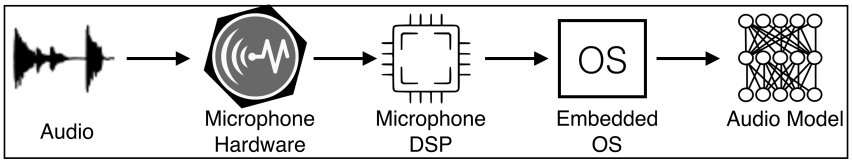


Figure 2.3: Sensing and inference pipeline for audio models.

environment noise, two types of approaches are proposed: first are the traditional noise subtraction [90] and statistical speech enhancement [91] techniques that aim to enhance the signal-to-noise ratio in the data. More recently, deep neural network based noise removal techniques have also been proposed that aim to learn a model that can map noisy speech to its clean counterpart [92]. On the other hand, [93, 94, 95] showed that noise-robust speech models could be trained by augmenting the speech samples with different ambient noise samples, and using representation learning to extract noise-robust features. Further, [96] employed a teacher-student learning approach wherein a teacher model is trained on a large clean speech dataset, and is then used to generate soft labels for training a noisy speech model using supervised learning. However, this work requires parallel data in the clean and noisy conditions, which could be challenging to obtain in practical settings.

Microphone-induced Heterogeneity. The microphone used to record the audio signal also introduces heterogeneity in the data. As shown in Figure 2.3, before an audio signal even reaches the ML model, it goes through a number of processing stages. Firstly, the analog signal is captured by the microphone's acoustic sensor and converted into a digital signal. Thereafter, the signal is processed by an on-board Digital Signal Processor (DSP), where audio enhancement techniques such as noise reduction, echo cancellation, dereverberation are applied to it. Optionally, multi-channel microphone arrays [97, 46] employ acoustic beamforming algorithms to combine the audio outputs of different microphone channels. Finally, the operating system exposes the processed audio signal to user applications such as an ML model for training or computing inferences.

Both the *hardware* and *software* components of this pipeline can introduce confounding artifacts in the signal. Das et al. [44] found that acoustic sensors exhibit variability in their outputs, which is caused by changes in the chemical composition of sensor components, wear in the manufacturing machines, or changes in temperature and humidity between manufacturing and operational conditions. Through a study done on 52 smartphones, they showed that this hardware-induced variability also gets reflected in the MFCC features, and it could be used to fingerprint and uniquely identify a device with an accuracy of 97%. We expect that these forms of hardware variability would be even more prominent for low-cost embedded-scale

microphones gaining popularity these days. To counter these hardware effects, Garg et al. [51] proposed a technique to calibrate microphones against a reference microphone using a frequency sweep signal or a reference sound.

On top of the hardware-related variations, the software processing pipelines of microphones can add further heterogeneities to the data. The on-board DSP on the microphones run a number of audio enhancement algorithms, whose parameters are fixed by each microphone manufacturer. For example, in the case of multi-channel microphone arrays [97, 46], acoustic beamforming algorithms are employed to improve the SNR of the audio. However, implementation differences in such algorithms across manufacturers could lead to variability in the output data from different microphones. Moreover, [48, 49] showed that acoustic beamforming algorithms are also sensitive to microphone parameters such as gain mismatches between the constituent microphones of an array, and could add unintended biases in the speech data that is passed to the ML model. As a solution, [98, 99] proposed doing the beamforming-based speech enhancement jointly with acoustic modeling and found that it outperforms the approach where beamforming is done agnostic of the acoustic modeling task. This solution is attractive when the ML model has access to the raw speech waveforms from each channel of the microphone array, however it is not applicable in many real-world scenarios where the device manufacturers only expose the beamformed speech output to the ML models.

2.2.2 Challenges in Scaling HAR Models

Research in HAR has revealed three key challenges in scaling HAR models to real-world scenarios.

User-induced heterogeneity. When HAR systems are deployed in practice, they will encounter heterogeneity in the accelerometer and gyroscope data caused by variations in body shapes, fitness levels, movement characteristics of different users [100, 101]. Unsurprisingly, such variations could result in performance degradation of an HAR system. A number of prior works have proposed developing user-specific models [77, 102, 103] where an HAR model is trained separately for each user to capture the unique movement characteristics. While these works achieve impressive recognition performance, on the downside, they need labeled data from each end-user of the HAR system, which increases the cost of training the model.

The alternate approach is to train user-independent models, that have lower recognition performance than the user-specific models initially [104, 105], but could be fine-tuned or personalized for individual users by collecting labeled data from them

during deployment [106, 107]. To reduce the labeling effort needed for model personalization, [108, 109] proposed employing active learning techniques to find the most informative samples for labeling. Although active learning approaches reduce the labeling burden, they do not entirely eliminate it. In §2.4.2, we discuss some of the recent domain adaptation literature that aims to transfer HAR models to new users only using unlabeled data.

Device-induced heterogeneity. Biases in IMU sensors that generate accelerometer and gyroscope data for HAR models can also induce heterogeneity in the data. Broadly, there are two types of biases in IMU sensors: deterministic and stochastic. Deterministic biases are caused by variations in sensor components across manufacturers, imperfections introduced in the analog circuitry of the sensor during the manufacturing process [43], or even by the packaging used for the unit [110]. Another important cause of deterministic biases is the temperature differences between initial calibration and operational stages [111]. On the contrary, stochastic biases emanate from the electronic noise interfering with IMU operations and are typically assumed Gaussian in nature [50, 111]. Prior research [112, 113] has highlighted that deterministic biases are the more prominent cause of heterogeneity in IMU data. In particular, [43] did a study with IMU-equipped smartphones and found that biases in their accelerometers are computationally so significant that they could be used to fingerprint a smartphone and identify it with over 99% accuracy from a pool of 100 devices. Moreover, Grammenos et al. [50] showed that IMU biases across devices could also degrade the accuracy of HAR models trained to detect locomotion states of a user by up to 10%. Various sensor calibration techniques [113, 50] have been proposed to counter the deterministic IMU errors; for example, [50] first identifies when a smartphone is in a stationary state and then use the earth's gravity as a reference to calibrate the IMU.

In addition to the hardware-induced biases, Stisen et al. [45] found that run-time factors such as CPU load on a smartphone can lead to instability in the sampling rate of IMU sensors. Moreover, they uncovered unpredictable delays in the attachment of timestamps to a sensor sample by the smartphone operating system, which also introduces sampling rate errors in the IMU data. As a solution, they proposed a clustering technique based on the Affinity Propagation algorithm, wherein devices are assigned to different clusters based on their respective IMU heterogeneities. Thereafter, an activity recognition model is trained for each cluster of devices on their aggregated labeled data. A limitation of this approach lies in its scalability — for every new device on which the HAR system is deployed, we first need to collect labeled data and repeat the clustering step. Thereafter, the HAR model needs to be

retrained for the cluster to which the device is assigned.

Placement-induced heterogeneity. Finally, the placement of the IMU on the human body itself could be a major source of heterogeneity in sensor data. Inertial sensors can be worn on the wrist (in a smartwatch), on the ear (in an earbud), or be placed inside a user's trouser and shirt pockets (smartphones). More critically, the sensor placement is not always static and may even change during the course of an activity based on a user's preference. For instance, a smartphone may move from the pocket to a user's hand, and then to the ear and then go in a handbag – all while the user is engaged in a certain physical activity. Researchers have tackled this challenge by developing placement-specific HAR models [114, 115] and have shown that these models are more accurate than placement-independent models such as [116]. A limitation of this approach is that it requires collecting labeled data from each body position where the sensor is likely to be placed, thereby raising questions on its scalability and cost-effectiveness.

2.3 Sensing Heterogeneity and Domain Shift

In the previous section, we discussed that variations in user preferences, deployment environment, and sensor data acquisition pipelines could potentially introduce heterogeneities in the data and hinder the scalability of HAR and speech models. This thesis offers solutions for two specific scalability challenges. In Chapter 4, we focus on countering the heterogeneities in speech data introduced by the hardware and software components of the sensing pipeline. Thereafter, in Chapter 5, we investigate how to improve the performance of HAR models in the presence of device and sensor placement-induced heterogeneities in inertial data.

The literature review in §2.2 highlighted various existing solutions to address these problems. For example, calibration methods have been proposed to fix hardware biases [113, 51] and new approaches for acoustic beamforming [98, 99] have been proposed to counter software-induced heterogeneities. These types of solutions assume low-level access to the sensor hardware or the ability to modify the signal processing algorithms in the sensing pipeline. However, in many practical settings, an ML model is considered an end-consumer of the sensor data and does not have access to the low-level sensing pipelines. For these settings, we argue that the composite effect of various data heterogeneities could be viewed as a form of domain shift between training (or *source domain*) and deployment conditions (or *target domain*), and can be addressed in the training and inference pipelines of ML models. In doing so, we can relax the assumption that the ML model will have low-level

access to the device hardware or the sensing pipeline, and develop solutions that rely only on the data received by the ML model.

We also reviewed various approaches to address these heterogeneities in the ML pipeline. For example, [114, 115] proposed to train placement-specific HAR models to counter placement-induced heterogeneities in the IMU data. Similarly, [45] clustered IMU devices based on the data biases in them and trained an HAR model for each cluster. Other works [96, 106] have looked at model personalization and teacher-student learning techniques for scaling ML models. The common limitation of these approaches is their reliance on labeled data from each deployment condition (e.g., from each device or each sensor placement condition). However, the collection of labeled sensor data is expensive and time-consuming, and repeating this exercise across multiple deployment conditions is infeasible.

Ideally, we would like to adapt or transfer a source domain model to a target domain, only using unlabeled data from the target domain. For example, one may buy a new embedded microphone, such as [47] to develop their own speech interaction device that can recognize speech commands from the user (e.g., unlock phone, play music). Since no labeled data is available from this new microphone to train an ML model, the developer can download a Spoken Keyword Classification model trained on labeled data from another microphone (i.e., source domain) to bootstrap their ML pipeline. As users interact with the new microphone, the developer can collect speech utterances from them (with their permission) and build an unlabeled dataset for the new microphone. Now the key challenge is to adapt the source domain model for the new target domain microphone, only using the unlabeled data collected from the users.

2.4 Unsupervised Domain Adaptation

Unsupervised Domain Adaptation (uDA) is a sub-field of machine learning which addresses the challenge of adapting a model trained in a source domain to a target domain using unlabeled target data. The solutions to sensing heterogeneity that we propose in this thesis are also based on the principles of (uDA). In this section, we present an overview of uDA and review prior works in this area.

2.4.1 Overview

Below we introduce some notations and describe the general objective of Unsupervised Domain Adaptation (uDA).

Notations. Let $\mathcal{X}$ and $\mathcal{Y}$ denote an input space and a label space. For simplicity, we assume that the label space is discrete: $\mathcal{Y} = \{1, 2, \ldots, k\}$. We use $\mathbf{x}$ and y to denote random variables which take values in $\mathcal{X}$ and $\mathcal{Y}$ respectively. We define a domain $\mathcal{D}$ as a distribution $p(\mathbf{x}, \mathrm{y})$ over $\mathcal{X} \times \mathcal{Y}$ and denote it as $\mathcal{D} = (\mathcal{X}, \mathcal{Y}, p(\mathbf{x}, \mathrm{y}))$. Further, let $p(\mathbf{x})$ denote the marginal data distribution over $\mathcal{X}$.

In the domain adaptation setting, we have a source domain $\mathcal{D}_S = (\mathcal{X}_S, \mathcal{Y}_S, p_S(\mathbf{x}, \mathrm{y}))$ and a target domain as $\mathcal{D}_T = (\mathcal{X}_T, \mathcal{Y}_T, p_T(\mathbf{x}, \mathrm{y}))$ such that $\mathcal{D}_S \neq \mathcal{D}_T$. One of the most common settings under which domain adaptation is studied is when the input spaces, label spaces, and the conditional distributions are identical between domains, i.e., $\mathcal{X}_S = \mathcal{X}_T$, $\mathcal{Y}_S = \mathcal{Y}_T$, $p_S(\mathrm{y}|\mathbf{x}) = p_T(\mathrm{y}|\mathbf{x})$, but the marginal data distributions vary between domains, i.e., $p_S(\mathbf{x}) \neq p_T(\mathbf{x})$. The assumption about conditional distributions $p(\mathrm{y}|\mathbf{x})$ being the same between domains is strong, but reasonable for the sensor-induced domain shifts identified in §2.3.

Objective of Unsupervised Domain Adaptation. Assume that we have access to labeled observations $\mathbb{X}_S = \{(\mathbf{x_{s_i}}, y_{s_i})\}_{i=1}^{N}$ from the source domain $\mathcal{D}_S$ where $\mathbf{x_{s_i}}$ is a data sample and y_{s_i} is the associated class label. Similarly, we have an unlabeled dataset from the $\mathbb{X}_T = \{\mathbf{x_{t_i}}\}_{i=1}^{M}$ from the target domain $\mathcal{D}_T$.

Using the labeled source dataset $\mathbb{X}_S$, we can train a classifier $f_S(.)$ to map the source domain samples to their corresponding labels. However, if the source classifier were to be deployed in the target domain, it could suffer significant accuracy degradation due to domain shift. The goal of unsupervised domain adaptation is to learn a target domain classifier $f_T(.)$ given $\mathbb{X}_S$, $\mathbb{X}_T$ and $f_S(.)$, but in the absence of any labeled observations from the target domain.

2.4.2 Techniques

We now review some relevant techniques for unsupervised domain adaptation (uDA) designed for deep neural networks. As discussed in §2.1, deep learning based methods are currently state-of-the-art in the areas of speech and human-activity recognition, and the work in this thesis studies domain shift specifically in the context of deep neural networks. For completeness, we note that there is also rich literature on domain adaptation prior to deep learning and we refer the reader to [117, 118, 11] for a detailed survey. Also note that the following is an initial review of uDA; while presenting our technical contributions in Chapters 4-7, we discuss additional related work relevant to the contents of each chapter.

One of the most common approaches in uDA is to align the feature representations

of source and target domains. In the context of deep learning, it is often achieved by training a feature extractor neural network and minimizing the discrepancy in the outputs of the feature extractor when it is fed samples from source and target domains. If the samples from both domains can be mapped into a domain-invariant feature representation, then a classifier trained on the source domain can be expected to generalize to the target domain under some assumptions. The proposed methods mainly differ in how the divergence between the outputs of the feature extractor is computed and minimized.

Tzeng et al. [119] and Long et al. [120] proposed using Maximum Mean Discrepancy (MMD) as a metric to compute the divergence between source and target feature representations. To compute MMD, the source and target feature representations are mapped to a reproducing kernel Hilbert space (RKHS) using a kernel function. More specifically, Tzeng et al. [119] proposed the use of grid search to find the layer depth and layer dimension at which to minimize the MMD loss between source and target representations. Long et al. [120] extended this work by proposing Deep Adaptation Network (DAN) that minimizes MMD at multiple layers of the network using a multiple kernel variant of MMD. Both these works were also evaluated on computer vision datasets, namely Office-31 and Office-Caltech, which contain images of everyday office objects captured in different conditions. Further, Sun et al. [121] proposed Deep CORAL where the distribution divergence is computed using second-order statistics (covariances) and evaluated their approach on vision datasets.

Adversarial training is another widely used method to align the source and target feature representations. Here an auxiliary binary classifier named as *domain discriminator* is employed to classify whether a feature is generated from the source or the target domain data. The training of the domain discriminator and the feature extractor neural networks take place in an adversarial manner, inspired by Generative Adversarial Networks [2]. While the domain discriminator is trained to correctly classify the domain of the data, the aim of the feature extractor is to fool the discriminator by generating feature representations that are domain-invariant. Both the two neural networks play this competitive game and in the process, both become better at their respective tasks, more importantly from the perspective of domain adaptation, the feature extractor learns to map source and target data into domain-invariant feature representations.

Based on this general concept, Ganin et al. [122] proposed DANN which uses a Gradient Reversal Layer between the feature extractor and the discriminator to per-

form adversarial training. Tzeng et al. [123] proposed ADDA which uses the label inversion trick to facilitate adversarial training of the domain discriminator and feature extractor. Further, Sankaranarayanan et al. [124] proposed using a conditional GAN as an auxiliary component to compute domain alignment. In this work, the conditional GAN is trained to generate source-like images from the shared feature representations which are then compared with the original source images using a discriminator. Finally, Shen et al. [125] proposed WDGRL where instead of a domain discriminator, a neural network is used to approximate the Wasserstein distance between the domains, which is then minimized using adversarial training. All the above approaches were evaluated on computer vision datasets.

There also have been applications of uDA techniques in speech modeling. [126, 127, 128] proposed using the DANN method from [122] for the tasks of speech, speaker and emotion adaptation respectively. Further, Hsu et al. [129] proposed an adaptation approach based on variational autoencoders in which they transform nuisance attributes of speech that could be domain-specific and irrelevant to the recognition task. By doing so, they are able to transform source-domain speech samples such that they appear to be from the target domain, and use them to augment the dataset for supervised learning. [130] also used the DANN approach to adapt the parameters of an ASR model to counter channel and environment noise. Finally, the exploration of uDA for human activity recognition (HAR) using inertial data is in nascent stages. HDCNN [131] was an early work on transferring HAR models from a smartphone (source domain) to a smartwatch (target domain) by minimizing the Kullback-Leibler (KL) divergence between each layer of the source and target classifiers. Akbari et al. [132] extended this work by generating stochastic features from the IMU data and aligning them between source and target domains by minimizing the KL divergence. These works represent some of the very recent initiatives in applying domain adaptation techniques to HAR.

In the next section, we summarize the key findings from our literature review and identify the gaps in the literature.

2.5 Discussion and Contributions

Through the background and literature review presented in this chapter, we have identified the following gaps in the literature, which are addressed in this thesis.

- In the context of speech data, hardware and software variations in recording microphones and sensing pipelines can introduce computationally-significant het-

erogeneities in the data. There have been efforts to fix hardware-related variations and software-related variations in the sensing pipeline by calibrating microphones [51] or by incorporating more robust acoustic beamforming algorithms [49] in the sensing pipeline. However, when speech models are deployed in a practical ML system, they are often treated as end-consumers of the data and may not have the visibility or ability to modify the underlying hardware parameters or the signal processing algorithms in the sensing pipeline. Therefore, we need to develop solutions to counter the domain shift caused by microphone-induced heterogeneities directly in the training or inference pipelines of speech models.

- Heterogeneity could be introduced in inertial data from accelerometers and gyroscopes due to variations in sensor hardware, run-time software-related effects, and sensor placement on the human body. While each of these have been studied in the literature in isolation [50, 45, 133], their composite effect on the performance of human activity recognition (HAR) models has not been studied. Moreover, the data-driven solutions to address them are not scalable as they require training separate ML models for each type of heterogeneity. Hence, there is a need for a solution that can handle the combined effect of these heterogeneities directly in the training or inference pipeline of HAR models.

- For the two problems discussed above, we can borrow solutions from the literature which use labeled data during deployment to fine-tune or personalize the ML model to deployment scenarios (e.g., for a new microphone or a new IMU sensor placement). However, labeling sensor data in each deployment scenario is expensive and time-consuming, and reduces the practicality of these solutions. Instead, we argue that it is more practical to address these problems through the lens of Unsupervised Domain Adaptation (uDA), which does not assume the availability of any labeled data in the deployment scenario (or the target domain).

  While uDA algorithms have been extensively studied for visual recognition tasks, their applications in our problem areas – addressing microphone-induced heterogeneity in speech models and IMU sensor placement related heterogeneities in HAR models – are in nascent stages. Our contributions in Chapters 4 and 5 aim to bridge this gap in the literature. Before presenting these solutions, we quantify the effect of these domain shifts on state-of-the-art speech and HAR models in Chapter 3.

- There has been substantial research in the area of unsupervised domain adaptation, which has focused on developing new algorithms and neural architectures

to adapt prediction models between domains. However, they have not explored in detail how these adaptation approaches would integrate with practical ML systems. For example, in some ML systems, the source prediction model may be **opaque**, i.e., its parameters may not be accessible during the adaptation process. This is a likely scenario when source model developers may want to keep their prediction model private for commercial reasons; examples include ML offerings by AI-as-a-service providers such as Google [134], Microsoft [135] and Amazon [136] which provide an API for users to send their sensor data to the model and obtain inferences. On the contrary, in **transparent** ML systems, the parameters of the prediction model can be accessed and modified during adaptation. Examples of transparent ML systems include scenarios when the prediction model is publicly known (e.g., downloaded from a public model repository) or when the source model developer themselves would like to adapt the model parameters to new domains.

As such, a major contribution of this thesis is to bring an ML-systems perspective in designing uDA algorithms. More specifically, while proposing a solution to address domain shift in speech models in Chapter 4, we consider the scenario of opaque ML systems and propose an adaptation solution that operates at the level of data instances and does not assume the knowledge of model parameters. Similarly, while studying the domain shift problem in HAR models in Chapter 5, we consider the scenario of transparent ML systems and propose a solution that modifies the source HAR model's parameters to adapt it to a target domain.

- A key assumption in many uDA algorithms is that the label spaces of the source and target domains are identical, i.e., $\mathcal{Y}_S = \mathcal{Y}_T$. However, in practical ML systems, it is infeasible to enforce this assumption because we have no control over the classes or labels that will be encountered in the target domain. In Chapter 6, we present a solution that relaxes this assumption and scales ML systems to target domains that may not have the same label space as the source domain.

- Finally, all the uDA algorithms reviewed in this chapter worked under the assumption that source and target domain datasets are hosted on the same machine. In many ML deployment scenarios, this would be a limiting assumption as domain datasets could be private, located on distributed nodes, and cannot be exchanged during the adaptation process. Moreover, in practice, ML systems would need to be scaled to not just one, but multiple target domains, and it is unclear how uDA algorithms would cater to such a setting. In Chapter 7, we present a distributed adaptation solution to scale ML systems to multiple target domains,

wherein all the domains are geographically distributed, and their raw data cannot be exchanged during adaptation.

2.6 Summary

We presented background on speech and inertial recognition models in §2.1, including the pre-processing and feature extraction pipelines, and the deep learning based architectures employed in the literature to train these models. Thereafter in §2.2, we explained the challenges in scaling speech and HAR models caused by heterogeneities induced in the data due to sensor hardware, signal processing algorithms, user demographics and user preferences such as sensor placement on the body. We reviewed the sensor systems and machine learning literature aiming to address many of these challenges, and identified the limitations of those works in practical ML systems.

Next, in §2.3, we argued that the composite effect of these heterogeneities could be viewed as a form of domain shift between training and deployment settings of an ML system. This domain shift could be countered using Unsupervised Domain Adaptation (uDA) techniques. As such, we reviewed some prominent uDA literature, mainly focusing on adversarial training approaches in §2.4.

Based on the literature review, we identified two application scenarios where uDA based approaches have not been explored in detail: (i) microphone-induced heterogeneity in speech models, and (ii) device and placement-induced heterogeneity in HAR models. Further, we uncovered that there is a lack of research on how uDA algorithms would scale in different ML system settings, such as opaque, transparent and distributed ML systems. These gaps in the literature motivate the contributions of this thesis, which are also summarized in §2.5. Later, while presenting our technical contributions in Chapters 4-7, we also provide a more focused review of uDA relevant to each chapter's contents.

In the next chapter, we begin our research investigation by quantifying the impact of microphone-induced heterogeneity on speech models, and device and placement-induced heterogeneity in HAR models.

Chapter 3

Quantifying the Effect of Domain Shift on Sensor Inference Models

In this chapter, we seek to address our first research question on quantifying the adverse effects of domain shifts induced by (i) microphone-induced heterogeneity on speech models (§3.1), and (ii) device- and placement-related heterogeneity on human activity recognition models (§3.2). The findings from this chapter will serve as the motivation for the subsequent chapters, where we propose unsupervised domain adaptation solutions to counter these domain shifts.

3.1 Microphone Heterogeneity in Speech Models

A major challenge for our investigation into this problem is the lack of available datasets that explicitly capture both hardware and software-induced heterogeneity in speech data. Although the popular CHiME-3 dataset [137] is often used to study the effect of channel variations in speech models, this dataset was recorded on single-channel microphones from the same manufacturer. As such, it does not contain hardware-induced heterogeneities often caused by differences in sensor manufacturer, or the software-induced heterogeneities due to variations in signal processing and beamforming algorithms in microphone arrays. In this section, we first present a methodology to collect large-scale datasets that have the desired microphone-induced heterogeneities in them. Thereafter, we quantify how these heterogeneities impact the accuracy of two speech-based ML models, namely Spoken Keyword Detection and Automatic Speech Recognition.

3.1.1 Data Collection Methodology

This data collection study aims to record large-scale speech datasets (in the order of hundreds of hours) from multiple off-the-shelf microphones that are representative

Device	Channels	Potential Use-cases	Advertised Signal Processing Capabilities
Matrix Voice	7	Internet of Things (IoT) systems, voice assistants, smart home products	Beamforming, dereverberation, noise cancellation
Respeaker	7	Internet of Things (IoT) systems, voice assistants, smart home products	Voice activity detection, beamforming, background noise suppression
PlayStation Eye	4	Gaming consoles	Voice location tracking, echo cancellation, beamforming, background noise suppression.
USB Mic	1	Embedded-scale IoT systems	-
Google Nexus 6	3	Smartphone interaction, voice assistants	Beamforming, background noise suppression
Shure MV5	1	Podcasting, home singing	Auto gain, Equalization

Table 3.1: Technical specification of the microphones used for data collection.

of the microphones used in real-world speech interaction applications. Moreover, in order to systematically study the impact of microphone-related factors, it is important to control for other variables in the dataset, such as the speech content and recording environment across microphones. Below is the methodology adopted to collect the datasets from multiple microphones.

Microphones. Speech-based ML systems are becoming pervasive and are being deployed in a range of devices, including smartphones, wearables, smart home appliances, video game consoles, and even Internet of Things (IoT) devices. Moreover, in the past few years, many embedded-scale microphones (both single- and multi-channel) have been released that can be used in conjunction with microcomputers such as Raspberry Pi and cloud-based speech models to develop customized speech analytics systems [47]. As such, it is important to study if the variability in microphones across such diverse use-cases will pose a challenge to speech classifiers. In this direction, we employ seven different microphones listed in Table 3.1 to collect speech data and quantify the effect of microphone-induced domain shift on speech models.

Matrix Voice[1] and ReSpeaker[2] are circular 7-channel microphone arrays that include on-device audio processing algorithms for de-reverberation and beamforming to combine the outputs of different channels. The potential use-cases of such microphone arrays are smart home products and voice assistants, such as Amazon Echo [47]. Both these microphone arrays use different models of MEMS microphones and implement customized signal processing and beamforming algorithms. As such, the data from these microphones is expected to contain both hardware and software-induced heterogeneities. Next, we use a PlayStation-Eye, a digital camera used in Sony PlayStation gaming consoles. It consists of a 4-channel microphone array used for speech interaction with the gaming console. The technical specifications for this array also include acoustic beamforming and noise suppression algorithms. Further, we employ a Google Nexus 6 as a representative smartphone microphone with which users are likely to interact while using speech applications on their mobile devices. Next, we use a single-channel USB microphone as an example of low-end microphones popularly used with Raspberry Pi to make IoT products. Finally, to establish a higher-quality microphone baseline, we employ a Shure MV5 Condenser microphone[3] which is often used in desktop applications such as podcasting, voice recordings. It can also be used as a plug-and-play microphone with mobile and wearable devices.

In summary, these microphones cover a broad set of use-cases in which speech-based ML systems are likely to be deployed. Besides, all the microphones are from different manufacturers, and four of them also employ signal processing algorithms for speech enhancement.

Method. While we can record speech utterances on all the above microphones in a small-scale experiment (e.g., by recruiting users to read some text under the same recording conditions), this data collection approach becomes very expensive for large-scale datasets, that are often needed to train deep learning models. As such, we adopt the methodology of *replay-and-record* to collect large-scale speech datasets on multiple microphones.

As shown in Figure 3.1, we use two types of host devices: a *replay host* and multiple *recording hosts*. The replay host is a laptop connected to a high-quality monitor speaker. Recording hosts serve as the host device for the various microphones introduced in Table 3.1, and are capable of initiating an audio recording on the attached microphone and saving the recorded output on the disk. We use a Raspberry Pi

[1] https://www.matrix.one/products/voice
[2] https://respeaker.io/usb_6+1_mic_array/
[3] https://www.shure.com/en-GB/products/microphones/mv5/

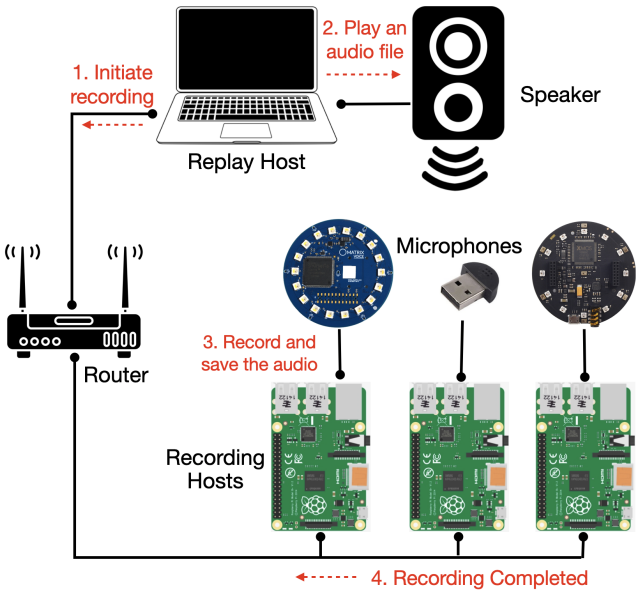


Figure 3.1: Data collection setup illustrating the Replay and Record methodology. Three microphones are shown in the figure, namely Matrix Voice, USB microphone and ReSpeaker.

as the host device for Matrix, ReSpeaker, USB, and PlayStation Eye microphones. The Shure microphone uses a Macbook Pro as the host device. The replay and recording hosts are connected with each other using a local area network (LAN).

We first download an existing publicly available speech dataset on the *replay host* and play the constituent speech files on the attached high-quality monitor speaker. The replayed speech is then simultaneously recorded on all the microphones in a quiet room, kept at a distance of 15cm from the speaker. To ensure that the speech recordings are time-synchronized across microphones, we implement a network-based synchronization scheme based on MQTT [138] messaging protocol. Before a speech file is played on the speaker, the replay host sends a MQTT message to the recording host of each microphone to *initiate an audio recording* process for the duration of the speech file (e.g., initiate a recording session for 10 seconds). Thereafter, the speech file is played on the monitor speaker and is recorded simultaneously by all microphones. After the recording completes, the recorded file is saved in WAV format on all the recording hosts, which then send a message to the replay host confirming that the recording is complete. Thereafter, the replay host proceeds to play the next file in the dataset.

This approach of replay-and-record has its pros and cons. It allows for collecting large-scale datasets needed to train deep learning models from multiple microphones, without requiring the recruitment of human subjects which can be both

time-consuming and extremely expensive. However on the downside, as the speech is replayed through a speaker, it is impacted by the speaker's transfer function. To counter this issue, we use a high-quality monitor speaker JBL LSR305 which has a reasonably flat frequency response [139], thereby ensuring minimal impact on the replayed audio's quality due to the speaker. More importantly, we argue that the replayed dataset still allows for studying the domain shift problem caused by different microphones, because the replay process itself could be interpreted as a form of shift on the original speech dataset, that is shared across all microphones.

Datasets. Following the replay-and-record methodology, we collect two multi-microphone speech datasets:

- *Multi-microphone Spoken Keywords:* The purpose of this dataset is to facilitate the development of Spoken Keyword Detection models, which identify the presence of a certain keyword class (e.g., Yes, No) in a given speech segment. The original dataset named Speech Commands was released by Google [140] and consists of 65,000 one-second long utterances belonging to 30 keyword classes. We replay this dataset on the monitor speaker and simultaneously record it on Matrix Voice, ReSpeaker and USB microphones. The dataset is split into training (75%) and test (25%) class-balanced subsets.
- *Libri-Adapt:* This dataset is built on top of the Librispeech-clean-100 dataset [141] that contains 100 hours of US English speech from users reading public-domain audiobooks. Using the replay-and-record methodology, the Librispeech-clean-100 training corpus is recorded on six different microphones listed in Table 3.1, resulting in a training dataset of 600 hours in duration. For each microphone, we also collect a 5.4 hour held-out test set derived from Librispeech test-clean corpus [141].

3.1.2 Experiments

Below we present the experimental setup and results of a quantification study that uses the two datasets presented in the previous section to study microphone-induced domain shift on various speech models.

Tasks and Architectures. The following neural network architectures are used for developing speech recognition models.

- *Spoken Keyword Detection (SKD)*: To study how microphone-induced domain shift affects Spoken Keyword Detection models, we use a small-footprint convo-

lutional neural network architecture proposed in [28] to train the SKD model. The input to this model is a two-dimensional tensor extracted from a one-second-long keyword recording, consisting of time frames on one axis and 24 MFCC features on the other axis. The model outputs a probability of a given audio recording belonging to a particular keyword class (e.g., Yes, No) or to an Unknown class. The model is trained on a source microphone's training set and evaluated on the test set recorded from a target microphone.

- *DeepSpeech2*: For experiments on Automatic Speech Recognition (ASR), we employ the Mozilla DeepSpeech2 pre-trained model [142] (release 0.5.0) as the base model. The model accepts a speech file in WAV format and generates a speech transcript; it has a relatively low word error rate (WER) of 8.22% on the LibriSpeech dataset. We first fine-tune this base model on the training data from a given source microphone (as collected in the *Libri-Adapt* dataset). Thereafter, the fine-tuned model for each source microphone is tested on the held-out test set from target microphones to compute the Word Error Rate.

- *Cloud ASR APIs*: Can microphone variabilities also impact cloud-scale ASR models that are trained with thousands of hours of data and have shown near-human accuracy on ASR tasks [135, 143]? Although we do not have access to the parameters of the cloud-scale ASR models, we can still query them using APIs provided by the cloud service. As such, we conduct experiments on ASR models from Google (using the Google Cloud Speech API [134]) and Microsoft (using the Bing Speech API [135]). Speech files from the *Libri-Adapt* test dataset are fed to these cloud-based ASR models through REST APIs, and Word Error Rate (WER) is computed on the output ASR transcripts.

Results. First, we present the effect of microphone heterogeneity on Spoken Keyword Detection (SKD) models. Table 3.2 reports the accuracy obtained on the Spoken Keywords dataset for a pair of training and testing microphones. As expected, when the SKD model is trained and tested on the same microphone, we obtain the highest detection accuracy (e.g., 81% on ReSpeaker dataset). On the contrary, when there is a mismatch between the training and test microphones, we observe a degradation in detection accuracy. For example, when the model trained on Matrix Voice is deployed on ReSpeaker and USB microphones, there is an absolute accuracy drop of 12.4% and 6.7% respectively.

A similar pattern is observed for the ASR task with the DeepSpeech2 model. In Table 3.3, we show the WER of DeepSpeech2 model when it is fine-tuned for different

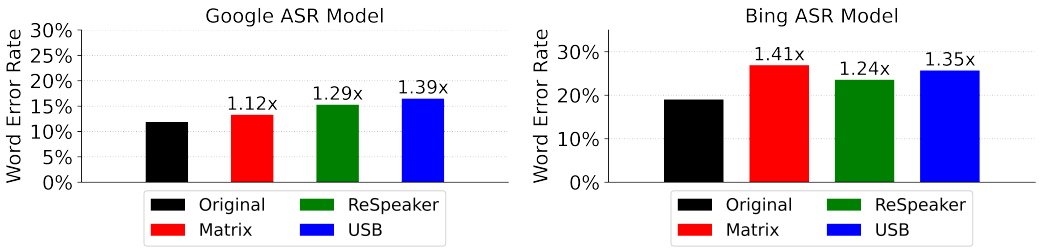
	Training Microphones		
	Matrix Voice	ReSpeaker	USB
Matrix Voice	**78.8**	66.5	75.8
ReSpeaker	66.4	**81.0**	76.1
USB	72.1	73.22	**82.3**

Table 3.2: Test set accuracy of the Spoken Keyword Detection model when trained and tested on various microphone pairs. The columns and rows correspond to the training and test microphone domains respectively.

	Training Microphones					
	Matrix Voice	ReSpeaker	USB	Shule	Nexus 6	PlayStation Eye
Matrix Voice	13.05	28.0	20.64	24.95	26.21	24.18
ReSpeaker	16.06	**12.66**	14.49	16.09	15.93	15.21
USB	13.30	14.44	**11.40**	14.41	15.39	15.38
Shule	**12.58**	13.10	11.61	**10.19**	12.96	13.07
Nexus 6	13.09	12.81	11.99	12.73	**12.27**	15.28
PlayStation Eye	14.01	13.65	14.24	16.16	16.52	**11.39**

Table 3.3: WER of a fine-tuned DeepSpeech2 model trained and tested on various microphone pairs. The columns correspond to the training microphone domain and rows correspond to the test microphone domain.

training microphones (in columns) and tested on other microphones (in rows). In most cases, we observe that when data from the same microphone is used for training and testing the model, it achieves the smallest WER, e.g., 11.39% in the case of PlayStation Eye. However, in the presence of domain shift caused by microphone mismatch, the WER increases to 24.18% when the PlayStation Eye model is tested on Matrix Voice. We also note that the WER obtained when models from other microphones are tested on Matrix Voice is significantly high (above 20% in all cases), however when the model is fine-tuned on Matrix Voice, we can achieve a reasonable WER of 13.05%. This significant gap presents a clear opportunity for domain adaptation algorithms to adapt speech models trained in one microphone domain to another.



(a) **(b)**

Figure 3.2: Impact of microphone variability on Google and Bing ASR models. Values on the bars illustrate the increase in WER over the original Librispeech dataset (black bar).

Next, we evaluate whether microphone variability also impacts cloud-scale ASR models. Figure 3.2 shows a comparison of WER obtained from the Google and Bing ASR models in four conditions: i) on the original Librispeech test data, ii) on Librispeech test data recorded on Matrix Voice, iii) on Librispeech test data recorded on ReSpeaker, and iv) on Librispeech test data recorded on the USB microphone. The latter three categories of test data come from our *Libri-Adapt* dataset, which is collected using the replay-and-record methodology.

In Figure 3.2, we observe that when ASR models are tested on speech data from Libri-Adapt, the WER increases over the baseline (i.e., the original Librispeech test data) by as high as 1.41 times. Although this finding is interesting, we cannot attribute this increase in WER only to microphone variability because the speech recordings in Libri-Adapt dataset may also have perturbations introduced by the replay speaker. However, we can compare the WER across speech segments recorded from the three microphones (Matrix, ReSpeaker, USB) as all of them will have the same speaker-induced effects. As we can observe in Figure 3.2, the WER varies between the three microphones (e.g., from 1.24x to 1.41x WER increase in the case of Bing ASR model), which suggests that even cloud-scale ASR models are not completely robust against microphone variability.

Finally, to understand the source of these microphone-induced variability at a signal level, we record a 4-second speech segment simultaneously on the three microphones, namely Matrix Voice, ReSpeaker, and USB inside a non-reflective anechoic chamber. Figure 3.3 shows the mel-spectrograms of the 4-second speech segment – we observe that the microphones exhibit differences in their frequency responses to the same speech input. For example, the data recorded from Matrix Voice has lower spectral power in the high-frequency ranges, which suggests that either this microphone array does not capture high frequencies well (hardware effect) or they are being filtered out by the microphone digital signal processor (software effect).

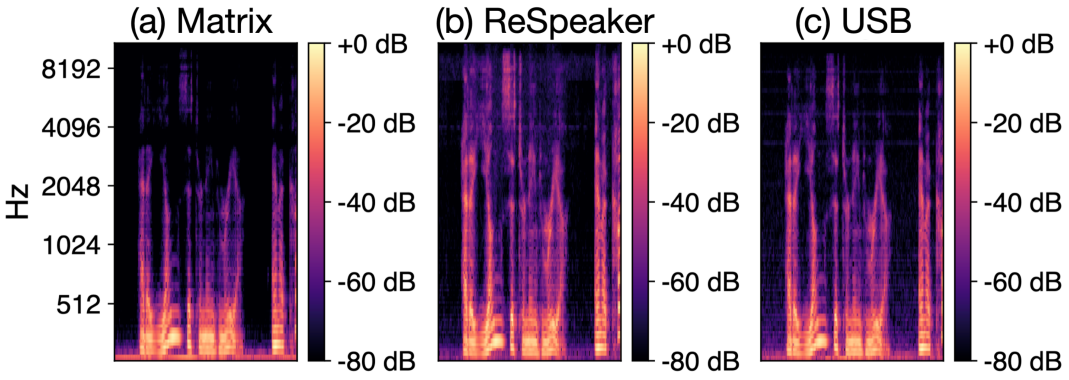


Figure 3.3: Difference in mel-spectrograms of a speech segment captured by three different microphones.

These kinds of subtle variabilities between microphones could induce domain shift between source and target domains, as reflected in our findings.

3.2 IMU Sensor and Placement Heterogeneity in HAR models

In this section, we quantify the effect of domain shift on the performance of Human Activity Recognition models, as caused by variations in IMU sensors and their placement on the human body.

3.2.1 Datasets

Unlike the microphone example above, there are HAR datasets already available containing IMU signals recorded from sensors placed at different body positions. Below we. describe the dataset used in our experiment along with the pre-processing operations done on the data.

We use the REALWORLD HAR dataset [1] which contains IMU data recorded from 15 participants performing 8 activities: *climbing stairs down and up, jumping, lying, standing, sitting, running/jogging, and walking*. As shown in Figure 3.4, each user was instrumented with different IMU-equipped smartphones and smartwatches, attached to 7 different body positions: head, chest, upper arm, waist, forearm, thigh, and shin. Users were then asked to naturally perform the physical activities, and accelerometer and gyroscope data was recorded from the devices simultaneously at a sampling rate of 50 Hz. Each activity was performed for 10 minutes, except for *jumping*, which was done for $\sim$1.7 minutes on average.

Figure 3.4: Figure from the REALWORLD dataset paper [1] showing different IMUs instrumented on the human body at different locations.

There are three major reasons why the REALWORLD dataset is ideal for this evaluation: a) the physical activities are performed in naturalistic settings as opposed to controlled experiments, b) the data across male and female participants are equally distributed which reduces any gender bias in the evaluation, c) to the best of our knowledge, this is the largest dataset of body position variability that is publicly available. In contrast, the popular OPPORTUNITY dataset [144] contains data only from four participants and only has one dynamic activity (i.e., walking).

Data characteristics and pre-processing. The accelerometer and gyroscope traces are segmented into time windows of 3 seconds, without any overlap. This window length was chosen empirically to align with the duration of various human activities in the dataset [4]. If a 3-second-long trace includes an activity transition, timestamp noise, or data points without labels, the trace gets discarded. The whole dataset is normalized to be in the range of -1 and 1. As shown in the IMU processing pipeline in Figure 2.2, the accelerometer and gyroscope axis are stacked on top of each other to create a 2-D representation of the time-series data. Finally, we use stratified splitting to divide the dataset into two parts: training set (75%) and test set (25%).

3.2.2 Experiments

Model Architecture and Training. We design a convolutional neural network (CNN) model based on the work by Hammerla et al. [79] and Almaslukh et al. [145]. The model consists of two components: a feature extractor and a classifier. The *fea-*

[4] We also experimented with one and two second long windows, however, there was no significant difference in HAR model performance. Therefore, we decided to use three-second windows, because longer windows reduce the number of ML model executions needed at inference time.

	Target Domains						
	head	chest	upperarm	forearm	waist	thigh	shin
head	**0.85**	0.52	0.50	0.22	0.19	0.48	0.39
chest	0.41	**0.92**	0.45	0.28	0.33	0.49	0.35
upperarm	0.29	0.38	**0.85**	0.23	0.22	0.48	0.54
forearm	0.12	0.11	0.11	**0.83**	0.26	0.11	0.09
waist	0.29	0.37	0.11	0.25	**0.93**	0.30	0.27
thigh	0.37	0.39	0.40	0.12	0.22	**0.92**	0.37
shin	0.18	0.32	0.35	0.13	0.28	0.26	**0.86**

Table 3.4: F_1 scores obtained when an HAR model is trained and tested on different body positions.

ture extractor is a 6-layer deep CNN with temporal (1-D) convolutional layers. We use LeakyReLU activations [146] with alpha= 0.3 and instance normalization [147] layers between convolutional layers for faster convergence. We also employ dropout regularization to avoid overfitting. The feature extractor takes as input a 3-second frame of pre-processed IMU data from a given body position and outputs a 150-dimensional feature vector. This feature vector is then passed as input to the *classifier*, which consists of two fully-connected layers and generates a k dimensional output where k is the number of activity classes (e.g., sitting, walking). For the REALWORLD dataset, $k = 8$.

The model is trained by optimizing the categorical cross-entropy loss. We use a mini-batch size of 64, which is chosen based on a hyperparameter search in {32,64,96,128}. Further, we employ the Adam optimizer [148] with a learning rate 1e-3, which is obtained by doing a hyperparameter search on {1e-2, 5e-2, 1e-3, 5e-3, 1e-4, 5e-4}. The training process is implemented in TensorFlow 2.0 and executed on a NVIDIA Tesla V100 GPU. TensorFlow's HParams API is used for hyperparameter tuning.

Evaluation Metrics. In line with the goal of this experiment, we train a model on data from a *training body position* (i.e., the source domain) and evaluate the model on a *test body position* (i.e., the target domain). Class-weighted F_1 score is used as the evaluation metric.

Results. Table 3.4 shows the results of our experiments. Each row in the table denotes a source domain from which we obtain the labeled dataset to train an HAR model. This model is then tested on various target domains (shown in columns), and we report the class-weighted F_1 score for each experiment setting.

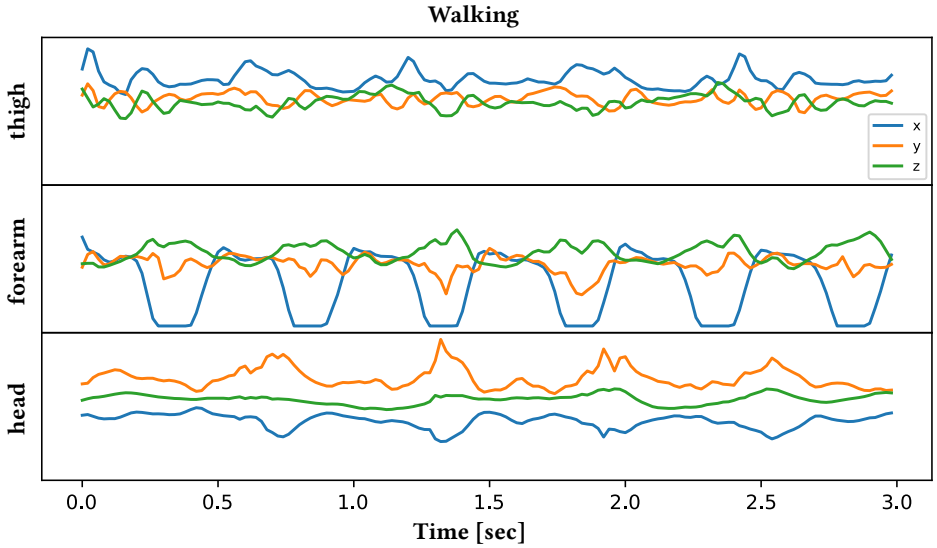


Figure 3.5: Accelerometer traces for the same physical activity (walking) collected from IMU sensors placed at three different body positions.

First, we analyze the performance of HAR models when they are trained and tested on the data from the same sensor and body position. This represents the ideal scenario with no domain shift caused by variations in sensor hardware or sensor placement; hence we can expect a high classification accuracy. In Table 3.4 (diagonal), it can be observed that models trained on *head* and *thigh* provide F_1 scores of 0.85 and 0.92 respectively, when they are evaluated on the test set from same body position on which they are trained.

Next, we analyze the performance of HAR models when they are trained and tested on data from different sensors and body positions. From Table 3.4 (non-diagonal), we see a significant accuracy degradation in all the source-target domain pairs. For example, when the source='thigh' model is tested on target='head', the F_1 score drops from 0.92 to 0.37. Figure 3.5 provides more intuition behind this finding, where we plot an accelerometer trace collected from three body positions while a user is walking. As we can see, the accelerometer data captured from IMUs placed at different body positions show clear differences, which highlights the challenges in scaling HAR models.

3.3 Summary

In this chapter, we quantified the effect of domain shifts caused by microphone-induced domain shift on various speech-based ML models. To facilitate this study, we collected two new speech datasets following the Replay-and-Record method-

ology. Our results show that this domain shift can degrade the performance of speech classification models such as Spoken Keyword Detection by as much as 15% drop in target domain accuracy. Even the state-of-the-art ASR models such as DeepSpeech2 and those offered by AI-as-a-Service providers are not immune to performance degradation. Further, we studied the domain shift caused by variations in IMU sensors and their placement on the performance of HAR models. Our experiments on the REALWORLD activity recognition dataset demonstrate the significant challenge that HAR systems face when they are deployed on body positions different from those on which they were trained.

In Chapter 4, we present an adaptation solution to counter microphone-induced heterogeneities in speech classification models. Later in Chapter 5, we consider the case of HAR models and present an unsupervised domain adaptation approach to scale them in the presence of device and placement induced domain shift.

Chapter 4

Scaling Opaque Machine Learning Systems

In this chapter, we propose an unsupervised domain adaptation approach to scale speech-based ML systems in the presence of domain shift induced by microphone heterogeneity. Our solution is grounded in a realistic scenario of ML deployment, which we call **opaque ML systems**. In an opaque ML system, we can feed data to the ML prediction model and obtain class predictions, however the parameters of the ML prediction model cannot be accessed or modified. This is a likely scenario when source model developers may want to keep their prediction model $f_S(.)$ private for commercial reasons; examples include ML offerings by AI-as-a-Service providers such as Google [134], Microsoft [135] and Amazon [136] which provide an API for users to upload their speech files and obtain speech transcripts in response.

4.1 Problem Setting

We are given a labeled dataset $\mathbb{X}_S = \{(\mathbf{x_{s_i}}, y_{s_i})\}_{i=1}^{N}$ from the source domain $\mathcal{D}_S$ where $\mathbf{x_{s_i}}$ is a sensor sample and y_{s_i} is the associated class label. For example, in the Multi-microphone Spoken Keywords dataset, x_{s_i} could be a speech spectrogram sample from the ReSpeaker microphone array and y_{s_i} could be the keyword class (e.g., Yes, No) corresponding to it. Given this labeled dataset, a neural network $f_S(.)$ could be trained using supervised learning to map the source domain samples to their corresponding labels. Thereafter, the model developer would like to deploy this trained neural network in a target domain $\mathcal{D}_T$ (e.g., a Matrix Voice microphone). From the quantification study presented in Chapter 3, we know that the hardware and software induced variations across microphones cause a domain shift between $\mathcal{D}_S$ and $\mathcal{D}_T$, which significantly degrades the accuracy of the source prediction model $f_S(.)$ in the target domain. Hence, our goal is to propose a solution that can counter the adverse effects of microphone-induced domain shift and improve the recognition

performance in the target domain.

4.2 Background and Related Work

In opaque ML systems, as we do not have access to the parameters of the source prediction model, we need a solution that can counter the domain shift between source and target microphones even before the data reaches the prediction model. Clearly, many of the domain adaptation solutions introduced in Chapter 2 do not apply here, as they relied on modifying model parameters to adapt a source domain model for use in a target domain. One possible solution to reduce the mismatch between microphone outputs is to calibrate the microphones with each other or with a common reference microphone using a frequency sweep signal [51]. However, to calibrate the frequency responses of microphones, we need to place both the source and target domain microphones in the same controlled environment (e.g., an anechoic chamber), which may not be feasible in practical settings. Another option is to leverage speech enhancement algorithms such as Minimum Mean-Square Error Short-Time Spectral Amplitude (MMSE-STSA) [91] to possibly reduce any channel-induced noise in the target domain dataset. We use it as a baseline in our evaluation and show that it does not satisfactorily alleviate the adverse effects of domain shift caused by microphone variations.

Finally, neural network-based approaches have been proposed for converting noisy speech to clean speech using denoising autoencoders [149, 150] and generative adversarial networks [92] . The core assumption in these techniques is the availability of paired or time-aligned data from clean and noisy conditions in order to learn a denoising neural network model. While it is easy to generate paired clean-noisy samples by augmenting clean speech with different types of ambient noise, it is impractical to assume the availability of paired speech data from the source and target microphones. The approach presented in this chapter is also based on a generative adversarial network, however, it does not assume the availability of paired speech samples from source and target microphones.

We formulate the problem of microphone variability as a data translation problem, i.e., given a speech segment recorded from a target microphone, can we translate it to the source microphone's domain? In other words, our goal is to learn a translation function $g : T \rightarrow S$, which can map a data sample from the target domain to the source domain. If a translation function can indeed be learned between source and target microphone domains, it can be subsequently used to reduce the domain shift caused by microphone variability.

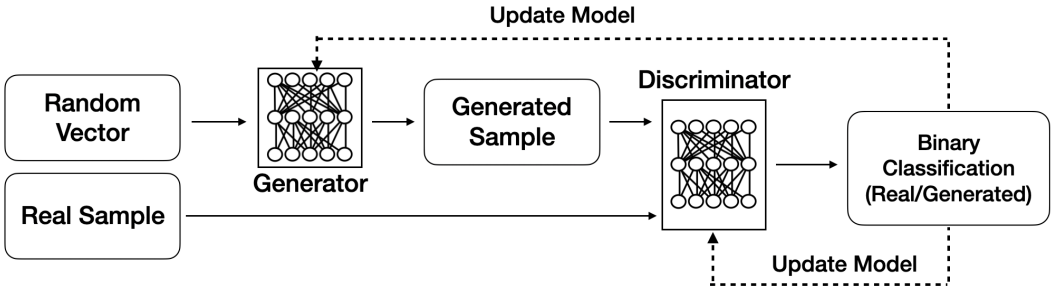


Figure 4.1: Generative Adversarial Network (GAN) proposed in [2].

4.2.1 Primer on Generative Adversarial Networks

Before presenting our solution, we provide a primer on generative adversarial networks (GANs), which form a core component of our proposed solution.

Generative Adversarial Networks. Deep learning has enabled many new applications of discriminative modeling, that is learning to predict a label y for a given input sample x. On the other hand, generative modeling is an unsupervised learning task to learn the patterns in a given dataset with an empirical distribution $p_{data}(\mathbf{x})$, such that the learned model can be used to generate new examples that plausibly could have been drawn from $p_{data}(\mathbf{x})$. For instance, in a keyword detection model, the task is to predict the probability of the presence of a keyword in a speech segment, and it is achieved in a *discriminative* fashion by choosing a keyword class with the highest output probability for the given audio segment. On the other hand, a generative modeling task here could be to generate an audio segment from a given keyword class.

Generative Adversarial Networks (GANs) proposed by Goodfellow et al. [2] are an approach to generative modeling using deep learning methods. As shown in Figure 4.1, a GAN consists of two neural networks, namely, a Generator (G) and a Discriminator (D), which compete against each other in an adversarial zero-sum game. The Generator G takes a random vector $\mathbf{z}$ often drawn from a Gaussian distribution as input and generates a data sample by evaluating $G(\mathbf{z})$. Besides random noise, other information can be fed into the Generator, in which case G is called a conditional generator. The Discriminator D on the other hand, is trained to distinguish between the real samples from $p_{data}(\mathbf{x})$ and the generated samples from G. In this way, the two neural networks G and D play a competitive game and in the process, both become better at their respective tasks: the Generator learns to generate data from $p_{data}(\mathbf{x})$, and the Discriminator becomes good at distinguishing data drawn from $p_{data}(\mathbf{x})$ vs. other data distributions.

Figure 4.2: A pair of grayscale and colored images, on which conditional GANs can be trained to learn a translation function.

Conditional GANs for data translation. GANs have also been extended for the task of data translation, particularly with images, where the Generator is conditioned on an input image[151]. Assume we want to learn a mapping or translation between two image domains, namely grayscale and colored, as shown in Figure 4.2. The GAN takes as input a *paired* set of images $(\mathbf{a}, \mathbf{b})$ from the two domains where $\mathbf{a}$ is a grayscale image and $\mathbf{b}$ is the corresponding colored image. The grayscale image $\mathbf{a}$ is fed to the generator in additional to the random vector $\mathbf{z}$, and the output $G(\mathbf{a}, \mathbf{z})$ is compared against the paired colored image $\mathbf{b}$ by the discriminator. D provides feedback to G on the likelihood of the generated data being drawn from the colored image domain. G uses this information to learn an even better mapping between grayscale and colored image domains. Eventually, once G is trained to convergence, it can be used as a translation function $G_{\text{grayscale}\rightarrow\text{colored}}$ to convert grayscale images into colored images.

Building on this general concept from the vision literature, Michelsanti et al. [92] proposed an approach of denoising speech samples; here instead of learning the $G_{\text{grayscale}\rightarrow\text{colored}}$ image mapping, they learned a $G_{\text{noisy}\rightarrow\text{clean}}$ speech mapping. However, as discussed earlier, these algorithms rely on the availability of paired data from the two domains. In our problem setting, collecting paired and time-aligned speech samples from different microphones would be infeasible in practice, particularly at scale. Hence, we seek an algorithm that can learn a translation function between microphone domains without paired supervision.

4.3 Mic2Mic: GANs with Cyclic Consistency for Speech Translations

We now describe our proposed solution, Mic2Mic, to learn a microphone translation function $g : T \rightarrow S$, which can map data instances from a target microphone to the source microphone's domain. More importantly, our aim is to learn the translation function without requiring paired data from the source and target domains.

4.3.1 Cyclic Consistency

Recall that the source domain contains labeled data while the target domain data is completely unlabeled. Let $\mathbb{X}_S = \{\mathbf{x_{s_i}}\}_{i=1}^N$ and $\mathbb{Y}_S = \{y_{s_i}\}_{i=1}^N$ denote a set of speech samples and their corresponding labels in the source domain $\mathcal{D}_S$, and $\hat{p}_s(\mathbf{x})$ denote the empirical marginal distribution defined by the source samples. Further, let $\mathbb{X}_T = \{\mathbf{x_{t_i}}\}_{i=1}^M$ denote a set of unlabeled speech samples in the target domain $\mathcal{D}_T$.

As discussed in the previous section, by training a conditional GAN, we can learn a mapping or translation function $G_{T\to S}$ to obtain $\hat{\mathbf{x}}_{\mathbf{t_i}} = G_{T\to S}(\mathbf{x_{t_i}})$, $\mathbf{x_{t_i}} \in \mathbb{X}_T$. This translation operation can induce a distribution over $\hat{\mathbf{x}}_{\mathbf{t_i}}$ that matches the empirical distribution defined by the source samples $\hat{p}_s(\mathbf{x})$, However, in the absence of any paired samples from the two domains (as is the case in our problem setting), this translation does not guarantee that each individual speech sample from the target domain is mapped to its corresponding equivalent in the source domain, because there are infinitely many translation functions $G_{T\to S}$ that can induce the desired distribution over $\hat{\mathbf{x}}_{\mathbf{t_i}}$.

In order to solve this challenge, we propose to impose the cycle-consistency property on the microphone translation function, as originally proposed by Zhu et al. [152] for visual recognition tasks. The core intuition behind cycle-consistency is that when an input sample $\mathbf{x_{t_i}}$ is translated from $T \to S$ and then back from $S \to T$, we should arrive at the same input sample. This can be achieved by learning two bijective translation functions $G_{T\to S}$ and $G_{S\to T}$, which are inverses of each other. In the next section, we present how to impose the cycle-consistency property during the adaptation process.

4.3.2 Mic2Mic architecture and training

In this section, we describe the architecture of Mic2Mic and the various losses that are optimized in the training process.

Training the source model. Let the source prediction model $f_S(.)$ be trained using supervised learning on the labeled source dataset by optimizing the cross-entropy loss between the model predictions and the ground truth labels.

$$\min_{f_S} \mathcal{L}_{\text{task}}\left(f_S, \mathbb{X}_\mathbb{S}, \mathbb{Y}_\mathbb{S}\right) = -\mathbb{E}_{(\mathbf{x_s}, y_s)\sim(\mathbb{X}_\mathbb{S}, \mathbb{Y}_\mathbb{S})} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log\left(f_S\left(\mathbf{x_s}\right)\right)$$

where K denotes the number of classes in the dataset.

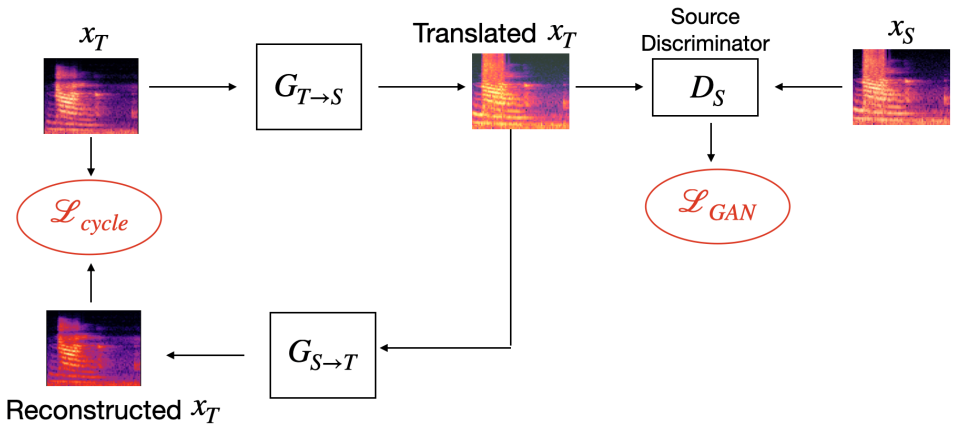


Figure 4.3: Architecture of Mic2Mic based on imposing cycle-consistency on data translations. The figure only shows the cycle with target data as input, the other cycle works similarly with subscripts S and T interchanged.

The weights of the source prediction model are frozen at this stage and are no longer updated in the rest of the training process. This aligns our problem setting of opaque ML systems where a model developer trains the source model and makes it available to user applications for inferences, but without revealing the model parameters.

Training the Mic2Mic translation model. Our proposed solution, Mic2Mic, consists of two generators $G_{T\to S}$ and $G_{S\to T}$, and two corresponding discriminators D_S and D_T. The *generators* networks follow the U-Net architecture[153] and consist of three convolutional layers for extracting higher-level features from spectrograms, three ResNet blocks for transforming the features from target domain to source domain, followed by three transpose convolutional layers for converting the transformed features into output spectrograms. Following the work of [152], we add skip connections[153] between the convolutional and corresponding transpose convolutional layers and use batch normalization layers between the ResNet blocks for faster convergence. For the *discriminator* models, a 4-layer deep fully-convolutional network is used with a batch normalization layer between two consecutive convolutional layers.

The training of Mic2Mic proceeds as shown in Figure 4.3. We first sample a batch of unlabeled data from the target and source domains and convert them to a mel-spectrogram representation. Mel-spectrograms from the target domain $\mathbf{x_t}$ are then fed to the generator $G_{\mathrm{T}\to\mathrm{S}}$, and the outputs $G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x_t})$ are evaluated using the source domain discriminator D_S which tries to separate them from the source domain spectrograms x_S. This is achieved by optimizing the **Adversarial Loss** as follows:

$$\mathcal{L}_{\mathrm{GAN}}\left(G_{\mathrm{T}\to\mathrm{S}}, D_S, \mathbb{X}_{\mathbb{S}}, \mathbb{X}_{\mathbb{T}}\right) = \mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_{\mathbb{S}}}\left[\log D_S(\mathbf{x_s})\right] + \mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_{\mathbb{T}}}\left[\log\left(1 - D_S\left(G_{\mathrm{T}\to\mathrm{S}}\left(\mathbf{x_t}\right)\right)\right)\right]$$

Next, to enforce cycle-consistency, the translated spectrograms $G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x_t})$ are fed to the reverse generator $G_{\mathrm{S}\to\mathrm{T}}$ to obtain the reconstructed spectrograms. By enforcing a L_1 penalty on the reconstruction error, we enforce the two generators to be inverse of each other. The **Cycle Consistency** loss can be expressed as:

$$\begin{aligned}\mathcal{L}_{\text{cycle}}\left(G_{\mathrm{T}\to\mathrm{S}}, G_{\mathrm{S}\to\mathrm{T}}, \mathbb{X}_{\mathbb{T}}, \mathbb{X}_{\mathbb{S}}\right) = \mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_{\mathbb{T}}}\left[\left\|G_{\mathrm{S}\to\mathrm{T}}(G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x_t})) - \mathbf{x_t}\right\|\right] \\ + \mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_{\mathbb{S}}}\left[\left\|G_{\mathrm{T}\to\mathrm{S}}(G_{\mathrm{S}\to\mathrm{T}}(\mathbf{x_s})) - \mathbf{x_s}\right\|\right]\end{aligned}$$

Finally, we use another loss term which encourages the generators $G_{\mathrm{T}\to\mathrm{S}}$ and $G_{\mathrm{S}\to\mathrm{T}}$ to be close to an identity mapping when they are fed samples drawn from X_S and X_T respectively. Known as the **Identity Loss**, this loss was originally proposed by Taigman et al. [154] and has shown to be beneficial in preserving the semantic properties of the data before and after translation. The identity loss for $G_{\mathrm{T}\to\mathrm{S}}$ can be expressed as:

$$\mathcal{L}_{\text{identity}}\left(G_{\mathrm{T}\to\mathrm{S}}, \mathbb{X}_{\mathbb{S}}\right) = \mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_{\mathbb{S}}}\left[\left\|G_{\mathrm{T}\to\mathrm{S}}(\mathbf{x_s}) - \mathbf{x_s}\right\|_{L_1}\right]$$

Taken together, these loss functions result in the following optimization objective for Mic2Mic:

$$\begin{aligned}&\min_{\substack{G_{\mathrm{T}\to\mathrm{S}}\\G_{\mathrm{S}\to\mathrm{T}}}}\ \max_{\substack{D_S\\D_T}} \mathcal{L}_{\text{Mic2Mic}}(\mathbb{X}_{\mathbb{S}}, \mathbb{X}_{\mathbb{T}}, G_{\mathrm{T}\to\mathrm{S}}, G_{\mathrm{S}\to\mathrm{T}}, D_S, D_T) \\ &\quad= \mathcal{L}_{\text{GAN}}\left(G_{\mathrm{T}\to\mathrm{S}}, D_S, \mathbb{X}_{\mathbb{S}}, \mathbb{X}_{\mathbb{T}}\right) + \mathcal{L}_{\text{GAN}}\left(G_{\mathrm{S}\to\mathrm{T}}, D_T, \mathbb{X}_{\mathbb{T}}, \mathbb{X}_{\mathbb{S}}\right) \\ &\quad+ \mathcal{L}_{\text{cycle}}\left(G_{\mathrm{T}\to\mathrm{S}}, G_{\mathrm{S}\to\mathrm{T}}, \mathbb{X}_{\mathbb{T}}, \mathbb{X}_{\mathbb{S}}\right) \\ &\quad+ \mathcal{L}_{\text{identity}}\left(G_{\mathrm{T}\to\mathrm{S}}, \mathbb{X}_{\mathbb{S}}\right) + \mathcal{L}_{\text{identity}}\left(G_{\mathrm{T}\to\mathrm{S}}, \mathbb{X}_{\mathbb{T}}\right)\end{aligned}$$

After optimizing the above objective, we are primarily interested in the Generator $G_{\mathrm{T}\to\mathrm{S}}$, which can take speech data collected from the target domain as input and translate them to the source domain. Note that the training of Mic2Mic is completely independent of the source prediction model $f_S(.)$ and hence satisfies the assumptions of an opaque ML system.

4.3.3 System Design

We now discuss the implementation of Mic2Mic as a component in the inference pipeline of ML systems. As shown in Figure 4.4, a speech sample collected from the target microphone during the inference stage is first converted to a mel-spectrogram representation. In the absence of Mic2Mic, the mel-spectrogram would have been

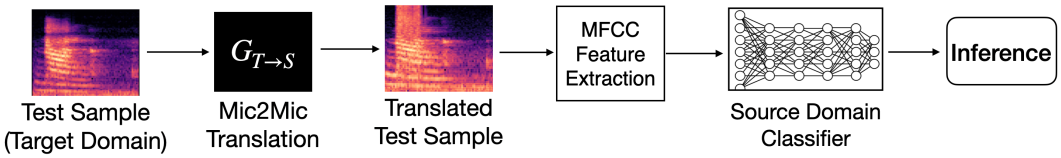


Figure 4.4: Integration of Mic2Mic's translation component ($G_{\mathrm{T}\to\mathrm{S}}$) in the inference pipeline of an opaque ML system in the target domain.

passed to the source domain classifier directly, resulting in poor inference performance.

Instead, with our proposed solution, the target domain mel-spectrogram is first fed as input to the trained Mic2Mic Generator $G_{\mathrm{T}\to\mathrm{S}}$, which translates it to its source domain equivalent. The translated spectrogram then optionally undergoes task-specific feature extraction, such as extracting MFCC features, and is then fed to the task-specific classifier from the source domain to output class labels. As many speech models (e.g., keyword detection [28], DeepSpeech2 [155]) use spectrograms or MFCC coefficients as input features, they are compatible with our current implementation.

In case no domain shift is detected in the system (e.g., source and target domain microphones are the same), the Mic2Mic translation component could be removed, or replaced with an identity function.

4.4 Evaluation

In this section, we evaluate the performance of Mic2Mic for speech classification tasks. Our findings show that Mic2Mic is able to effectively learn a microphone translation function using less than 30 minutes of unpaired speech data from the source and target microphones. Further, Mic2Mic can recover between 67% to 89% of the accuracy lost due to microphone variability for two speech classification tasks.

4.4.1 Tasks and Datasets

For this evaluation, we use three types of embedded microphones namely Matrix Voice, ReSpeaker and USB microphone. More details about the hardware specifications of these microphones were presented previously in Table 3.1 in Chapter 3.

Two representative speech classification tasks are used in our experiments, namely Spoken Keyword Detection (SKD) and Emotion Detection.

Spoken Keyword Detection. In this task, the goal is to identify the presence of a certain keyword class (e.g., Yes, No) in a given speech segment. To train a model for this task, the *Multi-microphone Spoken Keywords* dataset presented in Chapter 3 is used. Further, we use a small-footprint keyword detection architecture proposed in [28] to train the source domain classifier $f_S(.)$. The input to this model is a two-dimensional tensor extracted from a one-second-long keyword recording, consisting of time frames on one axis and 24 MFCC features on the other axis. The model outputs a probability of a given speech segment belonging to a certain keyword class (e.g., Yes, No) or to an Unknown class.

Emotion Detection. In this task, the goal is to identify the emotion of the speaker in a given speech segment. We record the *RAVDESS* dataset [156] on the three embedded microphones following the replay-and-record methodology introduced in Chapter 3. RAVDESS consists of 1440 speech files recorded by 24 actors where they expressed a range of emotions such as calm, happy, sad, angry, fearful, surprise, and disgust. The dataset is randomly split into training (75%) and test (25%) class-balanced subsets. We use a CNN-based speech emotion detection architecture proposed by [69] to train the source domain classifier $f_S(.)$.

Mic2Mic training data. In addition to training the task-specific models, we also need data to train Mic2Mic translation models. Ideally, we should ensure that the training data used for Mic2Mic does not overlap with the data used to train the task-specific models. To this end, we split the training datasets into two parts: 5% of the data is used to train the microphone translation models, while the remaining 95% is used to train the task-specific speech models.

Source Microphones	Target Microphones	Mean PSNR (Pre-translation)	Mean PSNR (Post-translation)
Matrix	ReSpeaker	20.51	28.08
	USB	26.48	28.56
ReSpeaker	Matrix	20.51	24.15
	USB	21.65	28.23
USB	Matrix	26.48	28.24
	ReSpeaker	21.65	24.21

Table 4.1: Comparison of PSNR between the spectrograms coming from two different microphones before and after Mic2Mic's translation operation.

4.4.2 Evaluation of the translation model

We first evaluate the efficacy of the learned translation model $G_{\mathrm{T}\to\mathrm{S}}$ in translating samples from a target domain to a source domain. In order to quantitatively evaluate the goodness of translated target samples, we need a ground truth in the source domain against which they can be compared. Recall that the data collection exercise described in Chapter 3 yielded speech datasets that are simultaneously recorded on multiple microphones – hence we can use the paired samples in the dataset to evaluate the translation model. It is important to clarify that paired samples are only used for evaluating the model; while training Mic2Mic we explicitly discourage any pairing between source and target datasets by randomly shuffling the datasets before each training epoch.

As the Mic2Mic translation model operates on 2D spectrograms, we use Peak Signal-to-noise ratio (PSNR) to compare target and source domain spectrograms pre- and post-translation. PSNR is a commonly used metric to calculate similarity between two images based on the mean square error between them. The higher the PSNR, the closer the images are to each other. We can employ the same metric to compute similarity between a spectrogram pair as follows:

$$PSNR\left(\mathbf{x_t}, \mathbf{x_s}\right) = 20.\, log_{10}\left(max\left(\mathbf{x_t}\right)\right) - 10.\, log_{10}\left(MSE\left(\mathbf{x_t}, \mathbf{x_s}\right)\right)$$

where MSE stands for the Mean Square Error between the two spectrograms and is computed as:

$$MSE\left(\mathbf{x_t}, \mathbf{x_s}\right) = \frac{1}{m.n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left[\mathbf{x_t}(i,j) - \mathbf{x_s}(i,j)\right]^2$$

Intuitively, we expect that the translation operation will lead to an increase in the PSNR between a pair of source and target microphone samples.

Results. Table 4.1 shows the pre- and post-translation PSNR averaged over 1000 spectrogram pairs randomly chosen from the Multi-microphone Spoken Keywords dataset. An apparent increase in the PSNR can be observed due to the translation operation, which suggests that Mic2Mic is able to reduce the domain shift between the source and target microphone data.

Further, Figure 4.5 presents a qualitative result to demonstrate the performance of ReSpeaker $\rightarrow$ Matrix translation model. In this figure, spectrograms *a* and *c* correspond to speech segments collected from ReSpeaker and Matrix microphones re-

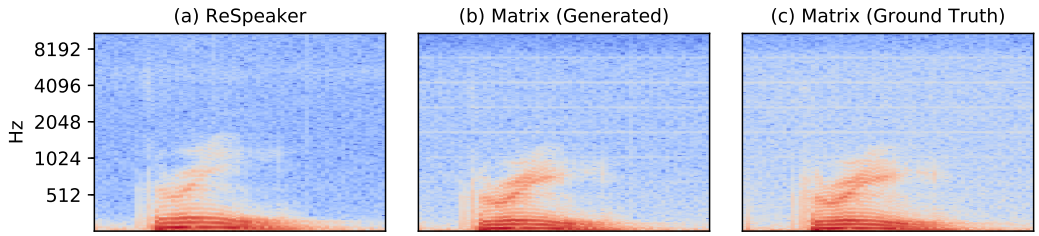


Figure 4.5: Performance of Mic2Mic for ReSpeaker→Matrix translation. (a) shows an example spectrogram from ReSpeaker and (c) shows the corresponding spectrogram for Matrix microphone. (b) shows the spectrogram generated by applying $G_{\text{ReSpeaker}\rightarrow\text{Matrix}}$ translation on (a).

spectively, whereas the spectrogram b is generated by applying $G_{\text{ReSpeaker}\rightarrow\text{Matrix}}$ translation model on spectrogram a. We observe that the generated spectrogram (b) is visually more similar to its corresponding ground truth (c), thereby qualitatively demonstrating that Mic2Mic managed to learn an effective translation function. Although this is a positive result, we note that the explainability of Mic2Mic remains poor and is a limitation that our approach inherits from the CycleGAN model on which it is based. For example, it is hard to pinpoint the exact transformations that Mic2Mic has learned to perform on the spectrograms in the time-frequency domain. We elaborate on this limitation in Section 4.5 and highlight some of the emerging works on understanding the internal representations of GANs.

4.4.3 Accuracy gains using Mic2Mic

After providing empirical evidence that Mic2Mic can reduce the domain shift between a given pair of microphones, we now present a series of experiments to evaluate the performance gains by adding Mic2Mic as a component in the ML inference pipeline.

Baselines. We compare the performance of Mic2Mic against three baseline inference pipelines:

- *Unmodified*: In this pipeline, the test data from the target domain microphone is directly passed to the source domain model without applying any modifications to it.
- *Speech Enhancement (SE)*: Before feeding the target test data to the source domain model, we first enhance its speech quality by applying a statistical speech enhancement technique known as Minimum Mean-Square Error Short-Time Spectral Amplitude (MMSE-STSA) estimation [91].

- *Calibrated*: An alternative approach to address microphone variability is to first measure the differences in frequency responses of source and target microphones in a controlled experimental setup and then negate those differences during the inference stage. While this approach of microphone calibration in controlled environments is not practical and scalable, we use it as a baseline to show how Mic2Mic compares with it in terms of performance. To compute a calibration offset between a pair of microphones, we play a frequency sweep signal and record it on both microphones in a non-reflective anechoic chamber.

  The power spectral density $R_i(f)$ of the microphone output as a function of frequency can be expressed as follows,

  $$R_i(f) = |H_i(f)|^2 . R_x(f)$$

  where $H_i(f)$ is the frequency transfer function of the i^{th} microphone and $R_x(f)$ is the power spectral density of the original frequency sweep signal.

  Thereafter, following the methodology proposed by [51], a per-frequency calibration offset $\Gamma_{S,T}(f)$ between the source and target microphones representing the difference in their frequency responses can be calculated as follows:

  $$\Gamma_{S,T}(f) = \frac{H_S(f)}{H_T(f)} = \sqrt{\frac{R_S(f)}{R_T(f)}}$$

  Finally, in the ML inference pipeline, the calibration offset Γ is applied on the data from the target microphone to compensate for the differences in frequency response between the source and target microphones. The corrected speech data is then fed to the source domain model for inference.

Results. In Figure 4.6, we show the test accuracy obtained by training and testing the Spoken Keyword Detection model on different microphones. For each microphone pair, we repeated the experiment five times and report the mean and 95% confidence intervals of the accuracy results.

It can be observed that when the model is trained and tested on the same microphone, we obtain the highest test accuracy as there is no accuracy degradation due to microphone-induced domain shift. However, when there is a mismatch between the training and test microphones, a significant accuracy loss is observed. For example, as shown in Figure 4.6a, when a model trained on Matrix Voice is deployed on ReSpeaker and USB microphones, there is an absolute accuracy drop of 12.4% and 6.7% respectively as compared to the accuracy obtained (78.8%) when the model is

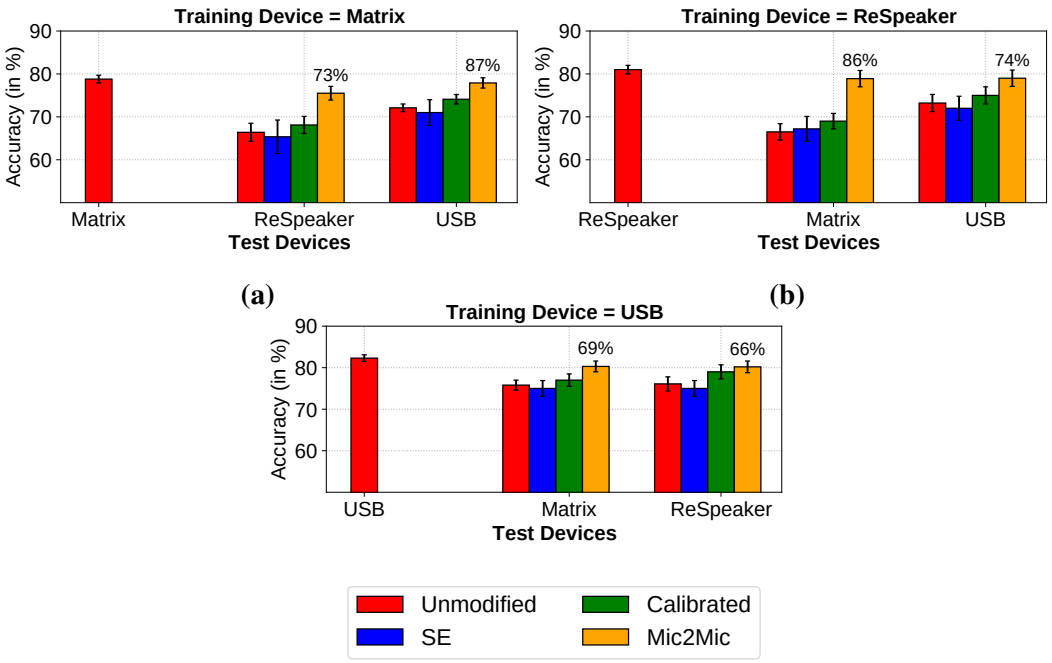


Figure 4.6: Accuracy of the Spoken Keyword Detection model under different scenarios of microphone variability. The numbers on the bars denote the percentage of accuracy recovered using Mic2Mic.

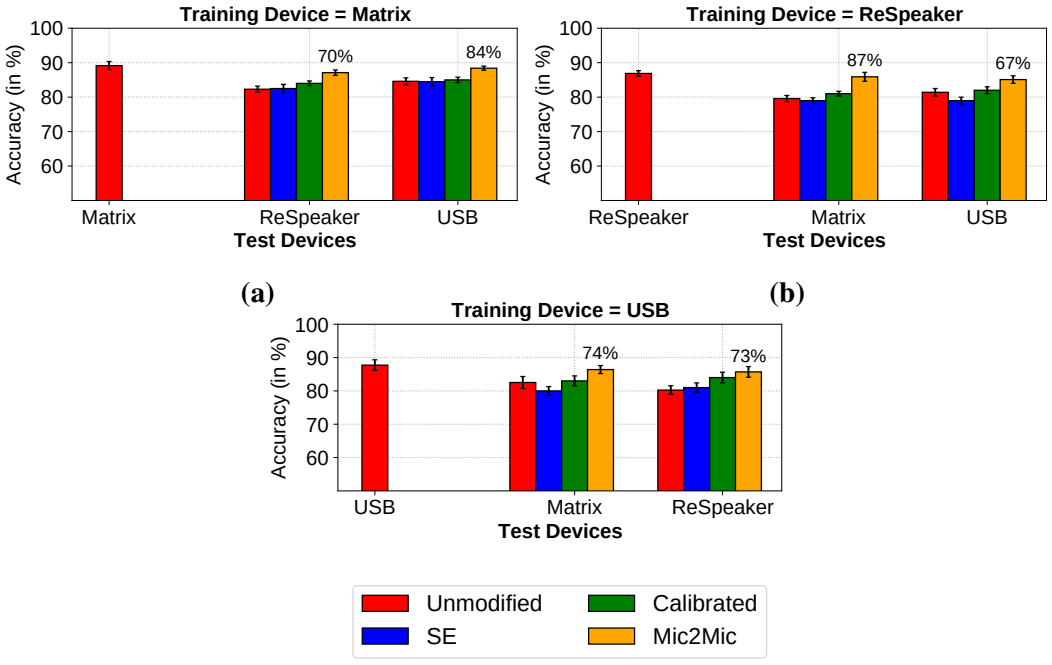


Figure 4.7: Accuracy of the Emotion Detection model under different scenarios of microphone variability. The numbers on the bars denote the percentage of accuracy recovered using Mic2Mic.

tested on Matrix Voice itself. By incorporating Mic2Mic in the inference pipeline of the test microphones, we can recover a significant proportion of this accuracy loss, i.e., 73% and 87% for ReSpeaker and USB microphones respectively. In doing so, Mic2Mic outperforms the two baseline approaches related to speech enhancement and microphone calibration. The microphone calibration baseline has the next best performance, and in one particular setting, i.e., USB→ReSpeaker, it provides similar performance as Mic2Mic.

In Figure 4.7, for the Emotion Detection task, we observe accuracy drops between 4%-9% in scenarios of microphone variability, and by adding Mic2Mic as a translation component in the inference pipeline, we can recover between 67-87% of the lost accuracy.

Training with multiple microphones. While the above experiments use data from just one microphone to train the source domain speech models, in practice developers will have access to data from diverse microphones. We now consider the scenario when the source domain model is trained with multiple microphones and evaluate if there is still any accuracy degradation due to microphone variability.

In Tables 4.2 and 4.3, we present the test accuracy when the source domain models are trained on data from multiple training microphones. Once trained, the models are evaluated in the source domain (i.e., on test data from the microphones on which the model was trained) and the target domain (i.e., the microphone which did not contribute to the training data). Even here, we observe an accuracy degradation in the target domain, e.g., the test accuracy goes from 82.35% to 74.7% when the Spo-

	Training Microphones		
	Matrix and ReSpeaker	**ReSpeaker and USB**	**USB and Matrix**
Source Microphones	82.9	78.63	82.85
Target Microphone	72.6	67.1	74.7
Target Microphone with Mic2Mic	80.6	77.1	80.4

Table 4.2: Test accuracy of the Spoken Keyword Detection model when it is trained on multiple microphones.

	Training Microphones		
	Matrix and ReSpeaker	**ReSpeaker and USB**	**USB and Matrix**
Source Microphones	87.35	89.35	91.3
Target Microphone	83.6	84.1	85
Target Microphone with Mic2Mic	86.4	87.8	89.3

Table 4.3: Test accuracy of the Emotion Detection model when it is trained on multiple microphones.

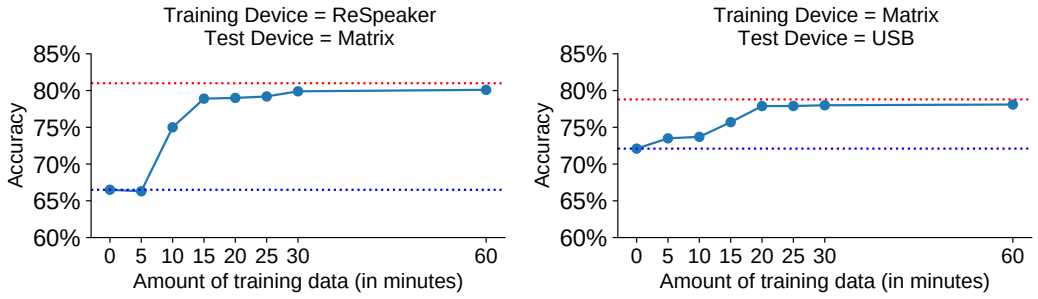
ken Keyword Detection model trained on USB and Matrix microphones is deployed on ReSpeaker. Finally, the last rows of Tables 4.2 and 4.3 show that by incorporating Mic2Mic in the inference pipeline of the target domain, we can recover a large proportion of the accuracy that is lost due to microphone heterogeneity.

4.4.4 How much data is needed to train Mic2Mic?

In this section, we present results on Mic2Mic's performance as the amount of data used to train the translation model is varied. As discussed earlier, training Mic2Mic only requires unlabeled and unpaired data from the source and target microphones. While collecting such data is indeed cheap, it will be ideal if the Mic2Mic translation model can be trained with minimal amount of data.

For this experiment, we systematically vary the amount of unpaired data available to train Mic2Mic's translation model. We gradually increase the unpaired data from the source and target microphones from 0 minutes to 60 minutes. Once a translation model is trained in each data configuration, we incorporate it in the inference pipeline of the Spoken Keyword Detection model in the target domain and evaluate the test accuracy.

Figure 4.8 illustrates the findings of this experiment for two different microphone combinations. The dotted red line shows the upper-bound test accuracy when the Spoken Keyword Detection model is tested on the same microphone on which it was trained. The dotted blue line shows the baseline accuracy when the model is deployed on a test microphone without incorporating Mic2Mic in the inference



(a) **(b)**

Figure 4.8: Accuracy of the Spoken Keyword Detection model in the test (or target) domain, as the amount of unlabeled data used to train the Mic2Mic translation model is varied.

pipeline. We observe that as more *unpaired* training data is supplied, the Mic2Mic translation model becomes better, and in turn, the inference performance of the Spoken Keyword Detection model increases. The inference performance plateaus around 30 minutes in Figure 4.8(a) and around 20 minutes in Figure 4.8(b), suggesting that Mic2Mic is able to learn a good translation model with less than 30 minutes of unpaired data in both scenarios.

4.5 Discussion and Limitations

In this section, we discuss the limitations of this work and outline the avenues for future work on this topic.

Explainability. A limitation of this work comes from the lack of explainability of the translation model. While we empirically demonstrated that the translation model is able to reduce the domain shift between microphone datasets and improve the downstream classification accuracy in the target domain, it remains unclear what exact transformations does Mic2Mic learn in the time-frequency representation of speech data. Recently, methods have been proposed [157, 158] to visualize and understand the internal representations of a generative adversarial network in the context of images. Such works could be extended for CycleGAN-based architectures such as Mic2Mic to uncover the kinds of time-frequency transformations being learned during the training process.

Scalability of Pairwise Translations. Another limitation of our current implementation is that we learn pairwise translations between training and test microphones. Hence, in order to scale this solution to a large number of devices, multiple such models need to be learned. Future work can investigate the development of a common microphone translation model that can be applied to any given pair of mi-

crophones. A similar solution in the image-to-image translation domain has been proposed, namely StarGAN [159], which uses an n-dimensional one-hot vector to encode the input and output image labels into the translation model architecture.

Scaling to other heterogeneities. We focused on addressing the composite effect of heterogeneities introduced by the hardware and software processing components of the microphone in speech data. However, in practice, speech-based models will encounter additional forms of heterogeneity such as variations in background noise or speaker accents, which can overlap with the already challenging scenario of microphone heterogeneity. As future work, we plan to explore the applicability of learning a domain translation function when more heterogeneities co-occur in the speech data.

Deployment Overhead. Mic2Mic adds an extra step of data translation in the inference pipeline of speech models. While it improves the prediction accuracy in the target domain, it also increases the latency of computing inferences. This could pose a challenge for some ML systems that aim to provide real-time inferences to users. Future work should explore optimizing the translation model to reduce its latency and power consumption, particularly on embedded devices. As our translation model consists of a number of convolution layers, techniques proposed for optimizing CNNs on embedded devices could be employed [160].

Other Related Works. Contemporaneous to our work, [161] applied a similar idea in the field of medical imaging. They found that X-ray images captured by different machines have substantial domain shift, and training a CycleGAN-based translation component can counter this shift. [162] is a follow-up work to ours, which aims to adapt speech recognition systems trained on speech recorded from telephones (source domain) to speech recorded on camcorders in-the-wild (target domain). [163] proposed to use the $G_{\mathrm{S}\to\mathrm{T}}$ mapping learned with a CycleGAN to map synthetic images (source data) to real images (target domain). Thereafter, they trained a model for the target domain using supervised learning on the translated source data and source labels. This approach however requires source domain labels during adaptation, which is a very strong assumption for opaque ML systems.

4.6 Summary

With the goal of scaling speech-based ML systems to new domains, we presented an adaptation solution, Mic2Mic, which reduces the shift between source and target domains by adding a data translation component in the inference pipeline in the

target domain. This translation component is trained on unlabeled and unpaired data from the source and target domains using a Cyclic Generative Adversarial Network and is able to recover up to 87% of the accuracy degradation in the target domain for two speech classification tasks. More importantly, by virtue of its design, Mic2Mic does not require modifying the parameters of the source domain prediction model, which makes it appropriate for use in an opaque ML system.

We also identified several limitations of our proposed solution in Section 4.5. In particular, a key limitation is the additional overhead that comes with adding Mic2Mic in the inference pipeline, which in turn could increase the latency of computing inferences on the data. In Chapter 5, we explore an adaptation solution wherein instead of adding an additional component in the inference pipeline, we directly modify the parameters of the source prediction model during adaptation in order to improve its performance in the target domain. This solution is presented in the context of human-activity recognition (HAR) systems that operate on inertial sensor data.

Chapter 5

Scaling Transparent Machine Learning Systems

The adaptation solution presented in Chapter 4 was premised upon the assumption that we do not have access to the parameters of the source prediction model during the adaptation process, which is a likely scenario in many commercial and proprietary ML systems. In this chapter, we focus on **transparent ML systems** wherein we can access and modify the parameters of the source prediction model during the adaptation process. Such a scenario can occur when either the prediction model is publicly known (e.g., downloaded from a public model repository) or when the source model developer themselves would like to adapt the model parameters to new domains.

A good example of the latter scenario is the task of Human Activity Recognition (HAR) based on inertial data collected from Inertial Measurement Unit (IMU) sensors placed on the human body. Imagine that a model developer collects a large amount of labeled data from IMU sensors on a smartphone placed in the user's thigh pocket (*source domain*), where the labels represent the physical activities of the users (e.g., running, walking, sitting). A prediction model can be trained on this dataset using supervised learning and then deployed through an HAR application installed on the smartphone. At test time however, the end-users may place their smartphones at body positions different from the training body position (i.e., thigh) – for instance, the smartphone can be placed in the chest pocket or inside an armband when the user is running. Moreover, the HAR application might be installed on smart devices containing different IMU hardware than the training smartphone. As we demonstrated in Chapter 3, due to device and placement induced heterogeneity, the distribution of the IMU test data will differ from that of the training data, and the model developer will need to adapt their HAR model to prevent any performance degradation in the target domain.

In this chapter, we present an unsupervised domain adaptation solution for scaling transparent ML systems and focus our evaluation on HAR models trained using inertial sensor data[1]. However, the general problem of scaling transparent ML models using is also applicable to other sensor modalities and heterogeneities. For example, the microphone heterogeneity problem presented in the previous chapter can also manifest in transparent ML systems, and the model developer can choose to adapt the weights of the source microphone model to counter it. While concluding this chapter, we briefly discuss how the solutions presented for HAR models could be extended to speech-based ML systems.

5.1 Problem Setting

Following the notations presented in Chapter 2, let $\mathcal{D}_S$ denote a source domain (e.g., 'thigh-worn IMU') and $\mathcal{D}_T$ be a target domain (e.g., 'wrist-worn IMU'). In the source domain, we are given a set of sensor samples $\mathbb{X}_S = \{(\mathbf{x_{s_i}}, y_{s_i})\}_{i=1}^{N}$ and labels $\mathbb{Y}_S = \{y_{s_i}\}_{i=1}^{N}$ where y_{s_i} is the physical activity class label associated with the corresponding sensor sample $\mathbf{x_{s_i}}$. Let $\mathbb{X}_T = \{\mathbf{x_{t_i}}\}_{i=1}^{M}$ be an unlabeled dataset, where $\mathbf{x_{t_i}}$ is a sensor sample from the target domain. No labeled observations are available in the target domain.

Further, let $t_S(.)$ be a task prediction model trained using supervised learning to minimize the empirical risk in the source domain. The task prediction model t_S can be decomposed into two components: $f_S(.)$ and $g_S(.)$, where $f_S : \mathcal{X} \rightarrow \mathcal{Z}$ represents a feature extractor that maps samples from the source domain into a latent feature space $\mathcal{Z}$. Similarly, $g_S : \mathcal{Z} \rightarrow \mathcal{Y}$ represents a task classifier that maps the source features to output labels. In the context of deep learning, $f_S(.)$ and $g_S(.)$ are parameterized with deep neural networks F_S and G_S respectively. For a K-way classification task, the optimization objective for F_S and G_S can be expressed as follows:

$$\min_{F_S, G_S} \mathcal{L}_{\text{sup}} = -\mathbb{E}_{(\mathbf{x_s}, y_s)\sim(\mathbb{X}_\mathbb{S}, \mathbb{Y}_\mathbb{S})} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \left[\log\left(G_S\left(F_S\left(\mathbf{x_s}\right)\right)\right)\right]$$

As we quantified in Table 3.4, due to the domain shift caused by changes in the

[1] A part of this work is published in [56] in which Youngjae Chang and I are the co-primary authors. Both of us had an equal contribution in implementing the solutions and coming up with the experiment design. However, in [56], we only experimented with the domain shift in accelerometer data. In this chapter, I focus on HAR models trained with both accelerometer and gyroscope data, which is a more common practice in the literature. As such, I re-implemented the code and retrained all the HAR models. Different from [56], I present new experiments and analysis on countering domain shift when both accelerometer and gyroscope data are used to train HAR models.

sensor hardware, sensing pipeline and sensor placement on the human body, feature encoder F_S and classifier G_S trained on the source domain are not optimal for doing predictions in the target domain. Hence, a solution is needed which can modify the parameters of the source neural networks F_S and G_S to improve their classification performance in the target domains.

5.2 Background and Related Work

As we discussed in Chapter 2, a number of works have tackled the problem of data heterogeneity in HAR models. To counter device-induced heterogeneities, [45] proposed the idea of automatically clustering IMU devices based on the heterogeneities in their sensor outputs. Thereafter, an activity recognition model is trained for each cluster of devices on their aggregated labeled data, and it is used as the representative model for the entire cluster during inference. A major limitation of this solution pertains to its lack of scalability; firstly, this solution requires collection of labeled data from every new device on which the HAR system is deployed, which is expensive and time-consuming. Moreover, for each new device, the clustering step needs to be repeated and followed up by retraining the HAR model for the cluster to which the device is assigned.

With regards to IMU placement-induced domain shifts, a widely used approach is to train placement-specific HAR models [114, 115], which are found to be more accurate than placement-independent models such as [116]. However, this approach also requires collecting labeled data from each body position where the sensor is likely to be placed, thereby making it costly and less practical.

In recent years, researchers have started applying unsupervised domain adaptation (uDA) based approaches in this problem space. One of the early works in this direction was HDCNN [131], which studied the transfer of HAR models from a smartphone (source domain) to a smartwatch (target domain). This solution is inspired by the idea of feature alignment proposed in [119]. Here, a target domain model is trained by minimizing the layer-wise Kullback-Leibler (KL) divergence between the activations of source and target models. The proposed approach is evaluated on a private dataset and showed promising results. We build upon this work in three ways: (i) we propose that in order to learn effective domain-invariant features, there is a need for principled data augmentation before performing the feature alignment, (ii) we use a different approach of adversarial training to align the feature representations (iii) we evaluate our solution on a much larger public dataset containing seven different sensor placements and show that it outperforms the DDC baseline

which inspired this prior work. Further, contemporaneous to our solution, Akbari et al. [132] extended the HDCNN work by replacing the deterministic features with a stochastic feature extractor. Once the features are extracted, they are aligned across domains by minimizing the KL divergence between them. A limitation of this approach is that it assumes the availability of paired or time-aligned IMU samples from the source and target domains for performing uDA. This is a very strong assumption, which can be easily violated in practical deployment settings. Instead, our solution does not enforce any pairing or time-alignment between source and target IMU samples.

5.3 Scaling HAR Models with Data-Augmented Adversarial Training

In this section, we present our solution to scale HAR models to target domains which have both device and placement-induced domain shift when compared to the source domain. Our proposed approach is built upon two promising techniques in the field of deep learning, namely data augmentation and adversarial learning.

5.3.1 Solution Overview

The core idea of our proposed solution is to modify the weights of the source domain feature extractor F_S such that it extracts domain-invariant feature representations of the source and target domain data. More specifically, we would like the feature representations of data to be invariant to the sensor hardware and the placement of the IMU sensor on the human body. If we can successfully adapt F_S to map source and target data to domain-invariant features, then the task classifier G_S could be further adapted to map the domain-invariant features to output labels by performing supervised learning on the labeled source data.

Inspired by previous works in the uDA literature [123, 164], we leverage adversarial training to adapt F_S and encourage it to extract domain-invariant features from the data. Moreover, to ensure that device-induced heterogeneities are adequately represented in the source and target datasets, we propose augmenting the source and target datasets by adding specific types of perturbations that are expected due to hardware [43, 50] and software-related factors [45]. Our results in Section 5.4.3 demonstrate that performing the proposed data augmentation in a principled way before adversarial training can significantly boost the performance of the model in the target domain.

5.3.2 Solution Description

Figure 5.1 shows the proposed architecture for Data-Augmented Adversarial Learning, and we now describe its constituent components.

Data Augmentation. In order to simulate device-induced heterogeneities in the IMU data, we use three data perturbation schemes, as described below.

- *Scaling*: Under this scheme, we scale each axis in the accelerometer and gyroscope data by a random coefficient c sampled from a normal distribution $\mathcal{N}(1, \sigma^2)$. Here σ is a hyperparameter that controls the magnitude of the scaling coefficient. This perturbation relates to the scale factor errors [165, 50] in inertial sensor data, which often occur due to temperature differences between the initial calibration and operation stages of the sensor.
- *Axis rotation:* Even when inertial sensors are used in a fixed body position, sensor orientations might vary between multiple wearing sessions; for example, Min et al. [166] showed that data from ear-worn IMU devices could vary between wearing sessions depending on how the device is placed inside the ear. To simulate this variability, we perform a rotation of both the accelerometer and gyroscope data with a random roll, pitch, and yaw.
- *Time-warping:* This scheme simulates deformation in the time axis; it first adds a random perturbation to the time intervals between consecutive data samples, and then re-samples each axis of the data with the original sampling rate using 1-D linear interpolation. This perturbation is related to sampling rate instability in inertial data that occurs due to delays in the attachment of timestamps to sensor readings by the operating system [45]. We use the time-warping implementation provided by the authors of [167].

Figure 5.2 illustrates the effect of the three perturbation schemes on a 3-second long accelerometer sample from the REALWORLD dataset (sampling rate = 50Hz).

During training, we first sample a batch of IMU data each from the source and

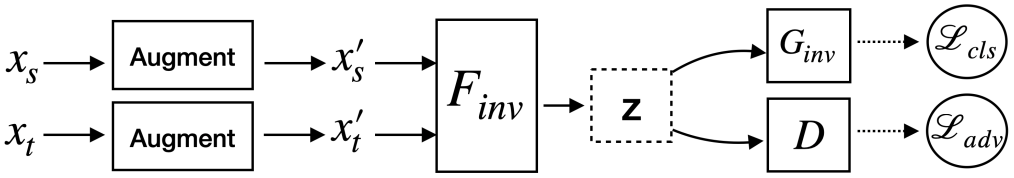


Figure 5.1: Architecture for Data-Augmented Adversarial Training

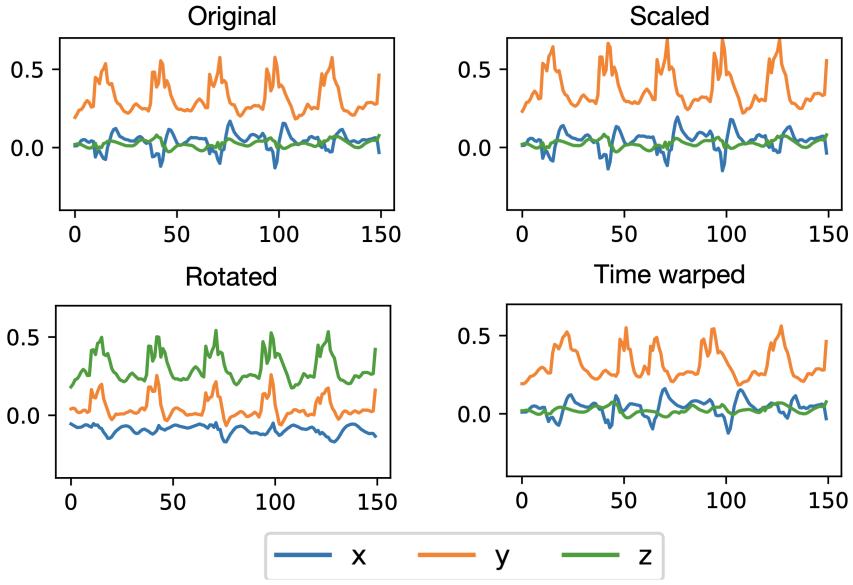


Figure 5.2: Effect of various data perturbation schemes on a 3-second long accelerometer segment from the REALWORLD HAR dataset.

target body positions, denoted by $\mathbb{B}_S$ and $\mathbb{B}_T$, respectively. Between these two data batches, there already exists domain shift induced by variations in the placement of the IMU sensor. To introduce device-related heterogeneities in the data, we perturb both $\mathbb{B}_S$ and $\mathbb{B}_T$ using the perturbation schemes described above. A perturbed batch of data is finally concatenated with the original batch to generate an augmented data batch. Let us denote the augmented data batches in source and target domains as $\mathbb{B}'_S$ and $\mathbb{B}'_T$ respectively.

Adversarial Feature Learning. After constructing the augmented batches of data containing both device and placement-induced heterogeneities in IMU data, we employ adversarial learning to extract domain-invariant feature representations from the source and target domains. This is achieved by assigning a *domain label* to data from each domain and training a domain discriminator, D, to classify whether a sample is drawn from the source or the target domain. The optimization objective of D can be written as:

$$\min_D \mathcal{L}_{adv_D} = -\mathbb{E}_{\mathbf{x_s}\sim\mathbb{B}'_S}[\log(D(F_{\text{inv}}(\mathbf{x_s}))] - \mathbb{E}_{\mathbf{x_t}\sim\mathbb{B}'_T}[\log(1-D(F_{\text{inv}}(\mathbf{x_t}))] \quad (5.1)$$

Here F_{inv} is the domain-invariant feature extractor that we would like to learn. F_{inv} in initialized with the weights of the source feature encoder F_{S} and its optimization objective is opposite to that of the domain discriminator. More specifically, F_{inv} aims to confuse the domain discriminator by generating features that are invari-

ant to source and target domains. This objective is shown in Equation 5.2 and is implemented by inserting a Gradient Reversal Layer (GRL) [122] between the feature encoder and the discriminator. During backpropagation, the Gradient Reversal Layer multiplies the gradients of the domain discriminator by a negative constant (-1 in our case) and passes them to the feature encoder F_{inv} to update its weights.

$$\min_{F_{\text{inv}}} \mathcal{L}_{adv_F} = -\mathcal{L}_{adv_D} \tag{5.2}$$

Looking at Equations 5.1 and 5.2, we can observe that the domain discriminator and feature encoder are *adversaries* of each other, tasked with competing objectives: while D aims to effectively discriminate data from the source and target domains, F_{inv} aims to learn domain-invariant features from the source and target data such that D cannot distinguish them. They play this adversarial game of trying to defeat the other, and in the process, both become better at their respective tasks. More importantly, the feature extractor F_{inv} becomes good at mapping samples from source and target domain to a domain-invariant feature representation, thereby reducing the domain shift.

Finally, to ensure that the features extracted by F_{inv} remain meaningful for the goal of HAR, we add the supervised cross-entropy loss to the overall optimization objective as follows:

$$\min_{G_{\text{inv}},F_{\text{inv}}} \mathcal{L}_{\text{task}} = -\mathbb{E}_{(\mathbf{x_s},y_s)\sim(\mathbb{B}'_S)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log\left(G_{\text{inv}}\left(F_{\text{inv}}\left(\mathbf{x_s}\right)\right)\right) \tag{5.3}$$

where G_{inv} is the adapted classifier that maps the invariant feature representations to the class labels. G_{inv} is initialized with the source domain classifier G_{S} and is optimized using the cross-entropy loss shown in Eq. 5.3.

Taken together, the overall optimization objective of our proposed solution can be written as:

$$\min_{D} \min_{\substack{F_{\text{inv}} \\ G_{\text{inv}}}} \mathcal{L}_{\text{optim}} = \mathcal{L}_{adv_D} + \alpha\mathcal{L}_{adv_F} + \beta\mathcal{L}_{\text{task}}$$

where α and β are hyperparameters that control the relative weight of each loss in the adversarial training process.

5.4 Evaluation

In this section, we present an evaluation of our proposed approach.

5.4.1 Experiment Setup

We begin by describing the dataset and pre-processing operations done on the data. We then proceed to discuss the architecture of the neural networks employed for the HAR task.

Dataset. Similar to the quantification experiments in Chapter 3, we use the REALWORLD HAR dataset [1] for our experiments. This dataset contains IMU data recorded from 15 participants performing 8 activities: *climbing stairs down and up, jumping, lying, standing, sitting, running/jogging, and walking*. As shown in Figure 3.4, each participant was instrumented with various IMU-equipped smartphones and smartwatches, attached to seven different body positions: head, chest, upper arm, waist, forearm, thigh, and shin.

Data characteristics and pre-processing. The accelerometer and gyroscope traces are segmented into time windows of 3 seconds, without any overlap. This window length was chosen empirically to align with the duration of various human activities in the dataset[2]. If a 3-second-long trace includes an activity transition, timestamp noise, or data points without labels, the trace gets discarded. The whole dataset is normalized to be in the range of -1 and 1. As shown in the IMU processing pipeline in Figure 2.2, the accelerometer and gyroscope axis are stacked on top of each other to create a 2-D representation of the time-series data. Finally, we use stratified splitting to divide the dataset into two parts: training set (75%) and test set (25%).

Model Architecture. We design a convolutional neural network (CNN) model based on the work by Hammerla et al. [79] and Almaslukh et al. [145]. The model consists of two components: a feature extractor and a classifier. The *feature extractor* is a 6-layer deep CNN with temporal (1-D) convolutional layers. We use LeakyReLU activations [146] with $alpha = 0.3$ and instance normalization [147] layers between convolutional layers for faster convergence. We also employ dropout regularization to avoid overfitting. The feature extractor takes as input a 3-second frame of pre-processed IMU data from a given body position and outputs a 150-dimensional feature vector. This feature vector is then passed as input to the *clas-*

[2]We also experimented with one and two second long windows, however, there was no significant different in HAR model performance. Therefore, we decided to use three-second windows, because longer windows reduce the frequency of ML model executions at inference time.

sifier, which consists of two fully-connected layers and generates a K dimensional output where K is the number of activity classes (e.g., sitting, walking). For the REALWORLD dataset, $k = 8$.

Our HAR architecture differs from [168, 145] in two ways. Firstly, all layers in the feature extractor are convolutional layers. We replaced the traditional max-pooling layers with convolutional layers with a large stride based on prior research [169], which showed that this simple substitution can speed up the training process without any significant loss in accuracy. Secondly, instead of flattening the outputs of the final convolutional layer, we apply Global Average Pooling on them to obtain the feature vector – this results in reducing the number of parameters to be learned and avoids overfitting on the source data. A baseline result on REALWORLD dataset (see table 3.4) shows that with this modified architecture, the HAR performance is on par with [145] while having the advantage of smaller model size.

Training process and mini-batch generation for uDA. The source domain model is trained by optimizing the categorical cross-entropy loss. We use a mini-batch size of 64, which is chosen based on a hyperparameter search in {32,64,96,128}. Further, we employ the Adam optimizer [148] with a learning rate 1e-3, which is obtained by doing a hyperparameter search on {1e-2, 5e-2, 1e-3, 5e-3, 1e-4, 5e-4}. For adversarial training, we follow the same methodology and got a batch size of 32 and learning rates of 1e-3 and 1e-4 for the discriminator and feature extractor, respectively. We use importance-weighted cross-validation [170, 171] to select these hyper-parameters. The model is implemented in TensorFlow 2.0 and trained on a NVIDIA Tesla V100 GPU.

5.4.2 Baselines

Our proposed solution is compared against four baseline approaches listed below:

- *Source Only.* This is the simplest baseline where an HAR model is trained on the labeled source dataset and tested on the target domain test set without performing any data augmentation or adversarial learning on it.
- *Data Augmentation training.* In this baseline, we apply the three proposed data augmentation schemes on the labeled source data, and train a source domain model on the augmented dataset. As shown in prior works in visual [172] and speech recognition [173], data augmentation can improve the generalizability of deep neural networks. As such, we evaluate how well does data augmentation perform when scaling HAR models to new domains.

- *Deep Domain Confusion (DDC).* This is a uDA technique [119] to enforce invariance between source and target feature representations by minimizing the Maximum Mean Discrepancy (MMD) between them. Although it has been primarily studied for visual recognition tasks, it can be implemented for HAR by applying the *MMD loss* on the outputs of the feature extractor F_{inv}. Since DDC is not based on adversarial learning, a domain discriminator is not needed. In addition to minimizing the MMD loss between source and target feature representations, we also optimize the supervised task loss $\mathcal{L}_{\text{task}}$ presented in Equation 5.3.

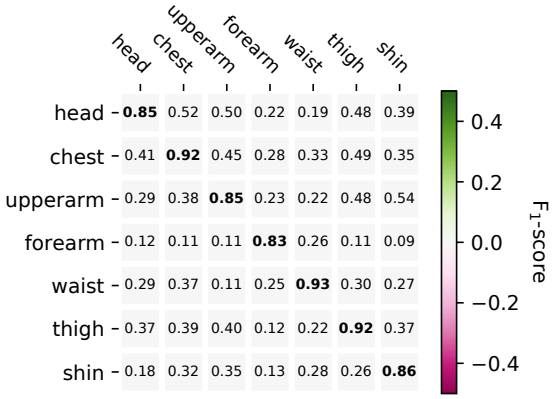
- *Domain-Adversarial Neural Networks (DANN).* This is the adversarial domain adaptation approach [122], which is extended by our proposed solution. In this baseline, no data augmentation is done on the source and target samples, and they are directly fed into the adversarial training architecture.

5.4.3 Results

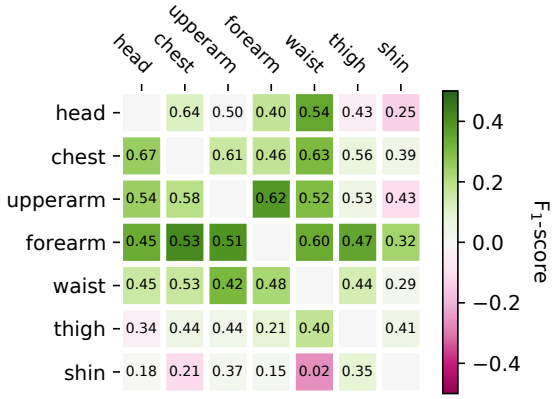
We now present our experimental findings on the REALWORLD dataset. Before presenting in-depth results in the form of heatmaps, we explain how to read and interpret them.

Interpreting the heatmaps. As shown in Figure 5.3, each heatmap is composed of cells, and each cell denotes an experimental setting. The body position written on the left of the cell indicates the source body position with a labeled dataset. The body position written on the top of the cell indicates the target body position with an unlabeled dataset. Each cell represents an experiment setting when a model trained in the source domain is tested on data from target domain, under different training and adaptation techniques. Delving deeper, each cell has two attributes. First, the number inside the cell denotes the class-weighted F_1 score obtained on a test set from the target domain. Secondly, the color of each cell denotes the increase/decrease in F_1 score compared to the *Source Only* baseline (where no adaptation or augmentation is applied to the source prediction model). Cells colored 'green' illustrate that F_1 score in the target domain increase due to adaptation or augmentation, while a 'pink' colored cell demonstrates a decrease in the F_1 score over the *Source Only* baseline. The color intensity represents the magnitude of change in the F_1 score.

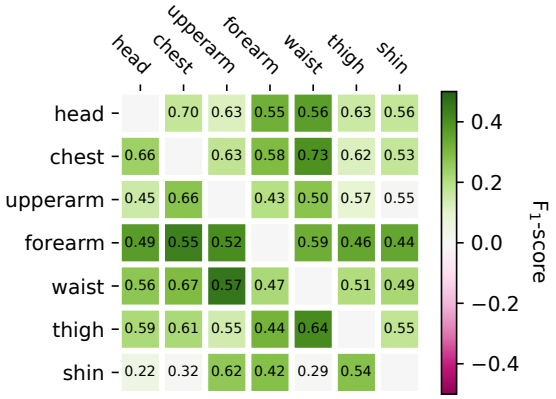
Adaptation Performance. In Figure 5.3a (diagonal cells), we report the performance of HAR models when they are trained and tested on the same body position. This represents the ideal scenario with no presence of domain shift; thereby, we expect high classification accuracies. For instance, a model trained on *head* and *chest*



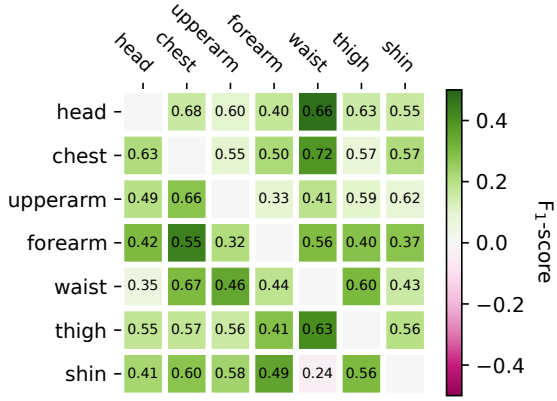
(a) Source-Only



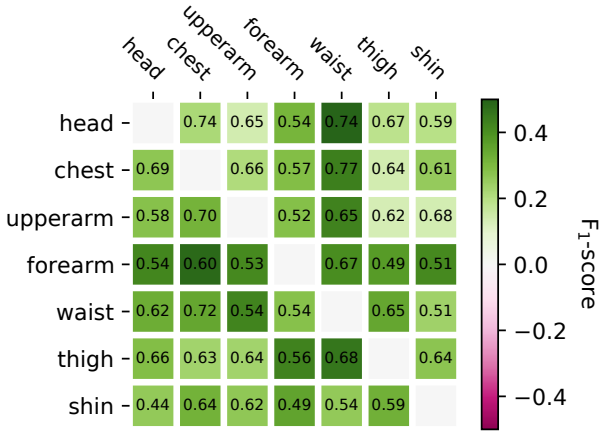
(b) Data Augmentation



(c) Deep Domain Confusion (DDC)



(d) Domain-Adversarial Neural Networks (DANN)



(e) Our proposed solution: Data-Augmented Adversarial Learning

Figure 5.3: Performance (F_1 scores) of HAR classifiers trained and tested on data from different body positions with various training approaches. Our proposed approach of data-augmented adversarial training outperforms other baselines in 41 out of the 42 experiment conditions.

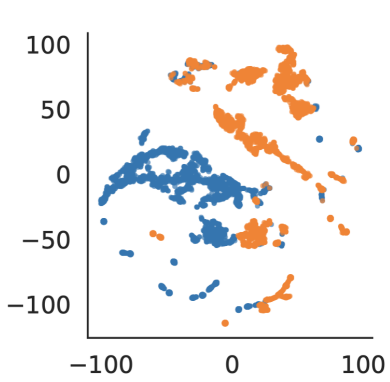
provide F_1 scores of 0.85 and 0.92 when tested on the same body position on which they were trained.

Further, Figure 5.3a (non-diagonal cells) shows the performance of an HAR model trained and tested on different body positions. We can observe a significant performance drop across all pairs of source-target body positions, e.g., when the source='head' and target='forearm', the F_1 score drops to just 0.22. There are however some pairs of body positions for which the performance drop was low, e.g., in the 'head'→ 'upperarm' and 'head'→ 'chest' experiments, the performance drop was significantly lower than 'head'→ 'forearm'. Intuitively, this can be explained by the fact that inertial data captured by forearm-worn sensors is significantly impacted by rotation of the elbow joint, thereby resulting in higher domain shift from the source body position 'head'.

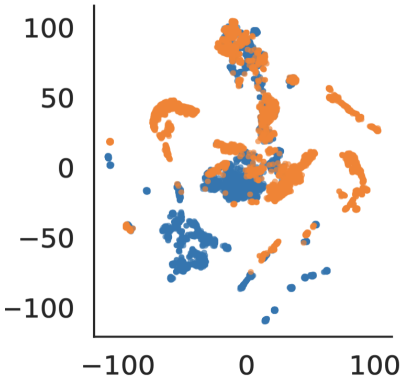
Further, in Figures 5.3b-5.3d, we show the performance of the baseline training strategies, namely Data Augmentation, DDC, and DANN. Finally, Figure 5.3e reports the performance of our proposed technique of Data-Augmented Adversarial Learning. We observe that our approach outperforms all other baselines in 41 out of the 42 experiment condition. For the waist → upperarm experiment, DDC provides a slightly higher performance over our technique. As an example, for the 'chest'→ 'waist' experiment, our approach improves the target domain F_1 score from 0.33 to 0.77.

Visualizing the feature representations. In Figure 5.4, we show the t-SNE [174] visualizations of the feature representations (output of the feature extractor) obtained by different training approaches for the waist→ chest experiment. For this purpose, we ran t-SNE for 5000 iterations with a perplexity of 35 on the features extracted from the test set of source and target body positions. In Figure 5.4a, we observe that the feature representations of the source and target data obtained under the Source-Only model (i.e., the model only trained with source domain data) are easily separable, which reflects the presence of domain shift in the data. In contrast, our proposed solution (Figure 5.4e) forces a significantly higher overlap in the source and target feature representations, which confirms that using data-augmented adversarial training, we are able to extract domain-invariant features from the data.

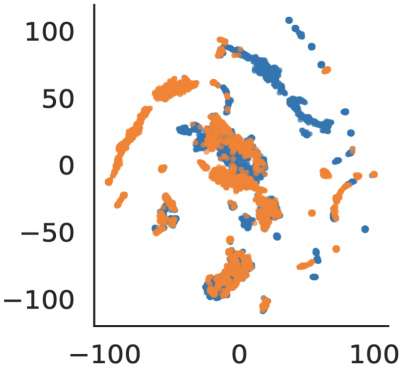
Mismatch in Class Distributions. Although the collection of unlabeled physical activity data (e.g., when the user is running, walking) in the target domain is relatively cheaper, it is important to note that we have little control over the class distribution in the unlabeled data. That is, we do not know beforehand the proportion of different activity classes in the unlabeled target data. Hence, it is important



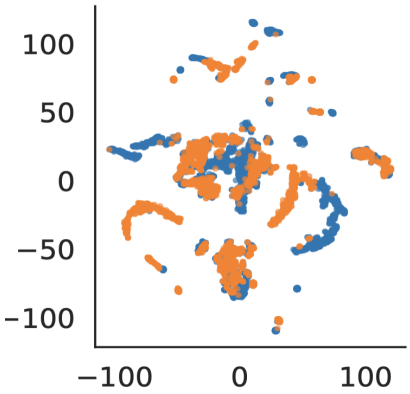
(a) Source-Only



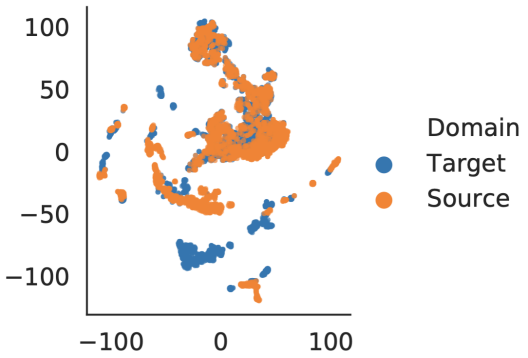
(b) Data Augmentation



(c) Deep Domain Confusion (DDC)



(d) Domain-Adversarial Neural Networks (DANN)



(e) Our Proposed Solution

Figure 5.4: t-SNE visualizations of the feature embeddings generated by different training approaches for the waist→ chest experiment.

to evaluate how well does our proposed technique work under class distribution mismatch between source and target domains.

For this purpose, we simulate various scenarios of class distribution mismatch. Since the source domain in labeled, we know its class distribution. Based on this knowledge, we can select target domain data to have different degrees of

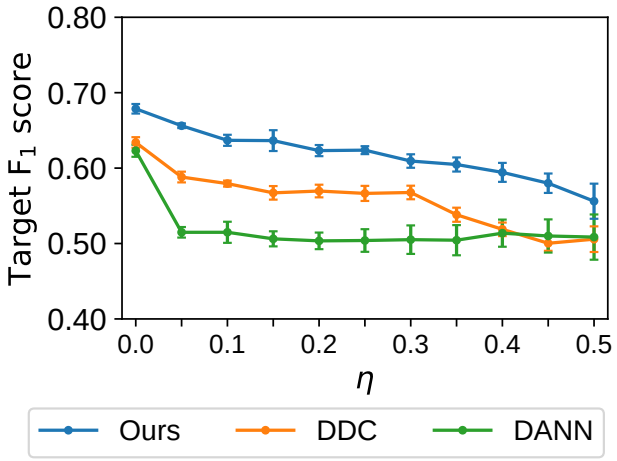


Figure 5.5: Effect of varying the mismatch in class distributions on target domain performance in the head→ thigh experiment.

mismatch with the source class distribution. We introduce a parameter $\eta = \{0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$ to represent this mismatch – when $\eta = 0.0$, unlabeled target data has the same class distribution as the source domain. On the other hand, when $\eta > 0$, e.g., $\eta = 0.05$ we introduce a 5% mismatch in the number of samples for each class between source and target domains. This is achieved by under-sampling data from the dynamic activity classes (climbing up/down, jumping, running) and oversampling data from the static activity classes (standing, sitting, lying) in the target domain. The motivation behind this approach is that HAR systems in practice would encounter a higher proportion of static activities than dynamic activities for an average user; as such, we decide to increase the proportion of static activities in the target domain.

Figure 5.5 shows the effect of varying η on the F_1 score in the target domain. As we can observe, our proposed technique can withstand small shifts in the class distributions, however for extreme cases (e.g., $\eta = 0.5$), there is a significant degradation in performance.

How much unlabeled data is needed in the target domain? Next, we study the effect of changing the amount of unlabeled data available in the target domain on the performance of various adaptation algorithms. For this study, we choose head→ thigh experiment as a representative adaptation setting and gradually increase the unlabeled data available in the target domain (i.e., thigh). The class distribution in the target domain remains the same as that in the source domain for this experiment.

In Figure 5.6, we plot the F_1 score obtained post-adaptation in the target domain as we vary the count of unlabeled samples from 100 to 900. For reference, we also

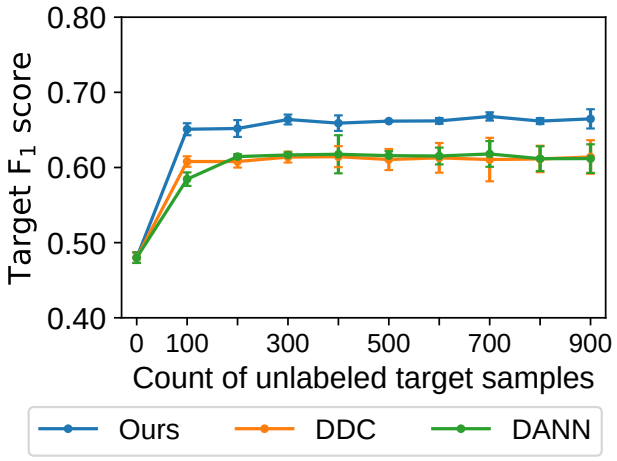


Figure 5.6: Effect of changing the size of unlabeled dataset in the target domain on adaptation performance in the head→ thigh experiment.

show the count = 0 setting, which reflects the performance of the 'head' prediction model in the 'thigh' domain without any adaptation. In this case, we obtain a F_1 score of 0.48 in the target domain.

Interestingly, we find that all adaptation algorithms reach close to their best performance with less than 300 samples. For example, our proposed approach achieves an F_1 score of 0.655 with just 100 samples from the target domain. Recall that each sample in our dataset represents a 3-second IMU segment – therefore, another way to interpret this result is that by performing data-augmented adversarial training on just 300 seconds of unlabeled data collected from 'thigh', we can increase the F_1 score of the 'head' classifier from 0.48 to 0.655.

5.5 Discussion and Limitations

In this section, we discuss the limitations of this work and highlight avenues for future research.

Limited exploration of model architectures. Our investigation of unsupervised domain adaptation for HAR systems was limited to a single model architecture. That said, our model architecture has shown on-par performance compared to the state-of-art HAR models [145], and hence, it was a reasonable choice for our study. However, we are cognizant that new neural network architectures are being proposed for HAR [81] and it is be important to explore how well the adaptation approach presented in this chapter can work with them.

Addressing Class Distribution Mismatch. We demonstrated that our approach

can withstand low mismatches in the class distribution between domains, however as the mismatch becomes bigger (e.g., $\eta = 0.5$), the performance of adaptation degrades. This challenge is also known as 'target shift' across domains, and there have been recent works [175, 176, 177] on correcting target shift during adaptation, which could be tested in the context of HAR systems.

Usable HAR systems. Although our proposed approach provides significant performance gains over the Source-only as well as other baselines, the F_1 score in the target domain after performing domain adaptation remains under 0.8 for all experimental conditions. Clearly, there is a need for more research to further improve the performance to make adapted HAR systems usable in practice. One potential approach worth investigating is semi-supervised domain adaptation [178], wherein a small amount of labeled data is collected in the target domain and is used during the adaptation process.

Extending to other sensor modalities. Although we evaluated our proposed approach of data-augmented adversarial training in the context of HAR models, it could be potentially extended for the microphone-heterogeneity challenge studied in Chapter 4. In a recent work, Park et al. proposed SpecAugment [173], which is an approach to augment speech datasets by adding time-domain and frequency-domain perturbations in the speech spectrograms. Since one of the major causes of domain shift in speech data is the variations in frequency responses of microphones, this approach can potentially simulate different microphone behaviors by masking certain frequency bands. Further, the augmented spectrograms from source and target domains could be used for adversarial training to adapt the weights of a source domain prediction model in order to improve its performance on a target microphone.

5.6 Summary

In this chapter, we proposed an unsupervised domain adaptation solution to scale transparent ML systems. The solution was proposed in the context of HAR models and aimed to counter the domain shifts caused by heterogeneities in sensor hardware, software stack, and sensor placements. We employed data augmentation to first systematically introduce realistic perturbations in the inertial sensor data, that are expected to occur due to hardware and software factors. After obtaining the augmented dataset, we performed adversarial training to adapt the weights of the feature extractor and encourage it to extract domain-invariant features from the inertial data. Through an evaluation on the REALWORLD HAR dataset, we demonstrated that our proposed approach effectively reduces the domain shift in the feature space

and improves the performance of HAR models in the target domain while outperforming several baseline uDA techniques. We also discussed how the proposed solution could be extended to other sensor modalities such as speech.

One key assumption that adversarial uDA techniques make is that the label space of the source and target domains are identical. In the next chapter, we discuss why this assumption can be easily violated in practical ML systems and propose an adaptation solution that extends adversarial uDA techniques to target domains which do not have identical label space to the source domain.

Chapter 6

Domain Adaptation Under Label Space Mismatch

In Chapter 5, we presented an adversarial domain adaptation approach to counter the adverse effects of domain shift in ML systems. This solution assumed that the input and label spaces of source and target domains are identical, i.e., $\mathcal{X}_S = \mathcal{X}_T$ and $\mathcal{Y}_S = \mathcal{Y}_T$. However, in practice, this assumption may not hold. Below we discuss two other problem settings in which the unsupervised domain adaptation can be studied, and then highlight the goal of this chapter.

- *Source and target domains share the same label space but have different input spaces i.e.,* $\mathcal{Y}_S = \mathcal{Y}_T$, $\mathcal{X}_S \neq \mathcal{X}_T$

  This setting is referred to as *heterogeneous domain adaptation* in the literature and is often studied when the data modality or input features differ between source and target domains. For example, Chen at al. [179] and Li et al. [180] trained vision models on RGB-D data (i.e., RGB images with depth information) captured from devices such as Microsoft Kinect (source domain). They investigated the adaptation of these models to RGB target domains where the depth information is not available. Since the feature spaces of the two domains are different, it becomes a problem of heterogeneous domain adaptation. Yao et al. [181] studied even more challenging tasks such as adapting from a textual source domain to a visual target domain.

  Although these settings are interesting and challenging, we argue that they are less likely to manifest in the problems studied in this thesis. As our aim is to scale ML systems under the presence of sensor-induced heterogeneities, it is reasonable to assume that the input spaces of source and target domains are identical, and only the underlying distribution $p(\mathbf{x}, y)$ changes. Hence, the problem setting of heterogeneous domain adaptation is out of the scope of this thesis.

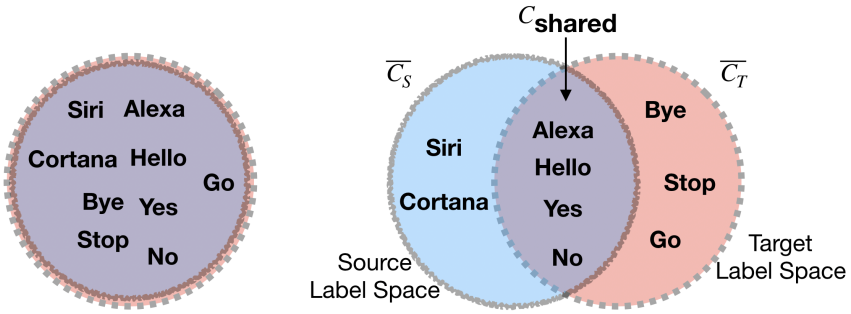


Figure 6.1: (a) Identical and (b) Mismatched Label Spaces in source and target Domains. The latter scenario is likely in practical ML systems and is the focus of this chapter.

- *Source and target domains share the same input space, but different label spaces i.e.,* $\mathcal{X}_S = \mathcal{X}_T$, $\mathcal{Y}_S \neq \mathcal{Y}_T$

  This is a practical setting that we study in this chapter. The adversarial uDA solution presented in Chapter 5 assumed that the label spaces between the domains are identical. However, during the deployment of an ML system, it is infeasible to enforce this assumption because we have no control on the classes or labels that will be encountered in the target domain. Instead, a more practical uDA setting is when the label spaces $\mathcal{Y}_S$ and $\mathcal{Y}_T$ only partially overlap. This setting is also relatively less explored in the context of adversarial uDA algorithms, and hence is the focus of this chapter. In the next section, we further motivate this problem setting and explain the technical challenges associated with it.

6.1 Problem Setting

Consider a scenario where a model developer has a labeled dataset of spoken keywords (e.g., Siri, Alexa) from a ReSpeaker microphone (source domain), and they train a Spoken Keyword Detection model using supervised learning. Now, they wish to deploy this model in an ML system with a Matrix Voice microphone (target domain) from which only unlabeled data is available. The microphone variability here is an example of *domain shift*, which may cause degrade the model performance in the target domain. As we showed in the previous chapters, one can employ adversarial uDA techniques to adapt the source model to the unlabeled target domain.

At their core, adversarial uDA techniques counter domain shift by aligning feature representations of the source and target domains. However, in doing so, they make

a strong assumption that the *label spaces of the source and target domains are identical* – in the example above, this would mean that the keyword classes (e.g., Siri, Alexa) must be identical in both source and target datasets (Figure 6.1(a)). However, in a practical ML system where the data in the target domain is collected without any associated labels, it is impossible to guarantee that the target keyword classes (or the target label space) will be identical to the source keyword classes. For instance, users in the target domain may speak a keyword that is not at all represented in the source dataset. As demonstrated in Fig 6.1(b), a more realistic situation is that the source dataset may contain some keyword classes that are not represented in the target dataset (e.g., Siri, Cortana) and vice versa (e.g., Bye, Stop). We refer to these classes as **private classes**.

This generalized problem setting raises two challenges:

a) **Mitigating negative transfer**. Prior research has shown that the presence of private classes in the dataset can lead to *negative transfer* [182] in adaptation, whereby the adapted classifier performs even worse than a classifier trained solely on the source domain. Note that under a uDA setting, as we have no knowledge of target labels, it is *impossible to know a priori* which classes are private in the target domain or the source domain. Therefore, we need a data-driven solution to identify the shared classes and perform adaptation only between them, while minimizing any negative transfer from private classes.

b) **Labeling private target data as 'unknown'**. After undergoing adaptation, when the classifier is deployed in the target domain, it will encounter target data from shared classes (e.g., Alexa) and private classes (e.g., Stop, Bye). While the classifier can provide labels for the shared class data, it has no knowledge of private class labels (recall that the target domain is unlabeled) – hence, ideally private target data should be classified as 'unknown'. However, prior work shows that neural networks tend to output high confidence predictions even for irrelevant or unrecognizable inputs [183, 40, 184]. This, in turn, means that private target instances may get incorrectly labeled as belonging to one of the source classes.

With the goal of addressing these two challenges, we present a solution named Adaptation under Mismatched Label Spaces (AMLS) wherein our key contribution is in proposing a weighting scheme to down-weigh the contribution of private classes and enhance that of the shared classes in the adaptation process to counter negative transfer. At the same time, AMLS can identify private target classes and classify them as 'Unknown'.

6.2 Background and Related Work

In this section, we review some prior works which have studied the relationship between label spaces and adversarial domain adaptation algorithms. Based on the assumptions made about the label spaces in both domains, these uDA works could be categorized in four buckets:

Closed-set Adaptation. This is the most common uDA setting in the literature (illustrated in Figure 6.1(a)) and places the strongest assumption that label spaces of source and target domains should be identical. Many of the well-known uDA techniques in the literature have been proposed under this assumption [123, 122, 126, 127, 128].

Partial Adaptation. In this setting, the label space of the target domain is assumed to be a subset of the source label space. This is a plausible scenario when a source model trained on a large-scale labeled dataset (e.g., ImageNet [21] with 1000 classes) has to be adapted for a smaller unlabeled target domain (e.g., one containing images of pet animals from 10 ImageNet classes). In this case, private classes are present in the source domain, and previous works have shown that data from private source classes can degrade the adaptation performance of uDA algorithms, leading to negative transfer [182]. To address this problem, Cao et al. [185] proposed to split the adversarial domain discriminator into multiple discriminators (one per source class) and replace the single discriminator loss with a weighted loss across all discriminators. The core idea behind the solution was to downweigh the contribution of discriminators belonging to private source classes in the adversarial adaptation process. Zhang et al. [186] also proposed an importance weighting solution that assigns importance weights to each source sample using an auxiliary domain discriminator. Further, Cao et al. [187] utilized the outputs of the label predictor to obtain class weights and used them to reweigh the contribution of different source samples in adaptation.

Open-set Adaptation. This setting is the opposite of Partial Adaptation discussed above; here, the target label space is assumed to contain private classes that are not represented in the source label space. From the perspective of deploying ML systems, this is also a realistic setting because we cannot put constraints on the type (or class) of data that would be encountered in the target domain. The goal of open-set adaptation is to correctly transfer knowledge from the source to the target domain in the shared classes and to classify the data from private target classes as 'Unknown'. To this end, Saito et al. [188] proposed adding an 'Unknown' class to the

source domain classifier and training it during adversarial learning against a feature extractor. Liu et al. [189] proposed a more general approach of progressive separation wherein multiple binary classifiers are used to first compute the probability of a target sample belonging to one of the source classes. The probability outputs are interpreted as similarity scores between a target sample and a source class, and a second level of 'fine-grained' binary classification is applied to separate samples with the highest and lowest similarity scores. The outputs of the second classifier are taken as importance weights for target samples during adaptation.

Universal Adaptation. The most realistic setting which places no assumption on the overlap of label spaces is known as Universal Adaptation. This is the scenario that is studied in this chapter, and here both source and target domains are allowed to have private classes. Indeed, as we discussed earlier, this aligns with the goal of this thesis of scaling ML systems to unconstrained settings, where we have no labeled data and no control over the target label space $\mathcal{Y}_T$. A recent work in this space is by You et al. [190] who proposed an end-to-end architecture, called Universal Adaptation Network (UAN), which combines prediction entropy and domain similarity into a common metric to separate shared and private classes. UAN was evaluated on a number of visual adaptation tasks and showed significant performance gains over the baseline methods. The work in this chapter builds on UAN (which we use as a baseline in the experiments), and we show that our proposed solution can achieve performance improvements over UAN in multiple speech classification tasks.

6.3 Adaptation under Mismatched Label Spaces (AMLS)

In this section, we formally introduce the notations and problem setting of Universal Adaptation, and then present our solution, AMLS, for extending adversarial uDA algorithms to this problem setting.

6.3.1 Notations

We are given a source domain with input samples $\mathbb{X}_S$ and labels $\mathbb{Y}_S$, and a target domain with input samples $\mathbb{X}_T$. No labels are available in the target domain. Let $p(\mathbf{x})$ and $q(\mathbf{x})$ denote the empirical distributions defined by $\mathbb{X}_S$ and $\mathbb{X}_T$ respectively. For ease of readability, we omit the random variable $\mathbf{x}$ in the notations and refer the marginal data distributions as simply p and q.

We denote the label sets of the source and target domains as C_S and C_T respec-

tively. The set of classes shared between source and target domains are denoted by $C_{\text{shared}} = C_S \cap C_T$. Finally, $\overline{C_S} = C_S \setminus C_T$ and $\overline{C_T} = C_T \setminus C_S$ represent the private label sets of the source and the target domains. Let $p_\cap$ and p_* respectively denote the empirical distributions of source data with label sets C_{shared} and $\overline{C_S}$. Let $q_\cap$ and q_* respectively denote the empirical distributions of target data with label sets C_{shared} and $\overline{C_T}$.

It is worth reiterating that since we have no knowledge of the target labels during training, it is not possible to know C_T, C_{shared}, $\overline{C_S}$ or $\overline{C_T}$ a priori, as they all depend on the knowledge of target label set.

6.3.2 Challenges for Adversarial uDA

In the Universal Adaptation setting, the overall goal of an adversarial uDA algorithm remains the same: how to adapt a model trained in the source domain such that it can achieve good inference performance in the target domain. In the closed-set adaptation setting, adversarial uDA algorithms [123, 122, 126, 127, 128] achieve this goal by aligning feature representations of the source and target domains, using feedback from a discriminator neural network. However, due to the presence of private classes in the Universal Adaptation problem, there is data in the source domain (i.e. belonging to $\overline{C_S}$) and the target domain (i.e., belonging to $\overline{C_T}$) for which no natural feature alignment is possible. Since the uDA algorithms designed for the closed-set setting do not consider label space mismatch, they may attempt to align the data from private classes and cause negative transfer [182] in adaptation. Hence, we need a solution that can identify the shared classes in both domains and only perform adaptation between them.

Further, when we deploy the adapted model in the target domain, it will encounter target data from both C_{shared} and $\overline{C_T}$. While a well-trained domain adaptation solution can ensure that the target data from shared classes is correctly classified, it cannot provide any labels for data from the private target classes ($\overline{C_T}$). Hence, we need a solution that can classify data from $\overline{C_T}$ as 'Unknown'.

In summary, the goal in universal adaptation is to learn a classifier for the target domain which: (i) provides accurate inferences for target data from shared classes C_{shared} under the presence of domain shift, (ii) mitigates the negative transfer caused by private classes $\overline{C_S}$ and $\overline{C_T}$ in the adaptation process, and (iii) assigns an 'Unknown' label to data instances from the private target classes $\overline{C_T}$.

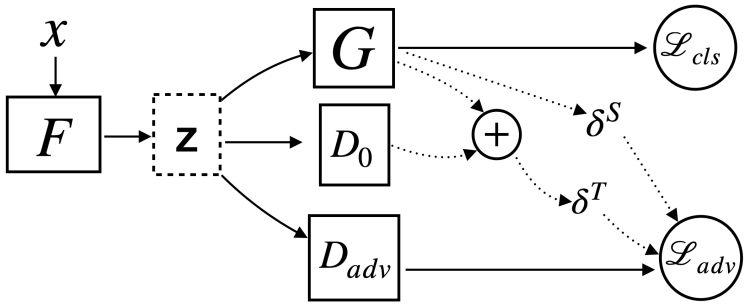


Figure 6.2: Architecture of AMLS. Solid boxes represent neural network components and circles denote various losses that are optimized.

6.3.3 Solution Overview

We propose AMLS, an end-to-end architecture for uDA under the presence of label space mismatch. As discussed, the core challenge here comes from the presence of private classes in both domains, which may lead to *negative transfer*. Intuitively, this negative transfer can be mitigated if we can isolate the shared classes C_{shared} and only align their feature representations, while ignoring or down-weighing the contribution of the data from private classes during adaptation.

Figure 6.2 illustrates the proposed architecture for AMLS. Similar to prior works on adversarial domain adaptation, this solution consists of a feature extractor F, a classifier G and an adversarial discriminator D_{adv}. When an input $\mathbf{x}$ is fed to this architecture, a feature representation $\mathbf{z} = F(\mathbf{x})$ is obtained. The extracted features are then passed to G to obtain a softmax probability distribution $\hat{y} = G(\mathbf{z})$ over the source labels C_S. The classifier G is trained on labeled source data by optimizing a supervised cross-entropy loss as follows:

$$\mathcal{L}_{\text{cls}} = -\mathbb{E}_{(\mathbf{x_s},y_s)\sim(\mathbb{X}_\mathbb{S},\mathbb{Y}_\mathbb{S})} \sum_{k=1}^{|C_\text{S}|} \mathbb{1}_{[k=y_s]}[\log(G(F(\mathbf{x_s}))] \tag{6.1}$$

Next, the task of aligning feature representations of source and target domains is performed by D_{adv} using adversarial learning. In closed-set adaptation, D_{adv} would align the representations over the entire label space, however, this can lead to *negative transfer* due to the presence of private classes. One way to mitigate this challenge is to force D_{adv} to give higher importance (or weights) to the shared classes in the feature alignment process as compared to the private classes. Thus, the (weighted) adversarial loss formulation for D_{adv} could be written as:

$$\mathcal{L}_{adv} = -\mathbb{E}_{\mathbf{x_s}\sim p}[\delta^S(\mathbf{x_s})\log(D_{adv}(F(\mathbf{x_s}))] - \mathbb{E}_{\mathbf{x_t}\sim q}[\delta^T(\mathbf{x_t})\log(1 - D_{adv}(F(\mathbf{x_t}))] \tag{6.2}$$

where δ^S and δ^T are the weights to be assigned to a source and target sample respectively in the adaptation process. Ideally, we would like to assign higher weights to samples from shared classes and lower weights to private samples. Formally, these criteria could be written as:

$$0 \leq \mathbb{E}_{\mathbf{x_s} \sim p_*} \delta^S(\mathbf{x_s}) < \mathbb{E}_{\mathbf{x_s} \sim p_\cap} \delta^S(\mathbf{x_s}) \leq 1 \tag{6.3}$$

$$0 \leq \mathbb{E}_{\mathbf{x_t} \sim q_*} \delta^T(\mathbf{x_t}) < \mathbb{E}_{\mathbf{x_t} \sim q_\cap} \delta^T(\mathbf{x_t}) \leq 1 \tag{6.4}$$

6.3.4 AMLS: Weighting Schemes

One of the key contribution and technical novelty of this work is in proposing a technique to estimate δ^S and δ^T that satisfy Equations 6.3 and 6.4. Below we describe the two weighting schemes.

Estimating Target Weights. When an input sample $\mathbf{x_t}$ from target domain is fed to the classifier G, we get a probability distribution over the source class set C_S in the form of softmax outputs.

$$\hat{y}_t = G(F(\mathbf{x_t}))$$

We argue that the classifier G will be more confident in its predictions $\hat{y}_t$ for inputs from the shared classes C_{shared} as compared to those from the private classes $\overline{C_T}$. This is a reasonable hypothesis because despite the presence of domain shift, classes in C_{shared} are likely to be closer to the source domain as compared to the private classes $\overline{C_T}$. Hence, any measure of prediction confidence that satisfy Equation 6.4 could be used as a weighting function to separate C_{shared} and $\overline{C_T}$.

We propose to employ prediction margins as the criterion for classifier confidence. Prediction margins have been prominently used in active learning [191] to sample data instances for which a classifier is least confident. Formally, Margin M is defined as $M(\mathbf{x_t}) = \hat{y}_t^1 - \hat{y}_t^2$, where $\hat{y}_t^1$ and $\hat{y}_t^2$ represent the highest and second-highest softmax outputs in $\hat{y}_t$. When a classifier has high confidence about its top prediction, M will be high. On the contrary, for a data sample for which a classifier is less confident, M, i.e., the difference between the top two softmax outputs will be low. Equation 6.5 follows from this logic.

$$0 \leq \mathbb{E}_{\mathbf{x_t} \sim q_*} M(\mathbf{x_t}) < \mathbb{E}_{\mathbf{x_t} \sim q_\cap} M(\mathbf{x_t}) \leq 1 \tag{6.5}$$

It is easy to observe that the use of prediction margins satisfies the target weighting criterion in Eq. 6.4. However, due to the presence of domain shift, the margins

obtained on target data could be noisy and may lead to incorrect weights for target samples.

As such, in addition to the prediction margins, we also propose to employ a separate non-adversarial domain discriminator to provide an additional signal about private and shared target classes. First, the non-adversarial domain discriminator D_0 is trained to separate samples from source and target domains. This can be achieved in the form of a binary classification task, where we assign label=1 to the source data and label=0 to the target data, and train the discriminator by optimizing the Binary Cross-Entropy loss as shown below.

$$\min_{D_0} \; \mathbb{E}_{\mathbf{x_s}\sim p}\left[L_{\text{BCE}}\left(D_0\left(F(\mathbf{x_s})\right),1\right)\right] + \mathbb{E}_{\mathbf{x_t}\sim q}\left[L_{\text{BCE}}\left(D_0\left(F(\mathbf{x_t})\right),0\right)\right]$$

Once D_0 is trained, we feed the unlabeled target samples to it and obtain the probability that they belong to the source domain. We can expect that D_0 would output a higher probability for samples from the shared target classes (C_{shared}) as they have more similarity with the source domain. On the other hand, lower probabilities are expected for samples from private target classes ($\overline{C_T}$) which have no overlap with the source domain. Therefore, the following should hold true.

$$0 \leq \mathbb{E}_{\mathbf{x_t}\sim q_*}\left[D_0(\mathbf{x_t})=1\right] < \mathbb{E}_{\mathbf{x_t}\sim q_\cap}\left[D_0(\mathbf{x_t})=1\right] \leq 1 \tag{6.6}$$

The use of both margin and domain discriminator signals to estimate target weights can offset any noise in one of the measurements; as such, we propose to calculate target weights as,

$$\delta^T(\mathbf{x_t}) = 0.5 * (M(\mathbf{x_t}) + [D_0(\mathbf{x_t})=1]) \tag{6.7}$$

which indeed satisfies the target weighting criterion shown in Eq. 6.4.

Estimating Source Weights. As shown in Eq. 6.3, the goal of weighting source samples is to assign higher weights to samples from shared classes and lower weights to samples from private source classes during the adaptation process.

To estimate the source weights, we leverage an interesting property of $\hat{y}_t = G(F(\mathbf{x_t}))$. Recall that $\hat{y}_t$ is a probability distribution over the source label space C_S, denoting the probability of a *target sample* $\mathbf{x_t}$ belonging to different classes in the source set C_S. We hypothesize that source classes (C_{shared}) which are shared with the target domain will be given higher probabilities in $\hat{y}_t$ and the private source classes $\overline{C_S}$ will have lower probabilities. This is reasonable because target data $\mathbf{x_t}$ has no overlap with private source classes; hence the classifier should estimate low

probabilities for $\overline{C_S}$. Thus, by observing the output probability distribution over source classes, we can potentially distinguish shared and private source classes and assign appropriate weights to them.

However, due to the presence of samples from private target classes, these class probabilities could be noisy. To address this, we use the target weights δ^T to first weight the class probabilities obtained for each target sample $\mathbf{x_t}$, and then average the weighted class probabilities over an entire batch B of data to obtain the mean class probability vector η.

$$\eta = \frac{1}{|B|}\sum_{i=1}^{|B|} G(F(\mathbf{x_t^i})) * \delta^T(\mathbf{x_t^i}) \tag{6.8}$$

where $\delta^T(\mathbf{x_t^i})$ is the weight assigned to the i^{th} target sample $\mathbf{x_t^i}$ in the batch. Effectively, η could be interpreted as a $|C_S|$-dimensional vector, where η^k reflects the weight of the k^{th} source class in a given batch of data ($k \in \{1, 2, \ldots, |C_S|\}$).

Given η, we set the weight of a labeled source sample $(\mathbf{x_s}, y_s)$ as follows:

$$\delta^S(\mathbf{x_s}) = \eta^{y_s}$$

and expect that under this weighting scheme, samples from shared source classes will be assigned higher weights than samples from private source classes.

6.3.5 AMLS: Training and Inference Pipelines

We now explain the end-to-end training pipeline of AMLS. As with other adversarial training architectures, AMLS jointly optimizes the classification loss on the labeled source domain $\mathcal{L}_{\text{cls}}$ along with the weighted adversarial loss $\mathcal{L}_{\text{adv}}$ shown in Equation 6.2 where the source weights δ^S and target weights δ^T are calculated as described above.

In addition, we also propose to generate pseudo labels [192] for the target data and use supervised learning to further refine the classifier for use in target domain. For a given unlabeled target sample, a pseudo label can be generated by choosing the class $c \in C_S$ which has the maximum predicted probability by the classifier G. While pseudo-labels have been previously used for closed-set domain adaptation [193], it is challenging to use them in the presence of private classes in the target domain ($\overline{C_T}$). Since $\overline{C_T}$ has no overlap with the source domain classes C_S, the pseduo labels generated for samples from $\overline{C_T}$ will be incorrect; more critically, supervised training with incorrect pseudo labels may adversely impact the adaptation performance. To alleviate this challenge, we can leverage the target weights δ^T that were estimated earlier and only perform pseudo-label based supervised training on target samples whose weights are above a certain threshold δ_p. Formally, the pseudo label

classification loss can be expressed as follows:

$$\mathcal{L}_{\text{pseudo}} = -\mathbb{E}_{\mathbf{x_t}\sim q}\left[\mathbb{1}_{[\delta^T(\mathbf{x_t})>\delta_p]}\cdot\sum_{k=1}^{K}\mathbb{1}_{[k=\text{argmax}(\hat{y}_t)]}[\log(\hat{y}_t)]\right]$$

Putting them together, the combined optimization objective of AMLS is:

$$\min_{D_{\text{adv}}}\min_{F,G}\mathcal{L}_{\text{cls}}+\lambda\mathcal{L}_{\text{pseudo}}-\mathcal{L}_{\text{adv}}$$

where λ is a hyperparameter to control the contribution of pesudo-label classification loss during adaptation.

Inference Pipeline. Given a target sample $\mathbf{x_t}$, we first compute its weight $\delta^T(\mathbf{x_t})$. If the weight is below a threshold δ_0, it is likely that this sample belongs to a private class and we label it as 'Unknown'. Otherwise, we compute $\hat{y}_t = G(F(\mathbf{x_t}))$ and output argmax$(\hat{y}_t)$ as its label. Note that δ_0 and δ_p are hyperparameters that are tuned using cross-validation.

6.4 Experiment Setup

In this section, we describe the experimental setup used to evaluate AMLS, including the speech datasets, label space splittings, neural architectures, and baselines.

6.4.1 Tasks and Datasets

AMLS is evaluated on three speech classification tasks where we simulate label space mismatch by dividing the classes across the source and target domains.

- *Microphone Adaptation*: For this task, we use the *Multi-microphone Spoken Keywords* dataset presented in Chapter 3. This dataset has 65,000 spoken keyword recordings from 31 classes simultaneously recorded on three microphones, namely Matrix Voice (M), ReSpeaker (R) and USB (U). Each microphone represents a domain, and the task is to adapt a keyword detection model trained on a source microphone to a target microphone. We show results for M→R, R→U, and U→M adaptation tasks.

- *Gender Adaptation*: Next, we study cross-gender adaptation in a Keyword Classification model. For this experiment, we use the Spoken Keywords dataset [140] from Google, which consists of >100k speech utterances from 35 keyword classes (e.g., Yes, Right). We partitioned this dataset based on the speaker's gen-

der, which was obtained through a crowd-sourced gender-labeling exercise that we ran on Amazon Mechanical Turk. The partitions, Male (M) and Female(F) represent different domains, and we show results for M→F and F→M adaptation.

- *Emotion Dataset Adaptation*: Finally, we evaluate a challenging task of adapting a speech emotion classification model trained on a source dataset (CREMA-D [194]) to a target dataset (RAVDESS [156]). While CREMA-D dataset contains emotional speech recordings of 91 different users with varying ethnicity and ages (20-69 years), RAVDESS contains speech recordings from 24 young users based in Toronto, Canada (age range 21-33). In addition to the demographic differences between users, the recording environments used for the data collection study are also different and could induce domain shift between the datasets, thereby making it a challenging domain adaptation task. Both datasets contain emotional speech from six classes: Anger, Disgust, Fear, Happy, Neutral/Calm, and Sad; in addition, RAVDESS contains an additional class called Surprise. We consider the datasets as different domains and evaluate the CREMA-D→RAVDESS task.

6.4.2 Experiment Protocol

In order to evaluate AMLS under label space mismatch, we partition the label space of the various datasets across source and target domains.

For the Gender Adaptation task on the Spoken Keywords dataset, we assign a class number to each keyword class as per the alphabetical order, e.g., the first class in the alphabetical order is assigned label='1' and so on. Thereafter, we randomly sample 20 classes from the complete label set without replacement and consider them as source classes C_S. The remaining 15 classes are considered as private target classes $\overline{C_T}$. Next, we randomly sample 10 classes from C_S and consider them as the shared classes C_{shared} between source and target domains. Based on this partitioning, we obtain $\overline{C_S} = \{33,3,7,8,10,14,20,21,23,26\}$, $C_{\text{shared}} = \{4,5,9,11,12,15,17,24,27,28\}$ and remaining classes are in $\overline{C_T}$.

For Microphone Adaptation, we use a similar scheme of partitioning the label space and obtain $\overline{C_S} = \{1,3,5,9,10,11,12,22,25,28\}$, $C_{\text{shared}} = \{6,8,14,16,21,23,24,27,29,30\}$ and remaining classes are in $\overline{C_T}$.

For Emotion Adaptation with 7 classes, we use $C_{\text{shared}} = \{\text{Calm, Angry, Fear}\}$, $\overline{C_S} = \{\text{Happy, Sad}\}$ and $\overline{C_T} = \{\text{Disgust, Surprise}\}$.

Baselines. AMLS is compared with four baselines: i) *Source Only* where the target data is fed to the source domain model, without adaptation, ii) *ADDA* [123], iii) *DANN* [122] and iv) UAN [190]. ADDA and DANN are well-known adversarial uDA techniques, but do not consider label space mismatch – hence, they serve as representative baselines for existing speech uDA works. UAN is designed for vision tasks to handle label space mismatch and hence is an appropriate state-of-the-art baseline for universal adaptation.

Model Architectures. In line with prior works [69, 28], we use convolutional neural networks (CNNs) to build the *Keyword Classification* (KC) and *Emotion Classification* (EC) models. The inputs to these models are two-dimensional tensors extracted from speech utterances, consisting of time frames on one axis and 40 MFCC features on the other axis. The architectures are as follows: for KC, feature extractor F: [Conv: $\{64,64,64\}$], classifier G: [FC: $\{256, 128\}$] where Conv represents the number of convolution kernels in each layer and FC denotes the number of units in each hidden layer. For EC, feature extractor F: [Conv: $\{128, 64,64,64\}$], classifier G: [FC: $\{256, 128\}$]. The architecture for D_{adv} and D_0 is [FC: $\{128, 128\}$]. We use importance-weighted cross-validation [170, 171] to select the hyper-parameters and our system is implemented in TensorFlow 2.0.

Evaluation Protocol. We follow the same evaluation protocol as earlier uDA works: 80% of the unlabeled data from the target domain is used during adversarial training, and the adapted model is tested on a 20% held-out target test set. All the data samples from private target classes are assigned a common ground truth of 'Unknown', and we report the mean accuracy across all $|C_{\text{shared}}| + 1$ target classes.

6.5 Results

In this section, we present the results of our experiments.

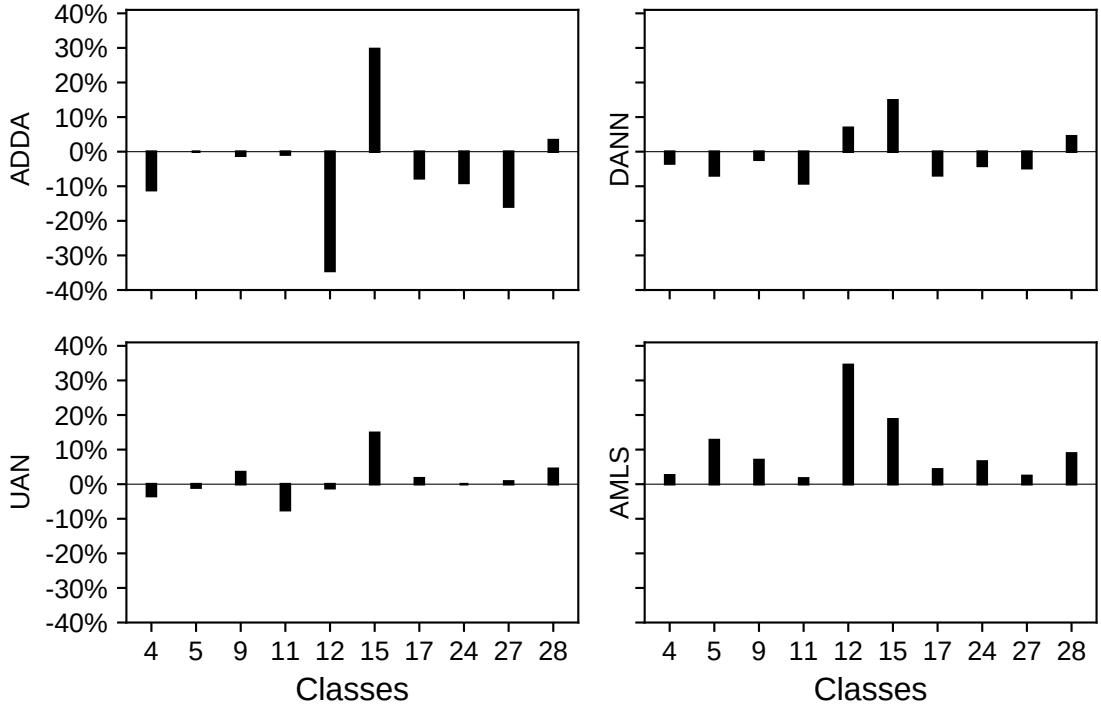
Target Domain Performance. In Table 6.1, we report the target test accuracy for various adaptation tasks averaged over shared and private classes. When adapting a keyword model trained on Male speakers to Female speakers (M→F) and vice-versa (F→M), we observe that the accuracy of ADDA and DANN is similar or even lower than the Source-only baseline, hinting to the occurrence of negative transfer. Both UAN and AMLS significantly boost the target performance, with AMLS providing accuracy gains between 3-7% over UAN. We observe similar trends for the microphone and emotion adaptation tasks, with AMLS outperforming UAN by 1.5-3%.

	Gender Adaptation		Microphone Adaptation			Emotion Adaptation
	M → F	**F → M**	**M → R**	**R → U**	**U → M**	**C → R**
Source	41.41±0.16	35.04±0.33	40.11±0.39	40.93±0.22	42.44±0.18	35.2±0.12
ADDA	41.54±0.77	28.90±1.06	39.21±0.50	39.05±0.61	39.29±1.13	30.0±0.23
DANN	40.13±0.70	29.85±0.70	40.19±0.63	41.20±0.34	42.33±0.83	28.18±0.21
UAN	66.13±0.53	60.82±0.44	66.30±0.58	67.22±0.65	67.20±0.47	38.96±0.40
AMLS w/o $\mathcal{L}_{\text{pseudo}}$	67.32±0.60	62.29±0.43	66.48±0.34	66.62±0.45	67.29±0.51	40.07±0.29
AMLS w/o δ^S and δ^T	68.87±0.71	60.38±0.51	65.11±0.39	65.70±0.47	65.10±0.69	37.91±0.31
AMLS	**73.78±0.54**	**64.1±0.50**	**69.02±0.44**	**67.98±0.36**	**68.77±0.55**	**41.45±0.27**

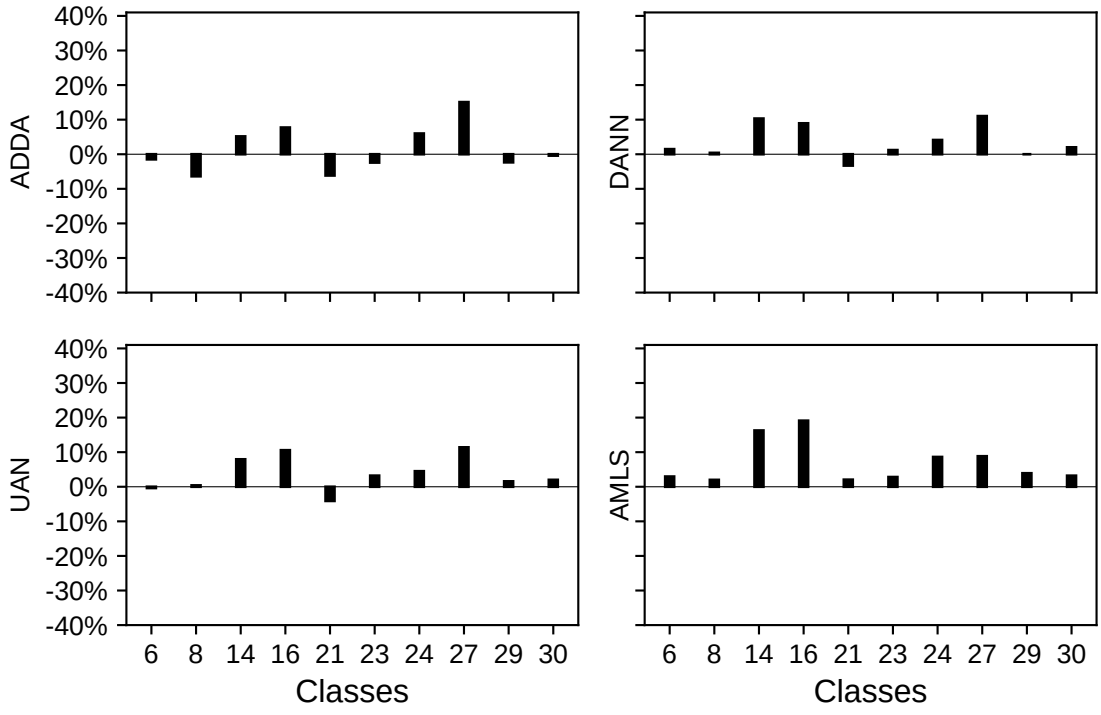
Table 6.1: Target domain accuracy averaged over shared (C_{shared}) and private ($\overline{C_T}$) classes, with highlighted 95% confidence intervals (over five experiment runs). AMLS significantly outperforms ADDA and DANN, and also provides gains over UAN, which is designed for universal adaptation.

Quantification of negative transfer. In Figure 6.3, we illustrate the change in per-class accuracy for the shared target classes after adaptation using various techniques. As evident in Figure 6.3a, both ADDA and DANN suffer from negative transfer in the M → F gender adaptation task, leading to accuracy degradation for some classes after adaptation. This is primarily caused due to the label space mismatch between domains, which these techniques do not account for during adaptation. UAN manages to reduce the negative transfer for most classes; however, it does end up with minor negative transfer for some classes. AMLS, on the other hand, is able to counter the negative transfer fully, thereby providing significant accuracy gains over other baselines. Similar trends can be observed in the M → R microphone adaptation task.

Varying the size of shared label space. We now compare AMLS with the baselines as the size of shared label space increases. As the total classes in each dataset ($C_S \cup C_T$) are fixed, by increasing the size of shared label space, we reduce the label space mismatch. We study the M→F Gender Adaptation task as an example and gradually increase the number of shared classes from 1 to 20. In Figure 6.4, we observe that when label space mismatch is high, ADDA and DANN baselines perform poorly due to negative transfer and inability to classify private target classes as 'Unknown'. UAN and AMLS provide significant gains in these scenarios, with AMLS outperforming UAN in all cases. As the label set mismatch reduces, the performances of ADDA and DANN improve, however UAN and AMLS still outperform them. In the other extreme case of no label space mismatch (not shown in the figure), all the adaptation techniques converge to similar accuracies.



(a) Gender Adaptation task: M $\rightarrow$ F



(b) Microphone Adaptation task: M $\rightarrow$ R

Figure 6.3: For two adaptation tasks, this figure shows the relative change in per-class accuracy of shared target classes after adaptation. Negative transfer can be observed in DANN and ADDA.

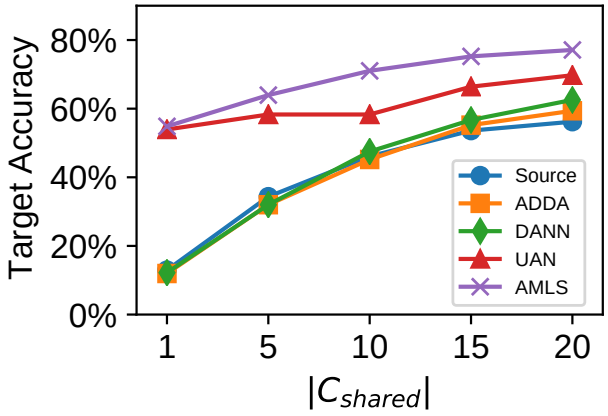


Figure 6.4: Comparison of different uDA approaches as $|C_{\text{shared}}|$ varies in the M→F Gender Adaptation task.

Ablation Study. Next, we present an ablation study to investigate the contribution of the various components of AMLS. More specifically, we evaluate two settings: i) when pseudo labels in the target domain are not used and hence $\mathcal{L}_{\text{pseudo}}$ is not optimized and ii) when only $\mathcal{L}_{\text{pseudo}}$ is optimized, but the adversarial loss $\mathcal{L}_{\text{adv}}$ is not adjusted based on source weights δ^S and target weights δ^T. From Table 6.1, it can be observed that AMLS provides poor performance if the adversarial loss and pseudo loss is not appropriately weighted to down-weigh the contribution of private classes; more specifically, in all experiments with the exception of (M → F), AMLS performance becomes worse than UAN. This result confirms that the weighting schemes proposed in this chapter are important for adaptation performance under mismatched label spaces. Table 6.1 also confirms that we can boost target domain performance using appropriately weighted pseudo classification loss.

Hypotheses Justification. Finally, we present an experiment to evaluate the validity of the hypotheses used to design the source and target weighting schemes in §6.3.4. Recall that the goal of both weighting schemes is to assign higher weights to samples from the shared classes and lower weights to samples from the private classes. Table 6.2 shows the mean (and 95% confidence intervals) of the weights obtained for source and target domains in two adaptation tasks: gender adaptation (F → M) and microphone adaptation (M → R).

We observe that for both adaptation tasks, weights assigned to the source samples (δ^S) from shared classes are significantly higher than the weights for private classes. We also report the results of a Welch's t-test on the weights from shared and private classes, and observe a significant difference between them ($p < 0.00001$). These findings confirm that our source weighting scheme manages to downweigh the con-

	F→M		M→R	
	Source Weights (δ^S)	Target Weights (δ^T)	Source Weights (δ^S)	Target Weights (δ^T)
Shared Classes	$0.64_{\pm 0.015}$	$0.69_{\pm 0.009}$	$0.67_{\pm 0.009}$	$0.65_{\pm 0.014}$
Private Classes	$0.09_{\pm 0.019}$	$0.53_{\pm 0.011}$	$0.11_{\pm 0.017}$	$0.50_{\pm 0.013}$
t-statistic	70.18	26.83	83.0	26.89
p-value	< 0.00001	< 0.00001	< 0.00001	< 0.00001

Table 6.2: Mean and 95% confidence intervals of the weights obtained for source and target samples in two adaptation tasks. A Welch's t-test is done to compare the weights from the shared and private classes.

tribution of private source classes in adaptation, which is important to avoid the negative transfer. We observe a similar trend for the target weights (δ^T), albeit the difference between shared and private classes is not as high as the source weights. Nevertheless, the difference is statistically significant ($p < 0.00001$) and confirms that at inference time we can use these weights to identify samples from the private target classes and label them as 'Unknown'.

6.6 Discussion and Limitations

The problem setting studied in this chapter relaxed a key assumption about the identical source and target label spaces in adversarial uDA algorithms. However, there are other factors that might still degrade uDA performance in practice. Even when the label spaces are identical, there may be a shift in the marginal label distributions between domains, i.e., $P_S(y) \neq P_T(y)$. For example, when the Keyword Detection model is deployed in a new domain, the users may speak certain keywords (e.g., Alexa, Siri) more than others (e.g., Stop, Go) depending on their personal preferences. It may lead to the target label distribution to become significantly different from the source label distribution. Prior works [176, 175, 195, 177] have studied this problem under the umbrella of *target shift* and shown that the adaptation performance of uDA can degrade, if target shift is not addressed.

Target shift could also manifest in the shared label space C_{shared} of source and target domains. In our experiments, we did not simulate this effect and evaluated AMLS assuming that there is no change in label distributions in the shared label space.

This remains a limitation that could be addressed using the techniques proposed in the computer vision literature [176].

6.7 Summary

In this chapter, we extended adversarial uDA algorithms to the universal adaptation setting wherein label spaces between source and target domains are not identical. Relaxing the assumption about shared label spaces is particularly useful when ML systems are scaled to target domains in unconstrained settings, where we have little control over the classes or labels in the target domain data. Our proposed solution, Adaptation Under Mismatched Label Spaces (AMLS), is based on a weighting scheme that automatically assigns higher weights to samples from the shared source and target domain classes, while down-weighing the contribution of private classes in the adaptation process. In AMLS, this weighting scheme is used to modify the adversarial learning objective of uDA, and is also combined with supervised learning in the target domain by generating pseudo labels for selected target data. Through a series of experiments on speech classification tasks, we demonstrated that AMLS could mitigate negative transfer due to the presence of private classes and outperform a state-of-the-art universal adaptation baseline named UAN.

All the adaptation solutions presented in the thesis so far assumed that the source and target domain datasets are available on the same machine. In the next chapter, we relax this assumption and study how uDA algorithms can be used to scale distributed ML systems, where the source and target datasets are hosted on different machines and cannot be shared with each other due to privacy considerations.

Chapter 7

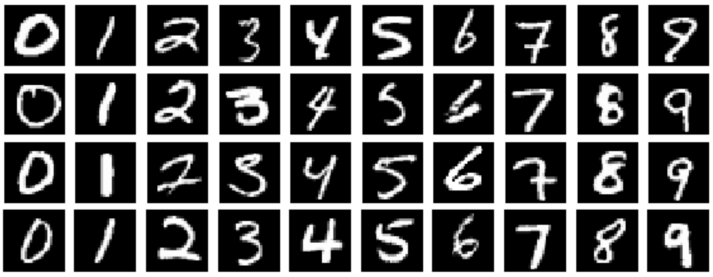
Scaling Distributed ML Systems with Multiple Target Domains

The problem settings considered in the previous chapters assumed that an ML system trained in a labeled source domain has to be scaled to a single unlabeled target domain. Moreover, the proposed uDA algorithms implicitly assumed that both the source and target datasets are available on the same machine, and can be accessed during the adaptation process. In this chapter, we propose solutions to extend uDA algorithms to settings where ML systems have to scale to multiple target domains, and when the source and target datasets are private and stored in distributed nodes.

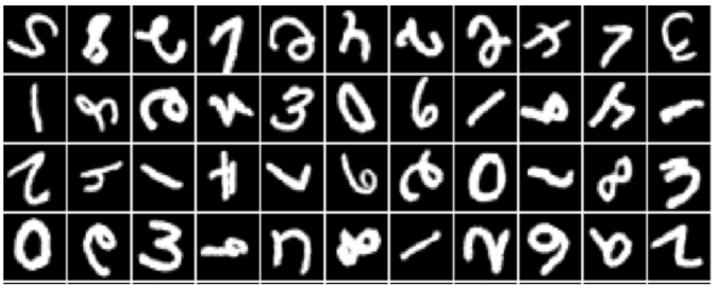
7.1 Motivation

As a motivating example, consider the recent interest in using ML to predict COVID-19 in patients by analyzing computerized tomography (CT) scans of their chest [196, 197, 198]. Assume that scientists in China (*source domain*) have collected a large labeled dataset of CT scans (e.g., [199]) and trained a COVID-19 prediction model on it. This model now needs to be deployed in some other countries (*target domains*) from where *labeled data is unavailable*. Due to virus mutations across the world, the CT scan samples from different countries may follow different data distributions, and it is possible that the source domain model may not work accurately for all target countries. Hence, uDA could become a promising approach to adapt the source model for each *target* country's data distribution. In this context, we introduce two challenges faced by uDA techniques:

(i) Static Adaptation Collaborators. For a given target domain, we define its *adaptation collaborator* (or simply, collaborator) as the domain whose prediction model is adapted using uDA for use in the target domain. For example, if we adapt a speech model from ReSpeaker → Matrix microphone, then ReSpeaker is considered the adaptation collaborator for Matrix.



(a) Samples from the MNIST dataset



(b) Samples from the Rotated MNIST dataset

Figure 7.1: Illustration of rotation-induced domain shift in the Rotated MNIST dataset.

Existing uDA methods are static by design, in that they assume that when an ML system is deployed in a new unlabeled target domain, it would *always* choose the labeled source domain as its adaptation collaborator. We argue that this static approach of *always* adapting from a labeled source is not optimal and can result in poor adaptation performance for a target. For instance, some target countries may have a virus strain very different from China (the labeled source), and hence adapting from the Chinese COVID-19 model may not be optimal for them.

To better explain this problem, we show an experiment on Rotated MNIST, a variant of MNIST [200]. Figure 7.1a shows samples from the MNIST dataset which contains images of 70,000 handwritten digits from 0 to 9. Rotated MNIST, as shown in Figure 7.1b is created by rotating the digits in MNIST dataset by different angles. For our experiment, 0°, i.e., no rotation, represents the labeled source domain on which we can train a prediction model using supervised learning. We take 30°, 60°, 45°, 90° and 15° as the unlabeled target domains for which we would like to learn a prediction model using uDA techniques.

Figure 7.2a shows the accuracy obtained in each target domain if we always choose the labeled source 0° as their collaborator. While this approach results in high accuracies for 15° and 30°, it performs poorly for other domains. Can we do better? *What if a target domain can adapt not just from the labeled source, but also from*

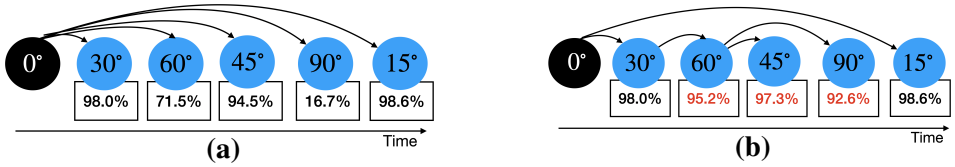


Figure 7.2: 0° is the labeled source domain while the domains in blue are unlabeled target domains. The numbers in rectangle denote the post-adaptation accuracy for a domain. (a) Static Design: Labeled Source acts the collaborator for each target domain. (b) Flexible Design: Each target domain chooses its collaborator dynamically. Previously adapted target domains can also act as collaborators. Note that choosing the right collaborator leads to major accuracy gains over the Static Design for many domains (shown in red).

other target domains which themselves have undergone adaptation in the past. Figure 7.2b shows that if target domains could flexibly choose their collaborators, they can achieve significantly higher accuracies. E.g., if 90° adapts from 60° (which itself underwent adaptation in the past), it could be achieve an accuracy of 92.6%, almost 75% higher than what could be achieved by adapting from 0°.

In summary, when uDA algorithms scale to settings with multiple target domains such as the COVID-19 example, selecting an optimal collaborator for each target domain becomes critical. How do we select this optimal collaborator is a key research question that this chapter aims to answer.

(ii) Support for Distributed Domains. Most adversarial uDA methods (e.g., [122, 123]) assume that datasets from the source and target domains are available on the same machine before the adaptation process begins. While this assumption makes it easy to develop and evaluate uDA algorithms, it may not hold in practice. In the example above, the CT scans of patients are sensitive health records, and the source and target domains may not be allowed to share it with each other due to privacy reasons. Similarly, in the microphone heterogeneity experiment in Chapter 4, the speech data could be private, and users in the source and target domains may not want to share it with each other. Clearly, this bottleneck can severely limit the adoption of uDA techniques in realistic settings. How can we extend uDA techniques to work in distributed settings while preserving the privacy of each domain?

7.2 Preliminaries and Problem Formulation

In this section, we formulate the technical problems that are studied in this chapter.

7.2.1 Notations and Primer

Let $\mathcal{D}_S$ denote a source domain with data $\mathbb{X}_S$ and labels $\mathbb{Y}_S$, and $\mathcal{D}_T$ be a target domain with data $\mathbb{X}_\mathbb{T}$, but *without* any labeled observations. For a K-way classification task, we train a feature extractor, F_S, and a classifier, G_S in the source domain by minimizing the categorical cross-entropy loss:

$$\min_{F_S,G_S} \mathcal{L} = -\mathbb{E}_{(\mathbf{x_s},y_s)\sim(\mathbb{X}_\mathbb{S},Y_S)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]}[\log(G_S(F_S(\mathbf{x_s}))] \tag{7.1}$$

Now the goal for uDA is to learn a feature extractor F_T for the unlabeled target domain, such that it minimizes the distance between the empirical source and target feature distributions. When the distance between feature representations is minimized, the source classifier G_S can be directly applied to target feature representation without learning a separate G_T.

To learn F_T, two losses are optimized using adversarial training: a discriminator loss $\mathcal{L}_{adv_D}$ and a mapping loss $\mathcal{L}_{adv_M}$. uDA algorithms have different ways of computing these losses— for example, ADDA [123] uses label inversion to minimize the Jensen-Shannon divergence between source and target feature distributions as follows:

$$\min_{DI} \mathcal{L}_{adv_D} = -\mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_S}[\log(DI(F_S(\mathbf{x_s}))] - \mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_\mathbb{T}}[\log(1-DI(F_T(\mathbf{x_t}))] \tag{7.2}$$

$$\min_{F_T} \mathcal{L}_{adv_M} = -\mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_T}[\log(DI(F_T(\mathbf{x_t}))] \tag{7.3}$$

where *DI* represents a domain discriminator that aims to distinguish source and target domains.

7.2.2 Problem Formulation

We now formalize our problem setting of scaling uDA to scenarios with multiple target domains, each with sensitive private data. Assume that a prediction model trained for the source domain $\mathcal{D}_S$ has to be adapted for multiple *unlabeled* target domains $\{\mathcal{D}_T^j \,|\, j = 1,\ldots,K\}$ with data $\mathbb{X}_T^j$ and *no* labeled observations. We assume that the ML system encounters new target domains sequentially, one at a time. Under this problem setting, our research objectives are:

(i) Optimal Collaborator Selection (OCS). For each target domain that joins the ML system, how do we select an optimal collaborator for uDA? We define a candidate set $\mathbb{Z}_\tau$ as the set of candidate domains that are available to collaborate with a target domain at step τ. When a uDA system initializes at step $\tau = 0$, only the labeled source domain has a learned model, hence $\mathbb{Z}_0 = \{\mathcal{D}_S\}$. At step $\tau = 1$, the first target domain $\mathcal{D}_T^1$ joins – at this time, only the source domain $\mathcal{D}_S$ has a learned

representation F_S. Therefore, $\mathcal{D}_T^1$ can perform uDA with the source and learn a representation F_T^1.

In general, at step $\tau = K$, $\mathbb{Z}_K = \{\mathcal{D}_S\} \cup \left\{ \mathcal{D}_T^j \middle| j = 1, \ldots, K \right\}$. For a new target domain $\mathcal{D}_T^{K+1}$, the goal of OCS is to find an optimal collaborator domain $\mathcal{D}_{\text{opt}} \in \mathbb{Z}_K$, such that:

$$\mathcal{D}_{\text{opt}} = \underset{i=1\ldots|\mathbb{Z}_K|}{\text{argmin}} \, \Phi(\mathbb{Z}_K^i, \mathcal{D}_T^{K+1})$$

where $\mathbb{Z}_K^i$ is the i^{th} candidate domain in $\mathbb{Z}_K$ and Φ is a metric that quantifies the risk of collaboration between $\mathbb{Z}_K^i$ and $\mathcal{D}_T^{K+1}$. As highlighted in §7.1, the choice of an optimal collaborator brings two key benefits for uDA: firstly, it relaxes the static nature of existing uDA techniques and allows for a more *scalable* architecture. Secondly, as shown in Figure 7.2b, choosing the right adaptation collaborator can boost the adaptation performance significantly.

(ii) Distributed Adversarial Training. Once a collaborator $\mathcal{D}_{\text{opt}}$ is selected, how do we enable distributed uDA training between $\mathcal{D}_{\text{opt}}$ and the target domain $\mathcal{D}_T^{K+1}$, while preventing any data leaks? This opens up two challenges: i) how do we distribute the adversarial uDA network architecture and training process across the nodes? ii) how do we ensure that the gradients exchanged between the distributed nodes during training cannot be used to reconstruct the raw data?

7.3 FRUDA: Framework for Realistic uDA

We first present our two algorithmic contributions on Optimal Collaborator Selection (named OCS) and Distributed uDA (named DILS). Later in §7.3.3, we explain how these two algorithms work in conjunction with each other in an end-to-end framework called FRUDA, and allow for scaling uDA algorithms in multi-domain, distributed ML systems.

7.3.1 Optimal Collaborator Selection (OCS)

For any given target domain, the goal of OCS is to find its optimal collaborator domain $\mathcal{D}_{\text{opt}}$ from a set of candidate domains. Our key idea is quite intuitive: the optimal collaborator should be a domain, such that adapting from it will lead to the highest classification accuracy (or equivalently, the lowest classification error) in the target domain. We first introduce some notations and then present the key theoretical insight that underpins OCS.

Notations. Let $\mathcal{D}$ be a domain on an input space $\mathcal{X}$ and with an associated labeling function $l : \mathcal{X} \to [0,1]$. A hypothesis is a function $h : \mathcal{X} \to [0,1]$. Let $\varepsilon_{M,\mathcal{D}}(h,l)$

denote the error of a hypothesis h w.r.t. l under the domain $\mathcal{D}$, where M is an error metric such as L_1 error or cross-entropy error. Further, a function $f : \mathcal{X} \to \mathbb{R}$ is θ-Lipschitz if it satisfies the inequality

$$\|f(x_1) - f(x_2)\| \leq \theta \|x_1 - x_2\| \text{ for all } x_1, x_2 \in \mathcal{X}$$

where $\theta \in \mathbb{R}_+$. The smallest such θ is called the Lipschitz constant of f.

Theorem 1. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two domains sharing the same labeling function l. Let θ_{CE} denote the Lipschitz constant of the cross-entropy loss function in $\mathcal{D}_1$. For any two θ-Lipschitz hypotheses h, h', we can derive the following error bound for the cross-entropy (CE) error in $\mathcal{D}_2$:*

$$\varepsilon_{\text{CE},\mathcal{D}_2}(h, h') \leq \theta_{\text{CE}} \left(\varepsilon_{\text{L}_1,\mathcal{D}_1}(h, h') + 2\theta W_1(\mathcal{D}_1, \mathcal{D}_2) \right) \tag{7.4}$$

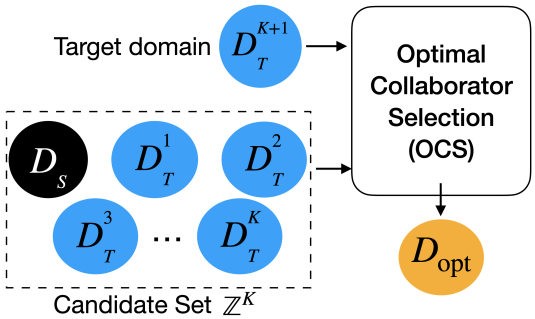
where $W_1(\mathcal{D}_1, \mathcal{D}_2)$ denote the first Wasserstein distance between the domains $\mathcal{D}_1$ and $\mathcal{D}_2$, and $\varepsilon_{\text{L}_1,\mathcal{D}_1}$ denotes the L_1 error in $\mathcal{D}_1$. Proof is in the Appendix.

Theorem 1 has two key properties that make it apt for our problem setting. First, it can be used to get an upper bound estimate of the CE error in a target domain ($\mathcal{D}_2$), given a hypothesis (or a classifier) from a collaborator domain ($\mathcal{D}_1$). Since target CE error is the key metric of interest in classification tasks, this bound is more useful than the one proposed by [125] which estimates the L_1 error in the target domain. Secondly, the bound depends on the Wasserstein distance metric between the domains, which could be computed in a distributed way without exchanging any private data between domains. This property is very important to guarantee the domain privacy, which is one of the objectives of our work. Please refer to the Appendix for implementation details on computation of Wasserstein distance in a distributed and privacy-preserving manner.

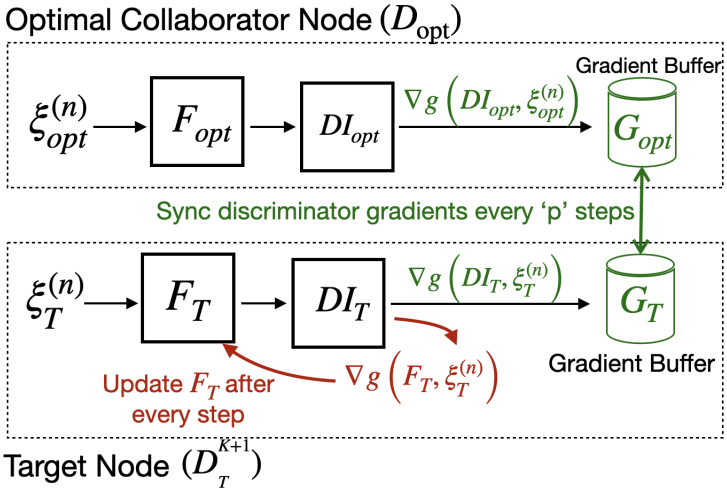
Selecting the optimal collaborator. Motivated by Theorem 1, we now discuss how to select the optimal collaborator for a target domain. Given a collaborator domain $\mathcal{D}_c$, a learned hypothesis h^c and a labeling function l, we can obtain an upper bound estimate of the CE error for a target domain $\mathcal{D}_T$ using Theorem 1 as:

$$\varepsilon_{\text{CE},\mathcal{D}_\text{T}}(h^c, l) \leq \theta_{\text{CE}} (\varepsilon_{\text{L}_1,\mathcal{D}_\text{c}}(h^c, l) + 2\theta W_1(\mathcal{D}_c, \mathcal{D}_T)) \tag{7.5}$$

We can tighten the bound in Eq. 7.5 to get a more reliable estimate of the target CE error. This is achieved by reducing the Lipschitz constant (θ) of the hypothesis h^c during training. In uDA, the hypothesis is parameterized by a neural network,



(a) Optimal Collaborator Selection



(b) DILS: Distributed uDA

Figure 7.3: (a) A new target domain D_T^{K+1} finds its optimal adaptation collaborator D_{opt} from a set of candidate domains. (b) D_T^{K+1} performs distributed uDA with D_{opt} to learn a model for its distribution.

and we can train neural networks with small Lipschitz constants by regularizing the spectral norm of each network layer [201].

Now that we have a way to estimate the target CE error, we can use it to select an optimal collaborator that yields the minimum target CE error. Let $\mathbb{Z} = \{\mathcal{D}^k | k = 1, \ldots, K\}$ be a set of candidate domains each with a pre-trained model h^k with Lipschitz constants θ^k_{CE} and θ^k. Let $\mathcal{D}_T^{K+1}$ be a target domain for which the collaborator is to be chosen. Assuming the availability of a test set to compute the true error in each candidate domain, we use Eq. 7.5 to select the optimal collaborator $\mathcal{D}_{\text{opt}}$ as:

$$\mathcal{D}_{\text{opt}} = \underset{k=1,\ldots,K}{\text{argmin}}\, \theta^k_{\text{CE}}(\varepsilon_{\text{L}_1,\mathcal{D}^k}(h^k, l) + 2\theta^k W_1(\mathcal{D}^k, \mathcal{D}_T^{\text{K+1}})) \tag{7.6}$$

7.3.2 Distributed uDA using DIscriminator-based Lazy Synchronization (DILS)

Upon selecting an optimal collaborator $\mathcal{D}_{\text{opt}}$ for the target domain $\mathcal{D}_T^{K+1}$, the next step is to learn a model for $\mathcal{D}_T^{K+1}$ by doing uDA with $\mathcal{D}_{\text{opt}}$. In line with our problem setting, both domains are located on distributed nodes and cannot share their raw private data with each other.

FADA [202] is a recently proposed technique for *adversarial* uDA in distributed settings. Our solution (DILS) differs from FADA in three important ways: (i) FADA was designed for a federated learning setup and assumes multiple labeled source domains, which is not the case in our setting. (ii) FADA exchanges feature representations of the data and corresponding gradients between nodes to achieve distributed training. Prior works [203, 204, 205] have shown that these are prone to privacy attacks and can be exploited to reconstruct the raw data. Instead, we leverage a unique characteristic of adversarial training architectures and show that adversarial uDA can be performed between distributed nodes only by exchanging the *gradients of domain discriminators*. The biggest benefit of our approach is that it provides protection against state-of-the-art gradient leakage attacks. (iii) FADA exchanges the gradients between nodes after every batch of data, which can increase the overall training time of uDA due to the communication overhead associated with gradient exchange. DILS, instead, adopts a lazy gradient synchronization approach which significantly reduces the uDA training time.

Approach. As shown in Figure 7.3b, we split the adversarial architecture across the distributed nodes. The feature encoders of the collaborator (E_{opt}) and target (E_T) reside on their respective nodes, while the discriminator DI is split into two components DI_{opt} and DI_T. At every training step n, both nodes feed their private training data ξ_{opt}^n and ξ_T^n into their encoders and discriminators, and compute the gradients of the discriminators, i.e., $\nabla g(DI_{\text{opt}}, \xi_{\text{opt}}^n)$ and $\nabla g(DI_T, \xi_T^n)$ respectively.

How often should we exchange the discriminator gradients between nodes? By exchanging discriminator gradients after every training step, we can keep the discriminators synchronized and ensure that distributed training converges to the non-distributed solution. However, this approach has a major downside, as gradient exchange every step incurs significant communication costs and increases the overall uDA training time. To increase training efficiency, we propose a *Lazy Synchronization* approach, wherein instead of every step, the discriminators are synchronized after every p training steps, thereby reducing the total gradient exchange by a factor of p. We denote the training steps at which the synchronization takes place as the

Algorithm 1: DIscriminator-based Lazy Synchronization (DILS)

Result: F_T

Input: Pre-trained F_{opt}; Randomly Initialize DI_{opt}; Initialize $F_T = F_{\text{opt}}$; DI_T=DI_{opt}; Sync up step p; total steps N ;

for $n = 1, 2, ..., N$ **do**

- Sample a batch of data on both nodes, $\xi_{\text{opt}}^{(n)}$ and $\xi_T^{(n)}$;
- Feed $\xi_{\text{opt}}^{(n)}$ and $\xi_T^{(n)}$ into the respective Encoder-Discriminator model separately on both nodes;
- Based on different loss functions, calculate the gradients locally. On collaborator node, calculate $\nabla g(DI_{\text{opt}}, \xi_{\text{opt}}^{(n)})$; On target node, calculate $\nabla g(F_T, \xi_T^{(n)})$ and $\nabla g(DI_T, \xi_T^{(n)})$;
- Add $\nabla g(DI_{\text{opt}}, \xi_{\text{opt}}^{(n)})$ to gradient buffer G_{opt}, add $\nabla g(DI_T, \xi_T^{(n)})$ to target gradients buffer G_T ;
- **if** *isTargetNode* **then**
  - Apply $\nabla g(F_T, \xi_T^{(n)})$ to F_T;
- **if** $n\%p == 0$ **then**
  - Exchange gradients buffer and update the latest synced gradients $g_{\text{sync}} = \frac{G_{\text{opt}} + G_T}{2p}$;
  - Clear G_{opt} and G_T ;
- Apply g_{sync} to DI_{opt} and DI_T separately;

sync-up steps while other steps are called *local steps*. We present our DILS strategy in Algorithm 1.

As evident, DILS works on the principle of using stale gradients (∇b_{sync}) from the last sync-up step to update the discriminators locally. This is particularly important as it prevents the distributed discriminators from diverging in the local steps. The stale gradients are refreshed at every *sync-up* step. Our results show that this approach reduces the uDA training time with minimal effect on target accuracy.

Privacy Analysis. One of the key advantages of DILS is that it provides protection against known privacy attacks. By exchanging information between the distributed nodes using *gradients of the discriminators*, DILS clearly affords privacy benefits over existing uDA algorithms since we no longer have to transmit raw training data between nodes. However, prior works have shown that model gradients can potentially leak raw training data in collaborative learning [206, 205, 204]. Therefore it is critical to examine: *can the exchange of discriminator gradients also indirectly leak training data of a domain*? Through a theoretical analysis of DILS provided in the Appendix, we show that gradient leakage attacks can at best allow an adversary to steal the feature representations of the data, if they get access to all the discrimi-

(a) Samples from the Digits dataset

(b) Samples from the Office-Caltech dataset

Figure 7.4: Illustration of the Digits and Office-Caltech datasets.

nator gradients. Moreover, since our training strategy does not exchange the feature extractor model F_T between the nodes, it is infeasible to reconstruct the raw data from the stolen feature representations.

7.3.3 Combining OCS with DILS

We now discuss how OCS and DILS work together to address the challenges introduced in §7.1. As shown in Figure 7.3a, a new target domain $\mathcal{D}_T^{K+1}$ first performs OCS with all candidate domains in $\mathbb{Z}^K$ to find its optimal collaborator $\mathcal{D}_{\text{opt}}$. This step makes uDA systems more flexible and ensures that each target domain is able to achieve the best possible adaptation accuracy in the given setting. Next, as shown in Figure 7.3b, $\mathcal{D}_T^{K+1}$ and $\mathcal{D}_{\text{opt}}$ use DILS to engage in distributed uDA. This step ensures that no private data is exposed during adaptation and yet the target domain is able to learn a model for its distribution. Finally, the newly adapted target domain $\mathcal{D}_T^{K+1}$ (with its model and unlabeled data) is added to the candidate set $\mathbb{Z}$ to serve as a potential collaborator for future domains.

7.4 Evaluation

Datasets. We evaluate FRUDA on the following image and speech datasets:

- *Rotated MNIST (RMNIST)* as shown in Figure 7.1b is a variant of the MNIST dataset with digits rotated clockwise by different degrees. Each rotation is considered a separate domain.

- *Multi-microphone Spoken Keywords* dataset, as introduced in Chapter 2, contains spoken keywords recorded from three different microphones: Matrix Voice, ReSpeaker and USB. Each microphone represents a separate domain.

- *Digits*, as shown in Figure 7.4a, is a widely-used meta-dataset for benchmarking uDA algorithms in the computer vision literature. It is constructed by combining five different datasets that contain images of digits from 0-9 in various styles. More specifically, it contains images from MNIST (M), Street View House Number or SVHN (S) [207], MNIST-M (MM), USPS (U) and SynNumbers (SYN) [208]. Each sub-dataset is considered a different domain in our experiments.

- *Office-Caltech* is also a vision dataset widely used to benchmark the performance of uDA algorithms. It contains images of office objects from 10 categories, which are collected under four different settings: downloaded from Amazon (A), captured on a DSLR (D) camera, captured on a web camera (W), and borrowed from the Caltech-256 (C) dataset. Each of these image settings is considered a separate domain. Figure 7.4b shows some samples from the Office-Caltech dataset.

General Setup. We follow the same evaluation protocol as earlier uDA works [209, 123]: a pre-trained model learned from the labeled source domain is given, unlabeled data from the source and target domains are used for adversarial adaptation, and the adapted model is evaluated on a held-out test dataset from the target domain. The difference here is that we aim to adapt for multiple target domains, joining sequentially in a random order. Further, we use a small subset (10%) of the training instances for doing collaborator selection. Our system is implemented with TensorFlow 2.0, and we use Message Passing Interface (MPI) as the communication interface between distributed nodes.

7.4.1 Performance of DIscriminator-based Lazy Synchronization (DILS) training

We first evaluate the convergence properties of DILS against two baselines: Non-Distributed uDA and FADA. As discussed in §7.3.2, FADA was originally designed for multiple sources in a federated learning setup. For a fair comparison with our single-source setting, we modify FADA by only implementing its Federated Adversarial Alignment component and setting the number of source domains to one.

Dataset	Order_1	Order_2
RMNIST	**0**,30,60,90,120,150,180, 210,240,270,300,330	**0**,180,210,240,270,300, 330,30,60,90,120,150
Spoken Keywords	**Matrix (M)**, USB (U), ReSpeaker (R)	**USB**, Matrix ReSpeaker
Digits	**MNIST Modified (MM)**, Synth Digits (Syn), MNIST (M), USPS (U), SVHN (S)	**Synth Digits**, MNIST, MNIST Modified, USPS, SVHN
Office-Caltech	**DSLR (D)**, Webcam (W), Caltech (C), Amazon (A)	**Webcam**, Caltech, DSLR, Amazon

Table 7.1: Domain orderings used in our experiments. Domains in bold correspond to the labeled source domain. All other domains are unlabeled and introduced sequentially in the system.

The modified FADA has the same optimization objectives as single-source DA, but it operates in a distributed setting. Hence, it is fair to compare it with DILS. As we discussed earlier, DILS provides *privacy benefits* over the baselines and is robust against gradient leakage attacks. However, two key questions still remain: (i) *Training Time*: to what extent can DILS reduce the training time for distributed uDA? (ii) *Target Accuracy*: can the use of stale gradients and lazy synchronization in DILS degrade the classification accuracy in the target domain?

Results. Table 7.2 shows the performance of DILS against the baselines on 4 datasets and 8 adaptation tasks. First, we look at the mean training times for uDA (denoted as 't' in Table 7.2). As expected, non-distributed uDA, wherein source and target datasets are required to be on the same machine, has the fastest convergence because there is no gradient communication time involved; however this comes at the expense of domain privacy. Importantly, the end-to-end convergence of DILS is 37% faster than FADA in a distributed setting, primarily due to the reduced overhead of gradient communication between nodes. Further, Table 7.2 shows a very promising result that DILS can achieve similar accuracy as the baselines, which confirms our theoretical analysis that lazy synchronization of discriminator gradients does not degrade the target accuracy. Averaged over all adaptation tasks, the accuracy difference between DILS and the baselines is less than 0.5%, and it could be further reduced by choosing a smaller p.

7.4.2 Performance of the proposed framework, FRUDA

Recall that FRUDA comprises of two algorithms: Optimal Collaborator Selection (OCS) and distributed uDA (DILS) algorithms. In §7.4.1, we established that DILS

	RMNIST			Office-Caltech		
Training	**30 → 60**	**150 → 180**	**t** (mins)	**W → C**	**D → A**	**t** (mins)
Source-Only	32.68	61.51	-	87.81	85.28	-
Non-Distributed	69.61	91.86	2.4	91.74	92.01	21
FADA	69.30	91.21	65.28	91.02	92.03	73.8
DILS	68.34	90.16	35.2	90.56	91.77	55.05

	Digits			Spoken Keywords		
Training	**M-M → U**	**Syn → U**	**t** (mins)	**C → U**	**R → C**	**t** (min)
Source-Only	57.54	79.32	-	73.98	67.47	-
Non-Distributed	82.14	90.1	5.69	79.85	77.52	5.8
FADA	82.13	89.9	86.94	79.81	77.39	37.8
DILS	81.66	89.78	54.6	79.33	77.38	22.5

Table 7.2: Target Domain Accuracy and mean uDA Training Time (t). DILS has a 37% faster convergence time than FADA on average, without a major drop in adaptation accuracy. The sync-up step p for DILS is set to 4.

is an effective algorithm for distributed uDA, in terms of training time, data privacy and adaptation accuracy. Now we evaluate how DILS can work in conjunction with OCS to scale uDA in practical settings.

Experiment Setup. In our problem setting, domains appear sequentially in an order. Let $\{\mathbf{D_S}, D_T^1, D_T^2 \cdots D_T^K\}$ denote an ordering of one labeled source domain $\mathbf{D_S}$ and K unlabeled target domains. For each target domain $D_T^i|_{i=1}^K$, we first choose a collaborator domain, which could be either the labeled source domain $\mathbf{D_S}$ or any of the previous target domains $D_T^j|_{j=1}^{i-1}$ that have already learned a model using uDA. Upon choosing a collaborator (using OCS or any of the baseline techniques), we use DILS to perform distributed uDA between the target domain and the collaborator, and compute the post-adaptation test accuracy Acc_T^i in the target domain. We report the mean adaptation accuracy obtained over all target domains, i.e., $\frac{1}{K}\sum_{i=1}^K \text{Acc}_T^i$.

For each dataset, we use two random orderings of source and target domains, e.g., for *Office-Caltech*, we choose Order$_1$ = **D**,W,C,A and Order$_2$ = **W**,C,D,A. Please refer to the Appendix for details on orderings used for other datasets, and our computing infrastructure. The optimization objectives of ADDA presented in Eq. 7.2 and 7.3 are used for adaptation in this experiment.

Collaborator Selection Baselines. We compare OCS against four baselines: (i) *Labeled Source* wherein each target domain only adapts from the labeled source domain; (ii) *Random Collaborator*: each target domain chooses a random collabo-

	RMNIST		Office-Caltech	
	Order_1	Order_2	**D**,W,C,A	**W**,C,D,A
No Adaptation	34.65	35.54	66.40	85.25
Random	28.66±6.50	37.11±4.32	69.18±1.51	80.1±2.44
Labeled Source	47.14 ± 0.85	49.08± 0.75	67.77±0.15	**90.62±0.13**
Multi-Collaborator	40.51±0.30	42.73±0.39	68.90±0.24	82.17±0.87
Proxy A-Distance	93.51 ± 0.22	74.14±0.05	69.37±0.2	**90.62±0.13**
FRUDA (Ours)	**97.08 ± 0.14**	**81.72±0.3**	**74.56±0.52**	**90.62±0.13**

	Digits		Spoken Keywords	
	Order_1	Order_2	Order_1	Order_2
No Adaptation	59.59	72.09	76.45	75.83
Random	62.77±2.19	69.13±4.0	80.17±1.60	77.34±1.09
Labeled Source	64.89±0.23	79.87±0.31	80.86 ± 0.09	79.91±0.05
Multi-Collaborator	60.94±0.13	75.91±0.30	76.90 ± 0.13	79.0±0.24
Proxy A-Distance	70.09±0.45	83.07±0.15	80.34 ± 0.19	80.02±0.31
FRUDA (Ours)	**73.01±0.87**	**85.31±0.26**	**81.43 ± 0.06**	**81.81±0.10**

Table 7.3: Mean accuracy over all target domains appearing in a given order, e.g., Order_1=**D**,W,C,A for Office-Caltech.

rator from the available candidates; (iii) *Proxy A-distance (PAD)* where we choose the domain which has the least PAD [210] from the target; (iv) *Multi-Collaborator* is based on MDAN [211], where all available candidate domains contribute to the adaptation in a weighted-average way. However, MDAN was developed assuming that all candidate domains are labeled, which is not the case in our setting. Hence, we modify MDAN and only optimize its adversarial loss during adaptation (details in the Appendix).

Results. Table 7.3 reports the mean accuracy obtained over the target domains for two domain orderings in each dataset. We observe that FRUDA outperforms the baseline collaborator selection techniques in most cases. Importantly, it can provide significant gains over the *Labeled Source* (LS) baseline, which verifies our hypothesis that the labeled source domain is not *always* optimal for uDA. We highlight three key results from Table 7.3: (a) in RMNIST, FRUDA provides 41% accuracy gains over LS on average — this could be partly attributed to the large number of target domains ($K = 11$) in this dataset. As the number of target domains increase, there are more opportunities for benefiting from collaboration selection, which led to higher accuracy gains over LS in this dataset. (b) For Office-Caltech (order = **W**,C,D,A), the labeled source domain **W** turns out to be the optimal collaborator for all target domains, and FRUDA managed to converge to the Labeled Source

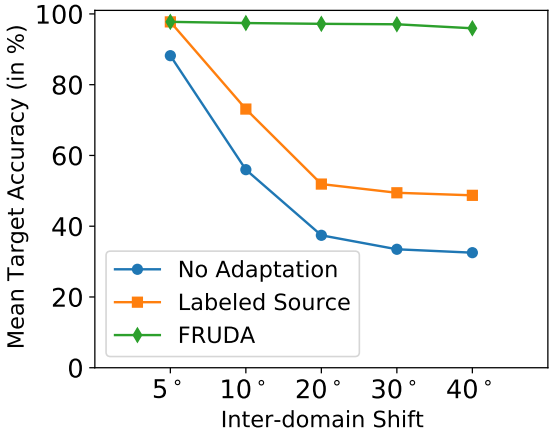


Figure 7.5: Effect of increase in the domain shift on the performance of FRUDA.

baseline. (c) Multi-collaborator baseline often performed worse than even LS. We surmise that this is due to negative transfer caused by some collaborator domains that are too different from the target domain. OCS, on the other hand, is able to filter out these bad collaborators, which leads to higher adaptation accuracy. In general, our results demonstrate that as uDA systems scale to multiple target domains, the need for choosing the right adaptation collaborator becomes important, hence warranting the need for accurate collaboration selection algorithms.

Delving deeper, we evaluate the performance of FRUDA as the divergence between domains increase. We take the RMNIST dataset and vary the inter-domain rotation from 5° (i.e., the domains are **0**°, 5°, 10°, 15°....) to 40° (i.e., the domains are **0**°, 40°, 80°, 120°....). Figure 7.5 reports the mean accuracy across all target domains, and we observe that the performance gains of FRUDA over the Labeled Source baseline increase as the inter-domain divergence increases. This is because as inter-domain rotation increases, the shift between the target domains and labeled source distribution (0°) also increases. Under higher domain shift, the labeled source domain no longer remains the optimal collaborator and the adaptation performance drops significantly. FRUDA, on the other hand, finds an optimal collaborator in each case and prevents accuracy degradation due to a higher domain shift.

The results presented in Table 7.3 used the adversarial loss functions of ADDA from Eq. 7.2 and 7.3 during adaptation. However, FRUDA is intended to be a general framework not limited to one specific uDA algorithm. We now evaluate its performance with three other uDA loss formulations (i) when a Gradient Reversal Layer (GRL) is used to compute the mapping loss [122], (ii) when Wasserstein Distance is used as a loss metric for domain discriminator [125] and (iii) CADA [212] which enforces consensus between source and target features. The optimization objectives

	RMNIST (Order_1)				Digits (Order_1)			
	ADDA	GRL	WassDA	CADA	ADDA	GRL	WassDA	CADA
No Adaptation	34.65	34.65	34.65	34.65	59.59	59.59	59.59	59.59
Labeled Source	47.14	47.26	44.39	41.30	64.89	65.51	70.34	65.22
FRUDA(Ours)	**97.08**	**97.35**	**91.15**	**83.37**	**73.01**	**69.80**	**75.36**	**70.19**

Table 7.4: Mean target accuracy for four uDA methods. Our framework can be used in conjunction with various uDA methods, and improves mean accuracy over the Labeled Source baseline.

for each of these uDA techniques are provided in the Appendix.

In Table 7.4, we observe that while different uDA techniques yield different target accuracies, FRUDA can work in conjunction with all of them to improve the overall accuracy over the *Labeled Source* baseline.

7.5 Related Work

Related work for OCS. There are prior works on computing similarity between domains, e.g., using distance measures such as Maximum Mean Discrepancy [213], Gromov-Wasserstein Discrepancy [214], A-distance [215], and subspace mapping [216]. However, our results in §7.4.2 show that merely choosing the most similar domain as the collaborator is not optimal. Instead, OCS directly estimates the target cross-entropy error for collaborator selection. Another advantage of OCS over prior methods is that it can work in a fully distributed manner without compromising domain privacy.

There are also works on selecting or generating intermediate domains for uDA. [217] studied a setting when source and target domains are too distant (e.g., image and text) which makes direct knowledge transfer infeasible. As a solution, they propose selecting intermediate domains using A-distance and domain complexity. However, as we discussed, merely using distance metrics does not guarantee the most optimal collaborator. Moreover, this work was done on a KNN classifier and did not involve adversarial uDA algorithms. [218] and [219] use style transfer to generate images in intermediate domains between the source and target. Although interesting, these works are orthogonal to OCS in which the goal is to select the best domain from a given set of candidates. Moreover, these works are primarily focused on visual adaptation, while OCS is a general method that can work for any modality. Finally, [220, 221] are techniques for incremental uDA in continuously

shifting domains. However, in our problem, different target domains may not have any inherent continuity in them, and hence it becomes important to perform OCS.

Related work for DILS. There is prior work on *distributed model training*, wherein training data is partitioned across multiple nodes to accelerate training. These methods include centralized aggregation [16, 18], decentralized training [19, 20], and asynchronous training [222]. Similarly, with the goal of preserving data privacy, *Federated Learning* proposes sharing model parameters between distributed nodes instead of the raw data [223, 224]. However, these distributed and federated training techniques are primarily designed for supervised learning and do not *extend directly to uDA architectures*. A notable exception is FADA by [202] which extends uDA to federated learning. As we extensively discussed in § 7.3.2, FADA exchanges the features and gradients of the *feature extractor* between nodes to achieve domain privacy, which are prone to privacy attacks. Instead, DILS operates by exchanging *discriminator gradients* between nodes, which bring both privacy and training-time benefits over FADA.

Related work on practical uDA. [225] and [226] are two very promising recent works on source-dataset-free uDA. Although the scope of these works are different from us, we share the same goal of making uDA techniques more practical. Specifically, we focus on developing general algorithms and frameworks to scale existing uDA approaches in realistic settings.

7.6 Discussion and Limitations

In this section, we discuss the limitations of this work and avenues for future research.

Extending to non-adversarial adaptation algorithms. Although our proposed framework can incorporate different adversarial uDA optimization objectives, we are cognizant that there are other uDA algorithms, such as those based on generative algorithms (e.g., [209]) and those with no adversarial learning component (e.g., [227]) which are currently not compatible with DILS. Future work could explore how such non-adversarial uDA algorithms could be extended for distributed ML systems.

Selecting Multiple Collaborators. Our optimal collaborator selection algorithm currently selects a single optimal collaborator, which is then used to perform pairwise uDA with the target. In the future, our method can be extended to select multiple collaborators that can jointly perform uDA with the target domain.

Scalability Assumption. We assumed that the ML system would scale sequentially, i.e., it will encounter target domains in a given order one at a time. As such, at any given step, we choose an optimal collaborator for the new target domain and perform uDA to learn a target model. However, there could be other ways for ML systems to scale; for example, a system may be deployed in multiple target domains simultaneously and our current approach of step-wise adaptation may not work here.

Security and Privacy. As with any distributed algorithm that sends data over the communication network, our method is also prone to security and privacy hacks. Although our strategy protects against gradient leakage attacks currently known in the literature, we are mindful of the possibility that our technique may not withstand new privacy attacks that will be developed in the future. Hence, we need to be constantly aware of the developments in the ML privacy literature and be ready to make amendments to our proposed technique.

7.7 Summary

This chapter extended the adversarial uDA solutions presented in the thesis in two important ways. Firstly, we showed that when there are multiple target domains in an ML system, there is merit in not always relying on the labeled source domain as the collaborator. Instead, it is better to flexibly choose an adaptation collaborator from all the domains in the system that have learned a prediction model. To this end, we proposed an Optimal Collaborator Selection algorithm, which finds an optimal collaborator for each target domain based on a collaborator's in-domain cross-entropy error and the Wasserstein distance between the collaborator and target domains. Our second contribution was in proposing a distributed adaptation algorithm, DILS, using which we can perform uDA even when the source and target datasets are geographically distributed, and it is not feasible to exchange raw data between them due to privacy reasons. The core idea in DILS is to perform uDA using lazy synchronization of the gradients of the domain discriminator between the distributed nodes, which not only results in similar accuracies as the non-distributed uDA, but also provides substantial speedup in convergence time and prevents gradient-based privacy attacks.

Chapter 8

Concluding Remarks

This thesis investigated the challenges associated with heterogeneity in sensor data and their impact on the scalability of machine learning systems to new domains. The research questions were mainly inspired by the practical scenarios in which ML systems are likely to be deployed, and a number of unsupervised domain adaptation solutions were proposed to counter the domain shift induced by sensing heterogeneities.

Below we summarize our key contributions, discuss the limitations of our proposed solutions, and highlight the avenues for future work on this topic.

8.1 Summary of Contributions

The main contributions of this thesis are as follows:

- In Chapter 3, we quantified the effect of microphone-induced heterogeneity on speech classification models by collecting large-scale multi-microphone datasets. It was one of the first systematic investigations of how microphone variability across off-the-shelf consumer-scale microphone devices can degrade the scalability of deep learning based speech models. The datasets developed towards this effort have been released in public and have been used in other research investigations [228]. We also evaluated the impact of domain shift induced by variations in IMU sensors and their placement on the human body. Our empirical results show that the performance of human activity recognition (HAR) models degrade drastically under these domain shifts.

- Looking at the problem of domain shift from the perspective of ML systems, we proposed adaptation algorithms for both opaque and transparent ML systems. In Chapter 4, we study *opaque ML systems* wherein the parameters of source domain prediction model are not accessible during adaptation. We presented Mic2Mic,

our solution to train a domain translation model using unlabeled and unpaired data from both domains. Our results show that by incorporating this translation model as a component in the inference pipelines of speech-based ML systems, we can recover a significant percentage of the accuracy that is otherwise lost due to microphone heterogeneity.

- In Chapter 5, we presented a uDA approach for *transparent ML systems* that modifies the parameters of the source prediction model during adaptation. Our solution is based on data-augmented adversarial training and outperforms a number of baseline uDA algorithms on the task of adapting HAR models to new domains. This solution paves the way for developing HAR systems that are more robust to the device usage preferences of end-users.

- In Chapter 6, we proposed a solution to relax a common, but limiting, assumption in adversarial uDA algorithms that the label spaces of source and target domains are identical. In practical ML systems where we have no control over the data collected in the target domain, this assumption does not hold. Our proposed algorithm dynamically assigns importance weights to samples from shared and private classes in each domain, prevents negative transfer during adaptation and also boosts the accuracy of the classes that are shared between the domains.

- Finally, Chapter 7 studies the case when the source and target domains are geographically distributed, and exchanging data between them could be expensive or may have privacy implications. This challenge is often overlooked in the ML literature but is critical to the success of unsupervised domain adaptation algorithms in practice. We proposed a distributed uDA approach, DILS, that allows for adversarial training to proceed without requiring the exchange of raw data between domains. Our second contribution in this chapter pertained to scaling uDA to ML systems with multiple target domains, and we presented an algorithm called OCS to select the optimal adaptation collaborator for each domain. Finally, we proposed FRUDA, an end-to-end uDA framework in which OCS and DILS work in conjunction to increase the adaptation performance of a multi-target ML system, while preserving each domain's privacy.

8.2 Limitations and Future Work

In Chapters 4,5,6 and 7, we highlighted the limitations of our proposed solutions. Below we list some additional limitations of the broader methodology adopted in this thesis, along with identifying topics for future work.

Scalability of pairwise adaptations. All our proposed solutions are designed for pair-wise adaptation. In other words, we assume that for each target domain in which an ML system in deployed, we need to learn a model by adapting from a source domain. A limitation of this approach is that when ML systems scale to a large number of target domains (e.g., to hundreds of microphones or IMU devices), we will have to repeat the adaptation step once for each target domain. This raises two challenges: (i) we need to collect *unlabeled* training data from each target domain to facilitate uDA, (ii) the aggregated training cost of uDA over all the target domains will become too high, which will have direct implications on the carbon footprint of ML systems.

We see two potential solutions to this issue. First, we can investigate unsupervised *multi-target* domain adaptation [229] which is a relatively unexplored topic in uDA. In this approach, data samples from multiple target domains are used in combination with the source domain samples to learn domain-invariant feature representations. Domain Generalization (DG) approaches are another promising alternative – here the goal is to train a model on multiple labeled source domains such that it can perform well on any unseen target domains out-of-the-box, without requiring any adaptation [230]. Although both these approaches have their own constraints (e.g., DG requires labeled data from multiple domains), they would clearly reduce the training costs associated with uDA in a multi-domain system.

Monitoring and Detecting Domain Shift. Before any of our proposed uDA solutions could be employed, the ML system needs to detect the occurrence of domain shift. We did not study the detection problem in the thesis, however it is an equally important piece of the puzzle. Fortunately, there are a number of works in the literature that have discussed and proposed solutions for monitoring [34, 33, 35] and detecting domain shifts [36, 175].

Availability of Source Domain Data. We assumed that source domain data is available during adaptation. While it is a reasonable assumption when the source model developer themselves want to adapt a model to new domains, there might be scenarios when this assumption is not practical. Particularly, in opaque ML systems, the model developer will only expose the trained model through an API and may not provide any data samples from the source domain. Hence, the ability to perform uDA in the absence of source data becomes critical in these settings. [231] is an early-stage visual recognition solution in this direction which could be extended for IMU and speech data.

Continual adaptation of ML systems. We assumed that the target domains are sta-

tionary, however it is plausible that they evolve continuously. Consider a camera-based recognition system for detecting the presence of tigers in a jungle. Let us assume that this system was trained in sunny weather conditions (source domain) using supervised learning. When this system is deployed in real-world, it will encounter different weather conditions such as light rain, heavy rain and snow (i.e., target domains). In this setting, one approach is to consider each of these domains as independent and employ the pair-wise adaptation algorithms proposed in this thesis to train a model for each of them. Alternatively, it can be argued that these target domains (light rain, heavy rain, snow) have some underlying relation and as such, an incremental adaptation approach is more appropriate which goes from Sunny $\rightarrow$ Light Rain $\rightarrow$ Heavy Rain $\rightarrow$ Snow, while at the same time ensuring that there is no catastrophic forgetting in the past domains. [220] is an interesting work in this direction, which could be extended for speech and HAR models.

Scaling ML systems using semi-supervised domain adaptation. The solutions developed in this thesis to scale ML models to new domains assumed that there is no labeled data available in the target domain. In some sense, this is the most challenging scenario for adapting models across domains. Instead, if we assume that the target domain contains a small amount of labeled data, we can design semi-supervised adaptation algorithms. This setting has been studied in the literature under the name of *few-shot adaptation* [178] and has shown promising results on visual recognition tasks. Similar semi-supervised adaptation techniques can be explored for microphone and IMU-induced domain shifts in speech and HAR models.

Resource-efficient uDA. ML systems are being actively deployed on edge devices with limited memory and computational capabilities. Therefore, it is important that both the training and deployment of uDA solutions is resource-efficient. We plan to investigate an interesting research question in future work: during uDA, in addition to adapting the parameters of a source domain neural network, can we also jointly learn a small-footprint neural network architecture for the target domain? While the former is important to enhance the performance of the source model in the target domain, the latter could improve the resource-efficiency of the target domain model.

Bibliography

[1] Timo Sztyler and Heiner Stuckenschmidt. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE Computer Society, 2016. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7456521.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

[4] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[5] Jason Wu, Sayan Ghosh, Mathieu Chollet, Steven Ly, Sharon Mozgai, and Stefan Scherer. Nadia: Neural network driven virtual human conversation agents. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, IVA '18, page 173–178, New York, NY, USA, 2018. Association for Computing Machinery.

[6] Basheer Qolomany, Ala Al-Fuqaha, Ajay Gupta, Driss Benhaddou, Safaa Alwajidi, Junaid Qadir, and Alvis C Fong. Leveraging machine learning and big data for smart buildings: A comprehensive survey. *IEEE Access*, 7:90316–90356, 2019.

[7] Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(2):1–16, 2018.

[8] Powerful computer vision algorithms are now small enough to run on your phone. https://www.technologyreview.com/2019/10/11/102546/ai-computer-vision-algorithms-on-your-phone-mit-ibm/, 2019.

[9] Abhinav Mehrotra and Mirco Musolesi. Intelligent notification systems. *Synthesis Lectures on Mobile and Pervasive Computing*, 11(1):1–75, 2020.

[10] Deep Learning on Android P. https://deepmind.com/blog/deepmind-meet-android/, 2018.

[11] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[12] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems*, pages 9453–9463, 2019.

[13] ImageNet Dataset. http://www.image-net.org/, 2015.

[14] Google TPU Pods. https://cloud.google.com/tpu/docs/training-on-tpu-pods, 2020.

[15] Yang You, Aydın Buluç, and James Demmel. Scaling deep learning on gpu and knights landing clusters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2017.

[16] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14)*, pages 583–598, 2014.

[17] Wei Dai, Abhimanu Kumar, Jinliang Wei, Qirong Ho, Garth Gibson, and Eric P Xing. High-performance distributed ml at scale through parameter server consistency models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[18] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[19] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

[20] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D2: Decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*, 2018.

[21] Now anyone can train Imagenet in 18 minutes. https://www.fast.ai/2018/08/10/fastai-diu-imagenet/, 2018.

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

[23] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé M Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML*, 2019.

[24] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1175–1191, New York, NY, USA, 2017. ACM.

[25] TensorFlow Serving. https://www.tensorflow.org/tfx/guide/serving, 2020.

[26] TorchServe. https://github.com/pytorch/serve, 2020.

[27] MLFlow Serving. https://www.mlflow.org/docs/latest/index.html, 2020.

[28] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017.

[29] Nicholas D Lane, Sourav Bhattacharya, Akhil Mathur, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3):82–88, 2017.

[30] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12. IEEE, 2016.

[31] Pete Warden and Daniel Situnayake. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, Incorporated, 2020.

[32] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*, pages 2503–2511, 2015.

[33] Neil D Lawrence. Data science and digital systems: The 3ds of machine learning systems design. *arXiv preprint arXiv:1903.11241*, 2019.

[34] Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D Sculley. What's your ml test score? a rubric for ml production systems. 2016.

[35] Tom Diethe, Tom Borchert, Eno Thereska, Borja Balle, and Neil Lawrence. Continual learning in practice. *arXiv preprint arXiv:1903.05202*, 2019.

[36] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. In *Advances in Neural Information Processing Systems*, pages 1396–1408, 2019.

[37] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[38] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[39] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*, pages 2712–2721, 2016.

[40] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[41] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

[42] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2018.

[43] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*. Citeseer, 2014.

[44] Anupam Das, Nikita Borisov, and Matthew Caesar. Fingerprinting smart devices through embedded acoustic components. *arXiv preprint arXiv:1403.3366*, 2014.

[45] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 127–140, New York, NY, USA, 2015. Association for Computing Machinery.

[46] ReSpeaker. http://wiki.seeedstudio.com/ReSpeaker_Mic_Array/, 2018.

[47] DIY hardware to emulate Amazon Echo. http://www.instructables.com/id/Build-DIY-Amazon-Alexa-With-a-MATRIX-Creator-on-Ha/, 2015.

[48] Ivan Tashev. Beamformer sensitivity to microphone manufacturing tolerances. 2005.

[49] Shengkui Zhao, Xiong Xiao, Zhaofeng Zhang, Thi Ngoc Tho Nguyen, Xionghu Zhong, Bo Ren, Longbiao Wang, Douglas L Jones, Eng Siong Chng, and Haizhou Li. Robust speech recognition using beamforming with adaptive microphone gains and multichannel noise reduction. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 460–467. IEEE, 2015.

[50] Andreas Grammenos, Cecilia Mascolo, and Jon Crowcroft. You are sensing, but are you biased?: A user unaided sensor calibration approach for mobile sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(1):11:1–11:26, March 2018.

[51] Saurabh Garg, Kian Meng Lim, and Heow Pueh Lee. An averaging method for accurately calibrating smartphone microphones for environmental noise measurement. *Applied Acoustics*, 143:222–228, 2019.

[52] Amos Storkey. When training and test sets are different: characterizing learning transfer.

[53] Akhil Mathur, Fahim Kawsar, Nadia Berthouze, and Nicholas D Lane. Libri-adapt: a new speech dataset for unsupervised domain adaptation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7439–7443. IEEE, 2020.

[54] Akhil Mathur, Anton Isopoussu, Fahim Kawsar, Nadia Berthouze, and Nicholas D Lane. Mic2mic: using cycle-consistent generative adversarial networks to overcome microphone variability in speech systems. In *ACM IPSN 2019*, IPSN '19, pages 169–180, New York, NY, USA, 2019. ACM.

[55] Akhil Mathur, Anton Isopoussu, Fahim Kawsar, Robert Smith, Nicholas D. Lane, and Nadia Berthouze. On robustness of cloud speech apis: An early characterization. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous*

Computing and Wearable Computers, UbiComp '18, page 1409–1413, New York, NY, USA, 2018. Association for Computing Machinery.

[56] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–30, 2020.

[57] Akhil Mathur, Fahim Kawsar, Nadia Berthouze, and Nicholas D Lane. Unsupervised domain adaptation under label space mismatch for speech classification. In *Proceedings of Interspeech*, 2020.

[58] Akhil Mathur, Shaoduo Gan, Anton Isopoussu, Fahim Kawsar, Nadia Berthouze, and Nicholas D Lane. Scaling unsupervised domain adaptation through optimal collaborator selection and lazy discriminator synchronization. *ArXiv e-prints*, 2020.

[59] Oliver Amft, Mathias Stäger, Paul Lukowicz, and Gerhard Tröster. Analysis of chewing sounds for dietary monitoring. In *International Conference on Ubiquitous Computing*, pages 56–72. Springer, 2005.

[60] Oliver Amft and Gerhard Troster. Methods for detection and classification of normal swallowing from muscle activation and sound. In *Pervasive Health Conference and Workshops, 2006*, pages 1–10. IEEE, 2006.

[61] Chenren Xu, Sugang Li, Gang Liu, Yanyong Zhang, Emiliano Miluzzo, Yih-Farn Chen, Jun Li, and Bernhard Firner. Crowd++: unsupervised speaker count with smartphones. In *Ubicomp '13*, pages 43–52. ACM, 2013.

[62] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018. ACM, 2015.

[63] Christos-Nikolaos Anagnostopoulos, Theodoros Iliou, and Ioannis Giannoukos. Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011. *Artificial Intelligence Review*, 43(2):155–177, 2015.

[64] Youngki Lee, Chulhong Min, Chanyou Hwang, Jaeung Lee, Inseok Hwang, Younghyun Ju, Chungkuk Yoo, Miri Moon, Uichin Lee, and Junehwa Song. Sociophone: Everyday face-to-face interaction monitoring platform using multi-phone sensor fusion. In *Proceeding of the 11th annual international*

conference on Mobile systems, applications, and services, pages 375–388, 2013.

[65] Wai-Tian Tan, Mary Baker, Bowon Lee, and Ramin Samadani. The sound of silence. In *Sensys '13*, page 19. ACM, 2013.

[66] Tara N Sainath and Carolina Parada. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[67] Fang Zheng, Guoliang Zhang, and Zhanjiang Song. Comparison of different implementations of mfcc. *Journal of Computer science and Technology*, 16(6):582–589, 2001.

[68] Douglas A Lyon. The discrete fourier transform, part 4: spectral leakage. *Journal of object technology*, 8(7), 2009.

[69] Abdul Malik Badshah, Jamil Ahmad, Nasir Rahim, and Sung Wook Baik. Speech emotion recognition from spectrograms with deep convolutional neural network. In *IEEE PlatCon 2017*, pages 1–5. IEEE, 2017.

[70] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha. Temporal convolution for real-time keyword spotting on mobile devices. *arXiv preprint arXiv:1904.03814*, 2019.

[71] Jianfeng Zhao, Xia Mao, and Lijiang Chen. Speech emotion recognition using deep 1d & 2d cnn lstm networks. *Biomedical Signal Processing and Control*, 47:312–323, 2019.

[72] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[73] Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *arXiv preprint arXiv:1408.2873*, 2014.

[74] Fahim Kawsar, Chulhong Min, Akhil Mathur, and Allesandro Montanari. Earables for personal-scale behavior analytics. *IEEE Pervasive Computing*, 17(3):83–89, 2018.

[75] North smart glasses. https://www.bynorth.com/, 2020.

[76] Jérémy Frey, May Grabli, Ronit Slyper, and Jessica R Cauchard. Breeze: Sharing biofeedback through wearable technologies. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[77] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33, 2014.

[78] Nils Y Hammerla, Reuben Kirkham, Peter Andras, and Thomas Ploetz. On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In *Proceedings of the 2013 international symposium on wearable computers*, pages 65–68, 2013.

[79] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1533–1540, 2016.

[80] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244, 2016.

[81] Shuochao Yao, Yiran Zhao, Huajie Shao, Dongxin Liu, Shengzhong Liu, Yifan Hao, Ailing Piao, Shaohan Hu, Su Lu, and Tarek F Abdelzaher. Sadeepsense: Self-attention deep learning framework for heterogeneous on-device sensors in internet of things applications. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1243–1251. IEEE, 2019.

[82] Yuwen Chen, Kunhua Zhong, Ju Zhang, Qilong Sun, and Xueliang Zhao. Lstm networks for mobile human activity recognition. In *2016 International Conference on Artificial Intelligence: Technologies and Applications*. Atlantis Press, 2016.

[83] Yu Guan and Thomas Plötz. Ensembles of deep lstm learners for activity recognition using wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(2):1–28, 2017.

[84] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2010.

[85] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59. IEEE, 2013.

[86] Patrick Cardinal, Najim Dehak, Yu Zhang, and James Glass. Speaker adaptation using the i-vector technique for bottleneck features. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[87] Andrew Senior and Ignacio Lopez-Moreno. Improving dnn speaker independence with i-vector inputs. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 225–229. IEEE, 2014.

[88] Yohan Chon, Nicholas D Lane, Yunjong Kim, Feng Zhao, and Hojung Cha. Understanding the coverage and scalability of place-centric crowdsensing. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 3–12. ACM, 2013.

[89] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T Chittaranjan, Andrew T Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. Stresssense: Detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 351–360. ACM, 2012.

[90] Saeed V. Vaseghi. *Spectral Subtraction*, pages 242–260. Vieweg+Teubner Verlag, Wiesbaden, 1996.

[91] 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*, 32(6):1109–1121, 1984.

[92] Daniel Michelsanti and Zheng-Hua Tan. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. *arXiv preprint arXiv:1709.01703*, pages 2008–2012, 2017.

[93] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur. A study on data augmentation of reverberant speech for robust

speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5220–5224. IEEE, 2017.

[94] Chanwoo Kim, Ananya Misra, Kean Chin, Thad Hughes, Arun Narayanan, Tara Sainath, and Michiel Bacchiani. Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home. 2017.

[95] Hu Hu, Tian Tan, and Yanmin Qian. Generative adversarial networks based data augmentation for noise robust speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5044–5048. IEEE, 2018.

[96] Jinyu Li, Michael L Seltzer, Xi Wang, Rui Zhao, and Yifan Gong. Large-scale domain adaptation via teacher-student learning. *arXiv preprint arXiv:1708.05466*, 2017.

[97] Matrix Voice. https://www.matrix.one/products/voice, 2019.

[98] Tara N Sainath, Ron J Weiss, Kevin W Wilson, Bo Li, Arun Narayanan, Ehsan Variani, Michiel Bacchiani, Izhak Shafran, Andrew Senior, Kean Chin, et al. Multichannel signal processing with deep neural networks for automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(5):965–979, 2017.

[99] Bo Li, Tara N Sainath, Ron J Weiss, Kevin W Wilson, and Michiel Bacchiani. Neural network adaptive beamforming for robust multichannel speech recognition. *Interspeech 2016*, pages 1976–1980, 2016.

[100] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[101] Gary Mitchell Weiss and Jeffrey Lockhart. The impact of personalization on smartphone-based activity recognition. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. Citeseer, 2012.

[102] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*, pages 1–17. Springer, 2004.

[103] Andrea Mannini and Angelo Maria Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2):1154–1175, 2010.

[104] Nirmalya Roy, Archan Misra, and Diane Cook. Infrastructure-assisted smartphone-based adl recognition in multi-inhabitant smart environments. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 38–46. IEEE, 2013.

[105] Wan-Yu Deng, Qing-Hua Zheng, and Zhong-Min Wang. Cross-person activity recognition using reduced kernel extreme learning machine. *Neural Networks*, 53:1–7, 2014.

[106] Yazan Al Jeroudi, MA Ali, Marsad Latief, and Rini Akmeliawati. Online sequential extreme learning machine algorithm based human activity recognition using inertial data. In *2015 10th Asian Control Conference (ASCC)*, pages 1–6. IEEE, 2015.

[107] Attila Reiss and Didier Stricker. Personalized mobile physical activity recognition. In *Proceedings of the 2013 international symposium on wearable computers*, pages 25–28, 2013.

[108] Lina Yao, Feiping Nie, Quan Z Sheng, Tao Gu, Xue Li, and Sen Wang. Learning from less for better: semi-supervised activity recognition via shared structure discovery. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 13–24, 2016.

[109] Hande Alemdar, Tim LM van Kasteren, and Cem Ersoy. Using active learning to allow activity recognition on a large scale. In *International Joint Conference on Ambient Intelligence*, pages 105–114. Springer, 2011.

[110] Craig Hillman, Cheryl Tulkoff, and DfR Solutions. Manufacturing and reliability challenges with qfn.

[111] P Aggarwal, Z Syed, X Niu, and N El-Sheimy. A standard testing and calibration procedure for low cost mems inertial sensors and units. *The Journal of Navigation*, 61(2):323–336, 2008.

[112] Iuri Frosio, Federico Pedersini, and N Alberto Borghese. Autocalibration of mems accelerometers. *IEEE Transactions on Instrumentation and Measurement*, 58(6):2034–2041, 2008.

[113] Shashi Poddar, Vipan Kumar, and Amod Kumar. A comprehensive overview of inertial sensor calibration techniques. *Journal of Dynamic Systems, Measurement, and Control*, 139(1), 2017.

[114] Louis Atallah, Benny Lo, Rachel King, and Guang-Zhong Yang. Sensor positioning for activity recognition using wearable accelerometers. *IEEE transactions on biomedical circuits and systems*, 5(4):320–329, 2011.

[115] Henar Martín, Ana M Bernardos, Josué Iglesias, and José R Casar. Activity logging using lightweight classification techniques in mobile devices. *Personal and ubiquitous computing*, 17(4):675–695, 2013.

[116] Apiwat Henpraserttae, Surapa Thiemjarus, and Sanparith Marukatat. Accurate activity recognition using a mobile phone regardless of device orientation and location. In *2011 International Conference on Body Sensor Networks*, pages 41–46. IEEE, 2011.

[117] Gabriela Csurka. A comprehensive survey on domain adaptation for visual applications. In *Domain adaptation in computer vision applications*, pages 1–35. Springer, 2017.

[118] Wouter M Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

[119] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[120] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[121] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[122] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[123] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[124] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2018.

[125] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[126] Pavel Denisov, Ngoc Thang Vu, and Marc Ferras Font. Unsupervised domain adaptation by adversarial learning for robust speech recognition. In *Speech Communication; 13th ITG-Symposium*, pages 1–5. VDE, 2018.

[127] Qing Wang, Wei Rao, Sining Sun, Leib Xie, Eng Siong Chng, and Haizhou Li. Unsupervised domain adaptation via domain adversarial training for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4889–4893. IEEE, 2018.

[128] Mohammed Abdelwahab and Carlos Busso. Domain adversarial for acoustic emotion recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(12):2423–2435, December 2018.

[129] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 16–23. IEEE, 2017.

[130] Sining Sun, Binbin Zhang, Lei Xie, and Yanning Zhang. An unsupervised deep domain adaptation approach for robust speech recognition. *Neurocomputing*, 257:79–87, 2017.

[131] Md Abdullah Al Hafiz Khan, Nirmalya Roy, and Archan Misra. Scaling human activity recognition via deep learning-based domain adaptation. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE, 2018.

[132] Ali Akbari and Roozbeh Jafari. Transferring activity recognition models for new wearable sensors with deep generative domain adaptation. In *Proceed-*

ings of the 18th International Conference on Information Processing in Sensor Networks, pages 85–96, 2019.

[133] Timo Sztyler. *Sensor-based human activity recognition: Overcoming issues in a real world setting*. PhD thesis, Mannheim, 2019.

[134] Google Speech API. https://cloud.google.com/speech/, 2015.

[135] Bing Speech API. https://azure.microsoft.com/en-us/services/cognitive-services/speech/.

[136] Amazon Transcribe. https://aws.amazon.com/transcribe/.

[137] Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. The third 'chime'speech separation and recognition challenge: Dataset, task and baselines. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015.

[138] MQTT Messaging Protocol. http://mqtt.org/, 2019.

[139] JBL LSR305 & LSR308 measurements. https://docs.google.com/document/d/1T9yLUksyFTu8DwtsaUwacoCIpRfmdIzx6vDRPjzfbCg/mobilebasic, 2017.

[140] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[141] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *ICASSP*, pages 5206–5210. IEEE, 2015.

[142] Pre-trained Mozilla DeepSpeech2 ASR model. https://github.com/mozilla/DeepSpeech/releases/tag/v0.5.0, 2019.

[143] How google translate squeezes deep learning onto a phone. http://googleresearch.blogspot.co.uk/2015/07/how-google-translate-squeezes-deep.html, 2015. Accessed: 2016-04-10.

[144] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, et al. Collecting complex activity datasets in highly rich networked

sensor environments. In *2010 Seventh international conference on networked sensing systems (INSS)*, pages 233–240. IEEE, 2010.

[145] Bandar Almaslukh, Abdel Artoli, and Jalal Al-Muhtadi. A robust deep learning approach for position-independent smartphone-based human activity recognition. *Sensors*, 18(11):3726, 2018.

[146] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[147] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932, 2017.

[148] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[149] Andrew L Maas, Quoc V Le, Tyler M O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. Recurrent neural networks for noise reduction in robust asr. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[150] Jun Du, Qing Wang, Tian Gao, Yong Xu, Li-Rong Dai, and Chin-Hui Lee. Robust speech recognition with speech enhanced deep neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[151] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[152] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2223–2232, 2017.

[153] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on*

Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.

[154] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[155] Mozilla DeepSpeech2. https://hacks.mozilla.org/2017/11/a-journey-to-10-word-error-rate/, 2017.

[156] Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5), 2018.

[157] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *In Proceedings of ICLR 2019*, 2019.

[158] Pouya Samangouei, Ardavan Saeedi, Liam Nakagawa, and Nathan Silberman. Explaingan: Model explanation via decision boundary crossing transformations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–681, 2018.

[159] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *arXiv preprint arXiv:1711.09020*, 2017.

[160] Sourav Bhattacharya and Nicholas D Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of Sensys '16*, pages 176–189. ACM, 2016.

[161] Cheng Chen, Qi Dou, Hao Chen, and Pheng-Ann Heng. Semantic-aware generative adversarial nets for unsupervised domain adaptation in chest x-ray segmentation. In *International workshop on machine learning in medical imaging*, pages 143–151. Springer, 2018.

[162] Phani Sankar Nidadavolu, Jesús Villalba, and Najim Dehak. Cycle-gans for domain adaptation of acoustic features for speaker recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6206–6210. IEEE, 2019.

[163] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[164] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2208–2217. JMLR. org, 2017.

[165] ZF Syed, P Aggarwal, C Goodall, X Niu, and N El-Sheimy. A new multi-position calibration method for mems inertial navigation systems. *Measurement Science and Technology*, 18(7):1897, 2007.

[166] Chulhong Min, Akhil Mathur, Alessandro Montanari, and Fahim Kawsar. An early characterisation of wearing variability on motion signals for wearables. In *Proceedings of the 23rd International Symposium on Wearable Computers*, pages 166–168, 2019.

[167] Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 216–220, 2017.

[168] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *IJCAI*, 2016.

[169] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net, 2014.

[170] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert MÃžller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May):985–1005, 2007.

[171] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018.

[172] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[173] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *Proc. Interspeech 2019*, pages 2613–2617, 2019.

[174] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[175] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International Conference on Machine Learning*, pages 3122–3130, 2018.

[176] Yitong Li, Michael Murias, Samantha Major, Geraldine Dawson, and David Carlson. On target shift in adversarial domain adaptation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 616–625, 2019.

[177] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoff Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *arXiv preprint arXiv:2003.04475*, 2020.

[178] Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 6670–6680, 2017.

[179] Lin Chen, Wen Li, and Dong Xu. Recognizing rgb images by learning from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1418–1425, 2014.

[180] Wen Li, Lin Chen, Dong Xu, and Luc Van Gool. Visual recognition in rgb images and videos by learning from rgb-d data. *IEEE transactions on pattern analysis and machine intelligence*, 40(8):2030–2036, 2017.

[181] Yuan Yao, Yu Zhang, Xutao Li, and Yunming Ye. Heterogeneous domain adaptation via soft transfer network. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1578–1586, 2019.

[182] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[183] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, pages 427–436, 2015.

[184] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[185] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2724–2732, 2018.

[186] Jing Zhang, Zewei Ding, Wanqing Li, and Philip Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8156–8164, 2018.

[187] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[188] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[189] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Qiang Yang. Separate to adapt: Open set domain adaptation via progressive separation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2019.

[190] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2720–2729, 2019.

[191] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[192] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013.

[193] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997, 2017.

[194] Houwei Cao, David G Cooper, Michael K Keutmann, Ruben C Gur, Ani Nenkova, and Ragini Verma. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE transactions on affective computing*, 5(4):377–390, 2014.

[195] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532, 2019.

[196] Chuansheng Zheng, Xianbo Deng, Qing Fu, Qiang Zhou, Jiapei Feng, Hui Ma, Wenyu Liu, and Xinggang Wang. Deep learning-based detection for covid-19 from chest ct using weak label. *medRxiv*, 2020.

[197] Ying Song, Shuangjia Zheng, Liang Li, Xiang Zhang, Xiaodong Zhang, Ziwang Huang, Jianwen Chen, Huiying Zhao, Yusheng Jie, Ruixuan Wang, et al. Deep learning enables accurate diagnosis of novel coronavirus (covid-19) with ct images. *medRxiv*, 2020.

[198] Fei Shan, Yaozong Gao, Jun Wang, Weiya Shi, Nannan Shi, Miaofei Han, Zhong Xue, and Yuxin Shi. Lung infection quantification of covid-19 in ct images with deep learning. *arXiv preprint arXiv:2003.04655*, 2020.

[199] Jinyu Zhao, Yichen Zhang, Xuehai He, and Pengtao Xie. Covid-ct-dataset: a ct scan dataset about covid-19. *arXiv preprint arXiv:2003.13865*, 2020.

[200] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[201] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael Cree. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.

[202] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *ICLR*, 2020.

[203] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. Nopeek: Information leakage reduction to share activations in distributed deep learning. *arXiv preprint arXiv:2008.09161*, 2020.

[204] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE*

Transactions on Information Forensics and Security, 13(5):1333–1345, 2017.

[205] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, pages 14747–14756, 2019.

[206] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[207] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[208] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[209] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1994–2003, 2018.

[210] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[211] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, pages 8559–8570, 2018.

[212] Han Zou, Yuxun Zhou, Jianfei Yang, Huihan Liu, Hari Prasanna Das, and Costas J Spanos. Consensus adversarial domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5997–6004, 2019.

[213] Jindong Wang, Yiqiang Chen, Wenjie Feng, Han Yu, Meiyu Huang, and Qiang Yang. Transfer learning with dynamic distribution adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(1):1–25, 2020.

[214] Yuguang Yan, Wen Li, Hanrui Wu, Huaqing Min, Mingkui Tan, and Qingyao Wu. Semi-supervised optimal transport for heterogeneous domain adaptation.

[215] Jindong Wang, Vincent W Zheng, Yiqiang Chen, and Meiyu Huang. Deep transfer learning for cross-domain activity recognition. In *proceedings of the 3rd International Conference on Crowd Science and Engineering*, pages 1–8, 2018.

[216] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012.

[217] Ben Tan, Yangqiu Song, Erheng Zhong, and Qiang Yang. Transitive transfer learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1155–1164, 2015.

[218] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2477–2486, 2019.

[219] Jongwon Choi, Youngjoon Choi, Jihoon Kim, Jin-Yeop Chang, Ilhwan Kwon, Youngjune Gwon, and Seungjai Min. Visual domain adaptation by consensus-based transfer to intermediate domain. In *AAAI*, pages 10655–10662, 2020.

[220] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. In *2018 IEEE International conference on robotics and automation (ICRA)*, pages 1–9. IEEE, 2018.

[221] Andreea Bobu, Eric Tzeng, Judy Hoffman, and Trevor Darrell. Adapting to continuously shifting domains. 2018.

[222] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*, 2017.

[223] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[224] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12, 2019.

[225] Jogendra Nath Kundu, Naveen Venkat, Ambareesh Revanur, R Venkatesh Babu, et al. Towards inheritable models for open-set domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12376–12385, 2020.

[226] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. *arXiv preprint arXiv:2002.08546*, 2020.

[227] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pages 153–171. Springer, 2017.

[228] Chulhong Min, Alessandro Montanari, Akhil Mathur, and Fahim Kawsar. A closer look at quality-aware runtime assessment of sensing models in multi-device environments. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 271–284, 2019.

[229] Behnam Gholami, Pritish Sahu, Ognjen Rudovic, Konstantinos Bousmalis, and Vladimir Pavlovic. Unsupervised multi-target domain adaptation: An information theoretic approach. *IEEE Transactions on Image Processing*, 29:3993–4002, 2020.

[230] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[231] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650, 2020.

Appendix A

Appendix

In this appendix, we provide the experimental details and theoretical justifications for the algorithms proposed in Chapter 7.

A.1 Experiment Details

Loss Formulations of uDA algorithms. We now discuss the adversarial training formulation of the various uDA algorithms with which we evaluated the efficacy of our proposed FRUDA framework in Chapter 7. Recall that we evaluate our method with four domain adaptation techniques: ADDA [123], Gradient Reversal [122] and Wasserstein DA [125], and CADA [212]. Below are the adversarial training formulations of these techniques as proposed in their original papers.

ADDA. Following the same notations used earlier in the paper, the adversarial loss formulations of ADDA can be represented mathematically as:

$$\min_{DI} \mathcal{L}_{adv_{DI}} = -\mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_S}[\log(DI(F_S(\mathbf{x_s}))] - \mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_T}[\log(1-DI(F_T(\mathbf{x_t}))] \tag{A.1}$$

$$\min_{F_T} \mathcal{L}_{adv_M} = -\mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_T}[\log(DI(F_T(\mathbf{x_t}))] \tag{A.2}$$

The Discriminator DI is optimized using $L_{adv_{DI}}$ where the domain data from source and target domains are assigned different domain labels (0 and 1). To update the feature extractor F_T, ADDA proposes to invert the domain labels, which results in the loss formulation given in L_{adv_M}.

In order to run ADDA in a distributed manner, we decompose the discriminator DI into DI_S and DI_T which results in the following loss functions:

$$\mathcal{L}_{adv_{DI_S}} = -\mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_S}[\log(DI_S(F_S(\mathbf{x_s}))] \tag{A.3}$$

$$\mathcal{L}_{adv_{DI_T}} = -\mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_T}[\log(1-DI_T(F_T(\mathbf{x_t}))] \tag{A.4}$$

$$\mathcal{L}_{adv_M} = -\mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_T}[\log(DI_T(F_T(\mathbf{x_t}))] \tag{A.5}$$

Thereafter, we compute local gradients for DI_S and DI_T,

$$\nabla g(DI_S, \mathbf{x_s}) = \frac{\delta L_{adv_{DI_S}}}{\delta DI_S} \tag{A.6}$$

$$\nabla g(DI_T, \mathbf{x_t}) = \frac{\delta L_{adv_{DI_T}}}{\delta DI_T} \tag{A.7}$$

and aggregate them in the sync-up step as shown in Algorithm 1. The aggregated gradients are used to optimize DI_S and DI_T, while F_T is optimized using the loss function in Equation A.5.

Gradient Reversal. The Gradient Reversal approach uses the same loss formulation for the discriminators DI_S and DI_T as ADDA as shown in Equations A.1, A.3, A.4.

However, to update the target extractor F_T, it leverages the gradient reversal strategy, resulting in the following loss function.

$$\begin{aligned}\min_{F_T} \mathcal{L}_{adv_M} &= -\mathcal{L}_{adv_{DI_T}} \\ &= \mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_T}[\log(1 - DI_T(F_T(\mathbf{x_t}))]\end{aligned} \tag{A.8}$$

The computation of local gradients for DI_S and DI_T follow the same process as shown in Equations A.6 and A.7.

Wasserstein DA. In this technique [125], the authors use the Wasserstein distance as the loss function for discriminator. Wasserstein distance between two datasets is defined as

$$\text{Wasserstein}(X_s, X_t) = \frac{1}{n_s}\sum_{\mathbf{x_s}\sim\mathbb{X}_S} DI(F_S(\mathbf{x_s})) - \frac{1}{n_t}\sum_{\mathbf{x_t}\sim\mathbb{X}_T} DI(F_T(\mathbf{x_t})) \tag{A.9}$$

where n_s and n_t are the number of samples in the dataset. The discriminator loss is computed as:

$$\min_{DI} \mathcal{L}_{adv_{DI}} = -\mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_S,\mathbf{x_t}\sim\mathbb{X}_T}[\text{Wasserstein}(\mathbf{x_s}, \mathbf{x_t})] + \gamma L_{grad} \tag{A.10}$$

where L_{grad} is the gradient penalty used to enforce the Lipschitz constraint on the discriminator. Further, the target extractor is optimized using the following loss function:

$$\min_{F_T} \mathcal{L}_{adv_M} = \mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_S,\mathbf{x_t}\sim\mathbb{X}_T}[\text{Wasserstein}(\mathbf{x_s}, \mathbf{x_t})] \tag{A.11}$$

We use the same strategy to compute local gradients for DI_S and DI_T as shown in Eq A.6 and A.7.

CADA. Consensus Adversarial Domain Adaptation is a technique recently proposed by [212] which enforces the source and target extractors to arrive at a consensus in

the feature space through adversarial training. It uses the same loss formulation for the discriminators DI_S and DI_T as ADDA as shown in Equations A.1, A.3, A.4. However, the key difference is that CADA optimizes both the source and target feature extractors in the training process, until the discriminator can no longer distinguish the features from source and target domains.

$$\min_{F_S} \mathcal{L}_{adv_{M1}} = -\mathbb{E}_{\mathbf{x_s}\sim\mathbb{X}_S}[\log(DI_S(F_S(\mathbf{x_s}))] \tag{A.12}$$

$$\min_{F_T} \mathcal{L}_{adv_{M2}} = -\mathbb{E}_{\mathbf{x_t}\sim\mathbb{X}_T}[\log(DI_T(F_T(\mathbf{x_t}))] \tag{A.13}$$

Thereafter, a shared classifier is trained on the sourced labeled data by keeping the source feature extractor fixed. The shared classifier can be used with the target extractor to make predictions.

$$\min_{C_{Shared}} \mathcal{L}_{clf} = -\mathbb{E}_{(\mathbf{x_s},y_s)\sim(\mathbb{X}_\mathbb{S},\mathbb{Y}_\mathbb{S})} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]}[\log(C_{Shared}(F_S(\mathbf{x_s}))]$$

Computing Wasserstein distance across distributed datasets in a privacy preserving manner. Our optimal collaborator selection algorithm requires computing an estimate of the Wasserstein (W_1) distance between a candidate domain (D_C) and the target domain (D_T). Let $\mathbb{X}_C$ and $\mathbb{X}_T$ denote the unlabeled datasets from the two domains. As shown by [125], the W_1 distance can be computed as:

$$W_1(\mathbb{X}_C,\mathbb{X}_T) = \frac{1}{n_C}\sum_{x_s\sim\mathbb{X}_C} DI(F_C(x_c)) - \frac{1}{n_T}\sum_{x_t\sim\mathbb{X}_T} DI(F_T(x_t)) \tag{A.14}$$

where n_C and n_T are the number of samples in the dataset, F_C and F_T are the feature encoders of each domain, and DI is an optimal discriminator trained to distinguish the features from the two domains. To train the optimal discriminator, following loss is minimized:

$$\min_{DI} \mathcal{L}_{adv_{DI}} = -\mathbb{E}_{x_c\sim\mathbb{X}_C, x_t\sim\mathbb{X}_T}[W_1(x_c,x_t) + \gamma L_{grad}]$$

where L_{grad} is the gradient penalty used to enforce 1-Lipschitz continuity on the discriminator.

Interestingly, Equation A.14 has a similar structure to the optimization objectives for ADDA and other uDA algorithms discussed above. Hence, we can use the same principle as DILS and exchange discriminator gradients between nodes to compute the Wasserstein Distance in a distributed manner, without requiring any exchange of raw data.

We initialize F_T with F_C and decompose the discriminator DI into two parts (DI_C

and DI_T) which reside on the respective nodes. The raw data from both nodes is fed into their respective encoders and discriminators, and we compute the gradients of each discriminator as follows:

$$\mathcal{L}_{DI}^{C} = \frac{1}{n_C} \sum_{x_s \sim \mathbb{X}_C} DI_C(F_C(x_c))$$

$$\mathcal{L}_{DI}^{T} = \frac{1}{n_T} \sum_{x_t \sim \mathbb{X}_T} DI_T(F_T(x_t))$$

$$\nabla g(DI_C, x_c) = \frac{\delta \mathcal{L}_{DI}^{C}}{\delta DI_C}$$
$$\nabla g(DI_T, x_t) = \frac{\delta \mathcal{L}_{DI}^{T}}{\delta DI_T}$$

Both nodes exchange their discriminator gradients during a synchronization step and compute aggregated gradients:

$$\nabla g(DI_{\text{agg}}, x_c, x_t) = \nabla g(DI_C, x_c) - \nabla g(DI_T, x_t) \tag{A.15}$$

Finally, both discriminators DI_C and DI_T are updated with these aggregated gradients, and gradient penalty is applied to enforce the 1-Lipschitz continuity on the discriminators. This process continues until convergence and results in an optimal discriminator. Once the discriminators are trained to convergence, we can calculate the Wasserstein distance as:

$$W_1(\mathbb{X}_C, \mathbb{X}_T) = \mathcal{L}_{DI}^{C} - \mathcal{L}_{DI}^{T}$$

To the best of our knowledge, this approach of computing Wasserstein distance in a distributed manner has not been explored before. Moreover, this ability to compute Wasserstein distance in a distributed manner makes it an ideal metric for collaborator selection in a privacy-preserving setting.

Modifications done to the MDAN baseline. As we noted in Section 7.4.2, we compare OCS against a Multi-Collaborator baseline wherein all available candidate domains contribute in domain adaptation with the target. To this end, we employ the MDAN [211] method proposed for multi-source domain adaptation. However, one key difference between MDAN and our problem setting is that MDAN assumes that there are multiple source domains and all of them are labeled. With this assumption, MDAN optimizes the following objective during uDA:

$$\text{minimize } \frac{1}{\gamma}\sum_{i\in[k]}\exp(\gamma(\hat{\varepsilon}_{S_i}(h)-\min_{h'\in H\Delta H}\hat{\varepsilon}_{T,S_i}(h')))$$

where $\hat{\varepsilon}_{S_i}$ is the classification error for source domain S_i obtained using supervised learning (assuming the availability of labels), and $\hat{\varepsilon}_{T,S_i}$ is the error of a domain discriminator trained to separate S_i and target T.

However, in our problem setting, there is only one labeled source and all other domains are unlabeled. As such, we do not have a way to compute the classification error $\hat{\varepsilon}_{S_i}$ for all candidate domains. Therefore, we only use the discriminator error proposed in MDAN as a way to weigh the contribution of each collaborator domain in the adaptation process.

A.2 Theoretical Justifications

Optimal Collaborator Selection (OCS) algorithm. Below we provide the proof of the theorem used for optimal collaborator selection.

Theorem 1. *Let D_1 and D_2 be two domains sharing the same labeling function l. Let θ_{CE} denote the Lipschitz constant of the cross-entropy loss function in D_1, and let θ be the Lipschitz constant of a hypothesis learned on D_1. For any two θ-Lipschitz hypotheses h,h', we can derive the following error bound for the cross-entropy (CE) error $\varepsilon_{\text{CE,D}_2}$ in D_2:*

$$\varepsilon_{\text{CE,D}_2}(h,h') \leq \theta_{\text{CE}}\left(\varepsilon_{\text{L}_1,\text{D}_1}(h,h')+2\theta W_1(D_1,D_2)\right) \tag{A.16}$$

where $W_1(D_1,D_2)$ denote the first Wasserstein distance between the domains D_1 and D_2, and $\varepsilon_{\text{L}_1,\text{D}_1}$ denotes the L_1 error in D_1.

Proof. The L_1 error between two hypotheses h,h' on a distribution D is given by:

$$\varepsilon_{\text{L}_1,\text{D}}(h,h') = \mathbb{E}_{x\sim D}\left[|h(x)-h'(x)|\right] \tag{A.17}$$

We define softmax cross-entropy on a given distribution D as

$$\varepsilon_{\text{CE,D}}(h) = \mathbb{E}_{x\sim D}\left[\left|\log S_{l(x)}h(x)\right|\right], \tag{A.18}$$

where S is the softmax function $\mathbb{R}^n \longrightarrow \mathbb{R}^n$, l is the labelling function, and $S_{l(x)}$ denotes the projection of S to the $l(x)$-component.

Then we have,

$$\begin{aligned}\varepsilon_{\mathrm{CE,D}}(h,h') &= \mathbb{E}_{x\sim D}\left[\left|\log S_{l(x)}h(x) - \log S_{l(x)}h'(x)\right|\right] \\ &= \varepsilon_{\mathrm{L_1,D}}\left(\log S_l h, \log S_l h'\right)\end{aligned} \tag{A.19}$$

Further, using the definition of Lipschitz continuity, we have

$$\left|\log S_{l(x)}h(x) - \log S_{l(y)}h(y)\right| \leq \theta_{\mathrm{CE}}|h(x) - h(y)|, \tag{A.20}$$

where θ_{CE} is the Lipschitz constant of the softmax cross-entropy function.

Next, we follow the triangle inequality proof from [125, proof of Lemma 1] to find that

$$\varepsilon_{\mathrm{L_1,D_2}}\left(\log S_l h, \log S_l h'\right) \leq \varepsilon_{\mathrm{L_1,D_1}}\left(\log S_l h, \log S_l h'\right) + 2\theta_{\mathrm{CE}}.\theta W_1(D_1, D_2), \tag{A.21}$$

where θ is a Lipschitz constant for h and h', if the label $l(x)$ were constant. Since $l(x)$ is constant outside of a measure 0 subset where the labels change, and h and h' are Lipschitz, so in particular measurable, Equation A.21 holds everywhere.

Then, by substituting from Eq. A.19 and Eq. A.20 in Eq. A.21, we get Theorem 1:

$$\begin{aligned}\varepsilon_{\mathrm{CE,D_2}}(h,h') &\leq \varepsilon_{\mathrm{CE,D_1}}(h,h') + 2\theta_{\mathrm{CE}}.\theta W_1(D_1,D_2) \\ &\leq \theta_{\mathrm{CE}}(\varepsilon_{\mathrm{L_1,D_1}}\left(h,h'\right) + 2\theta W_1\left(D_1,D_2\right))\end{aligned} \tag{A.22}$$

□

Privacy Analysis of DILS. Recall that a key feature of DILS is to exchange information between the distributed nodes using *gradients of the discriminators*. This clearly affords certain privacy benefits over existing uDA algorithms since we no longer have to transmit raw training data between nodes. However, prior works have shown that model gradients can potentially leak raw training data in collaborative learning [206], therefore it is critical to examine: *can the discriminator gradients also indirectly leak training data of a domain*? We begin by providing some theoretical justification on why our training algorithm prevents the reconstruction of raw data from discriminator gradients. Later, we study the performance of DILS under a state-of-the-art gradient leakage attack proposed by [205].

We consider the case when the domain discriminator is made of a single neuron and later, we will explain how this analysis generalizes to deeper discriminators.

Let $x = (x_1, x_2, ...x_n) \in \mathbb{R}^n$ be a n-dimensional training point sampled from the target domain. Let the target domain feature extractor F_T be a neural network which outputs a j-dimensional feature vector $e = \left(e_1, e_2, ...e_j\right) \in \mathbb{R}^j$. The squared error discriminator loss $\mathcal{L}_D$ is expressed as:

$$\mathcal{L}_D = \left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) - y \right)^2 \tag{A.23}$$

where W_i (i = 1...j), b and g are respectively the weights, bias, and activation function of the target discriminator, and y is the target domain label. Under this formulation, the gradient with respect to discriminator weight W_j is given as:

$$\begin{aligned}
\nabla_{W_j} &= \frac{\delta L_D}{\delta W_j} \\
&= \frac{\delta \left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) - y \right)^2}{\delta W_j} \\
&= 2\left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) - y \right) \frac{\delta \left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) \right)}{\delta W_j} \\
&= 2\left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) - y \right) \left(g'\left(\sum_{i=1}^{j} W_i e_i + b \right) \right) \frac{\delta \left(\sum_{i=1}^{j} W_i e_i + b \right)}{\delta W_j} \\
&= 2\left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) - y \right) \left(g'\left(\sum_{i=1}^{j} W_i e_i + b \right) \right) \cdot e_j
\end{aligned} \tag{A.24}$$

Similarly, the gradient with respect to the bias b is given as:

$$\begin{aligned}
\nabla_b &= \frac{\delta L_D}{\delta b} \\
&= \frac{\delta \left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) - y \right)^2}{\delta b} \\
&= 2\left(g\left(\sum_{i=1}^{j} W_i e_i + b \right) - y \right) \left(g'\left(\sum_{i=1}^{j} W_i e_i + b \right) \right) \cdot 1
\end{aligned} \tag{A.25}$$

Note that $\frac{\nabla_{W_j}}{\nabla_b} = e_j$, i.e., if an adversary steals the gradients of the weights and bias of the discriminator, they can potentially (and, at best) reconstruct the feature representation e of the training input. It is infeasible to reconstruct the input raw data x from the stolen features e, because our training strategy does not exchange the

feature extractor model F_T between the nodes. Also note that by adding regularization during the training process (e.g., using Dropout or L2 regularization), even reconstruction of the entire feature representations e can be prevented.

Although we provided the above justification assuming that the discriminator consists of a single neuron, our conclusion extends to scenarios where the discriminator is a neural network. Here, the gradients of the first hidden layer of the discriminator network can result in feature leakage:

$$\begin{aligned} \nabla_{W^{(1)}_{c,j}} &= \frac{\delta L_D}{\delta W^{(1)}_{c,j}} \\ &= \theta \cdot e_j \end{aligned} \tag{A.26}$$

where, $W^{(1)}_{c,j}$ is the weight connecting the j^{th} encoded feature e_j with the c^{th} node in the first hidden layer of the discriminator. θ is a real number. As described earlier, the gradient in Equation A.26, in conjunction with the gradient of the bias can at best result in reconstructing the feature representation e, however it does not reveal the training data x from the target domain.

Protection against a state-of-the-art privacy attack. The above analysis was conceptual in nature and intended to give an intuition on the privacy properties of DILS. We now take a recently proposed privacy attack as an example and study the behavior of DILS under it.

[205] showed that gradient matching can be a simple but robust technique to reconstruct the raw training data from stolen gradients. Let us say we are given a machine learning model $F(.)$ with weights W. Let ∇W be the gradients with respect to a private input pair $(\mathbf{x}, y)$. During distributed training, ∇W are exchanged between the nodes.

A reconstruction attack happens as follows: an attacker first randomly initializes a dummy input $\mathbf{x}'$ and label input y'. This data is fed into the model $F(.)$ to compute dummy gradients as follows:

$$\nabla W' = \frac{\partial \ell\left(F\left(\mathbf{x}', W\right), \mathbf{y}'\right)}{\partial W}$$

Finally, the attacker minimizes the distance between the dummy gradients and the actual gradients using gradient descent to reconstruct the private data as follows:

$$\mathbf{x}'^{*}, \mathbf{y}'^{*} = \arg\min_{\mathbf{x}', \mathbf{y}'} \left\| \nabla W' - \nabla W \right\|^2 = \arg\min_{\mathbf{x}', \mathbf{y}'} \left\| \frac{\partial \ell \left(F\left(\mathbf{x}', W\right), \mathbf{y}' \right)}{\partial W} - \nabla W \right\|^2 \quad \text{(A.27)}$$

[205] demonstrate the success of this attack on a number of image datasets.

Can this attack succeed on DILS? There are two key assumptions in this attack: (i) the weights W of the end-to-end machine learning model are available to an adversary in order for them to compute the dummy gradients, (ii) the gradients of all the layers (∇W) between the input x and output y are available to the adversary.

DILS never exchanges the weights of the target domain model (i.e., the feature encoder and the discriminator) during the adversarial training process. The target feature encoder is trained locally and only discriminator gradients are exchanged. Without the knowledge of the model weights W, an attacker can not generate the dummy gradients $\nabla W'$ necessary to initiate the attack on the target domain. Looking at the source or collaborator domain, we do exchange its feature encoder with the target domain in the initialization step of uDA, which could be used by the attacker to generate the dummy gradients $\nabla W'$. However, for the attack to succeed, the attacker also needs the real gradients (∇W) of all the layers between the input x and output y in the source domain. This includes gradients of the feature encoder F_S and the domain discriminator DI_S. In DILS however, we only exchange the gradients of the domain discriminator DI_S during training; the gradients of F_S are never exchanged. Without the knowledge of the gradients of F_S, an attacker cannot use Eq. A.27 to reconstruct the training data of the source domain.

In summary, we have proven that our strategy of distributed uDA based on discriminator gradients does not allow an attacker to reconstruct the private data of either the source or the target domain.