Physics-Directed Data Augmentation for Deep Model Transfer to Specific Sensor

WENJIE LUO, Nanyang Technological University, Singapore
ZHENYU YAN*, The Chinese University of Hong Kong, Hong Kong
QUN SONG, Nanyang Technological University, Singapore
RUI TAN, Nanyang Technological University, Singapore

Run-time domain shifts from the training phase caused by sensor characteristic variation incur performance drops of the deep learning-based sensing systems. To address this problem, existing transfer learning techniques require substantial target-domain data and incur high post-deployment overhead. Differently, we propose to exploit the first principle governing the domain shift to reduce the demand for target-domain data. Specifically, our proposed approach called PhyAug uses the first principle fitted with few labeled or unlabeled data pairs collected by the source sensor and the target sensor to transform the existing source-domain training data into the augmented target-domain data for calibrating the deep neural networks. In two audio sensing case studies of keyword spotting and automatic speech recognition, PhyAug recovers the recognition accuracy losses due to microphones' characteristic variations by 37% to 72% with 5-second unlabeled data collected from the target microphones. In a case study of acoustics-based room recognition, PhyAug recovers the recognition accuracy loss caused by smartphone microphone variation by 33% to 80%. In the last case study of fisheye image recognition, PhyAug reduces the image recognition error due to the camera-induced distortions by 72%.

CCS Concepts: • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Computing methodologies** → *Neural networks*; • **Hardware** → *Sensor applications and deployments.*

Additional Key Words and Phrases: Smart sensors, neural networks, data augmentation, domain adaptation

ACM Reference Format:
Wenjie Luo, Zhenyu Yan, Qun Song, and Rui Tan. 2022. Physics-Directed Data Augmentation for Deep Model Transfer to Specific Sensor. *ACM Trans. Sensor Netw.* 1, 1 (July 2022), 30 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Part of this work was completed when Zhenyu Yan was with Singtel Cognitive and Artificial Intelligence Lab for Enterprises, Nanyang Technological University.

A preliminary version of this work appears in The 20th International Conference on Information Processing in Sensor Networks (IPSN'21). This research was conducted at Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU), which is a collaboration between Singapore Telecommunications Limited (Singtel) and Nanyang Technological University (NTU) that is funded by the Singapore Government through the Industry Alignment Fund - Industry Collaboration Projects Grant. Authors' addresses: Wenjie Luo, Singtel Cognitive and AI Lab for Enterprises, Nanyang Technological University, Singapore; Zhenyu Yan, zyyan@cuhk.edu.hk, Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, Hong Kong; Qun Song, ERI@N, Interdisciplinary Graduate School, Nanyang Technological University, Singapore; Rui Tan, Singtel Cognitive and AI Lab for Enterprises, Nanyang Technological University, Singapore.

1550-4859/2022/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recent advances of deep learning have attracted great interest of applying it in various embedded sensing systems. The deep neural networks (DNNs), albeit capable of capturing sophisticated patterns, require significant amounts of labeled training data to realize the capability. A sensing DNN trained on a design dataset is often observed run-time performance degradations, due to *domain shifts* [13]. The shifts are generally caused by the sensor characteristics deviations in the real deployments from those captured by the design dataset.

Transfer learning [18] has received increasing attention for addressing domain shifts. It is a cluster of approaches aiming at storing knowledge learned from one task and applying it to a different but related task. Under the transfer learning scheme, ideally, with little new training data, we can transfer a DNN trained from the *source domain* (i.e., the design dataset) to the *target domain* (i.e., data captured by a specific sensor in real deployment). Prevalent transfer learning techniques, including *freeze-and-train* [22] and *domain adaptation* [18], require substantial training data collected in the target domain. The freeze-and-train approach retrains selected layers of a DNN with new target-domain samples to implement the model transfer. Domain adaptation trains a new DNN to transform the target-domain inference data back to the source domain. For instance, the Mic2Mic [12] trains a cycle-consistent generative adversarial network (CycleGAN) to perform the translation between two microphones with distinct characteristics. The training of CycleGAN requires about 20 minutes of microphone recording from both domains for a keyword spotting task [12]. In summary, although the prevalent transfer learning techniques reduce the demands on the target-domain training data in comparison with learning from scratch in the target domain, they still need substantial target-domain data to implement the model transfer. This paper exploits the first principles governing the sensor characteristics to reduce the demand on target-domain data for model transfer, vis-à-vis the aforementioned *physics-regardless* approaches [12, 18, 22].

Recent studies attempt to incorporate prior knowledge in the form of commonsense [35] or physical laws [26, 27] to increase the learning efficiency. The presentation of the prior knowledge to learning algorithms is the core problem of *physics-constrained machine learning*. In [26, 27], the closed-form physical laws are incorporated into the loss function of DNN training. The improved learning efficiency of the physics-constrained machine learning encourages exploiting first principles to address domain shifts more efficiently. However, physics-constrained machine learning requires new DNN architectures and/or training algorithms; it is not to exploit first principles in transferring existing DNNs to address the domain shift problems.

In the domain of physics-rich cyber-physical systems, the domain shifts caused by the property of the sensors are often governed by first principles. For example, the performance of a microphone is often characterized by the frequency response curve [13]; a fisheye camera is characterized by the polynomial function [29], etc. In this paper, we propose a new approach called *physics-directed data augmentation* (PhyAug) to use a minimum amount of data collected from the target sensor to estimate the parameters of the first principle governing the domain shift caused by a specific sensor, then use the parametric first principle to generate augmented target-domain training data. The augmented target-domain data samples are used to transfer or retrain the source-domain DNN. PhyAug has the following two key features. **First**, different from the conventional data augmentations that apply unguided *ad hoc* perturbations (e.g., noise injection) and transformations (e.g., scaling, rotation, etc) on existing training data to improve the DNNs' robustness against variations, PhyAug augments the training data strategically by following first principles to transfer DNNs. **Second**, PhyAug uses augmented data to represent the domain shifts and thus requires no modifications to the legacy DNN architectures and training algorithms. In contrast, recently proposed domain adaptation approaches based on adversarial learning [1, 12, 15, 30] update the DNNs

under new adversarial training architectures that need extensive hyperparameter optimization and even application-specific redesigns. Such needs largely weaken their readiness, especially when the original DNNs are sophisticated such as the DeepSpeech2 [16] for automatic speech recognition.

This paper applies PhyAug to four case studies and quantifies the performance gains compared with other possible approaches. The first two case studies aim to adopt DNNs for keyword spotting (KWS) and automatic speech recognition (ASR) respectively to individual microphones. KWS and ASR differ in DNN model depth and complexity. The domain shifts are from the microphone's hardware characteristics. We apply PhyAug to profile the source and the target microphones by playing a 5-second white noise, then augment the source-domain data to re-train the DNN for the target domain. PhyAug recovers the microphone-induced accuracy loss by 53%-72% and 37%-70% in KWS and ASR, respectively. The third case study is the acoustics-based room recognition (ARR). Our experiment shows that the room recognition accuracy drop can be up to 80% if the pre-trained model is evaluated using the data collected from a specific smartphone microphone. We apply PhyAug to profile the source and the target smartphones by recording 1-minute acoustic background spectrograms in any room simultaneously, then augment the source smartphone's data to train the DNN for the target smartphone. PhyAug recovers the accuracy loss by 33%-80% for the target smartphone. The fourth case study focuses on fisheye image recognition (FIR). We apply PhyAug to adapt a ResNet-50 DNN designed for pinhole cameras to a specific fisheye camera using the estimated parameters. The parameters estimation for a fisheye camera only requires around 20 image samples taken on a checkerboard picture. PhyAug recovers the camera-induced object recognition accuracy loss by 72% and avoids the compute-intensive image rectification.

The main contribution of this paper is as follows. We propose to leverage the first principle governing the sensing process and fitted with a small amount of source- and target-domain data to extensively augment the target-domain data for model transfer. This approach is more efficient than the physics-regardless transfer learning in terms of target-domain data sampling complexity. The first principle's parameters that capture the sensor characteristics can be obtained from either sensor specification or a one-time calibration process. With the derived the parametric first principle, we can avoid collecting substantial training data from each specific sensor forming an individual target domain. The data and source code for the case studies are made publicly available.[1]

The remainder of this paper is organized as follows. §2 overviews the PhyAug approach and reviews related work. §3, §4, §5, and §6 present the four case studies. §7 discusses several issues. §8 concludes this paper.

2 APPROACH OVERVIEW & RELATED WORK

The design of the DNN used for a sensing task is often driven by a training dataset that is either publicly available or collected by the system designer. However, when the DNN is deployed, the actual distribution of the inference data samples may be different from that of the training dataset. For instance, the sensors used for collecting the training and inference data samples can have different characteristics caused by hardware model differences and manufacturing imperfections across sensors of the same model. The basic principle of PhyAug is to obtain the sensor characteristics using low-overhead approaches and then learn the mapping from the source-domain sensor to the target-domain sensor. After the mapping is learned, we may convert the training data collected by the source-domain sensor to the augmented data that are consistent with the target-domain sensor. As a result, the deep model retrained using the augmented data can work effectively on the data collected by the target-domain sensor.

[1] https://github.com/jiegev5/PhyAug

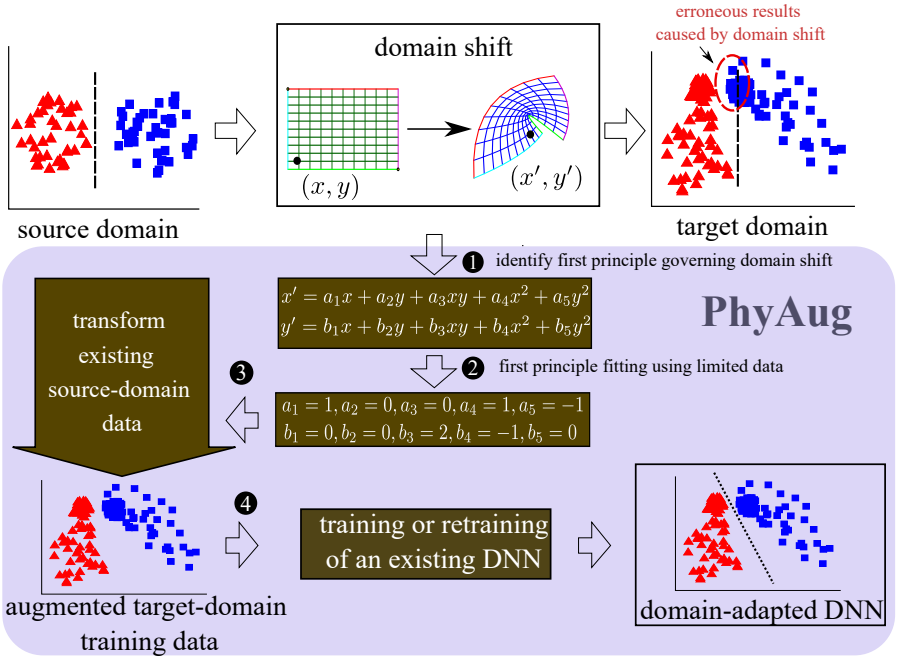


Fig. 1. PhyAug workflow.

2.1 PhyAug Approach Overview

Fig. 1 illustrates PhyAug's workflow using a simple example, where the DNN performs two-class classification based on two-dimensional data samples and the first principle governing the domain shift is a nonlinear polynomial transform. Such transform can be used to characterize camera lens distortion [21]. To simplify the discussion, this example considers class-independent domain shift, i.e., the transform is identical across all the classes. Note that PhyAug can deal with class-dependent domain shifts, which will be discussed later. The general transfer learning approaches regardless of the first principles need to draw substantial data samples from both the source and target domains. Then, they apply *domain shift learning* techniques to update the existing source-domain DNN or construct a prefix DNN [12] to address the domain shift. Extensive data collection in the target domain often incurs undesirable overhead in practice.

Differently, as shown in the Fig. 1, PhyAug applies the following four steps to avoid extensive data collection in the target domain. ❶ The system designer identifies the parametric first principle governing the domain shift. For the current example, the parametric first principle is $x' = a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2$ and $y' = b_1x + b_2y + b_3xy + b_4x^2 + b_5y^2$, where (x, y) and (x', y') are a pair of data samples in the source and target domains, respectively, and a_i, b_i are unknown parameters. ❷ A small amount of unlabeled data pairs are collected from the source and target sensors to estimate the parameters of the first principle. For this example, if the domain shift is perturbation-free, the minimum number of data pairs needed is the number of unknown parameters of the polynomial transform. If the domain shift is also affected by other unmodeled perturbations, more data pairs can be drawn to improve the accuracy of estimating the parameters under a least squares formulation. If the domain shift is class-dependent, the data pair sampling and parameter estimation should be performed for each class separately. ❸ All the existing source-domain training data samples are transformed to the target domain using the fitted first principle, forming an augmented training

Table 1. Categorization, used techniques, and requirements of various solutions to address domain shifts.

Category	Used technique	Solution	Applications in publication	Requirements				
				Source domain label	Target domain label	Paired label data	First principle	Target domain data volume*
Domain adaptation (Model transfer)	Adversarial learning	FADA [15]	computer vision	✔	✔	✔	–	low
		ADDA [30]	computer vision	–	–	–	–	high
		TransAct [1]	activity sensing	–	–	–	–	medium
		Mic2Mic [12]	voice sensing	–	–	–	–	high
	Meta learning	MetaSense[5]	voice & motion	✔	✔	–	–	low
	Contrastive learning	CDCL [32]	computer vision	✔	–	–	–	medium
	Data augmentation	**PhyAug**	voice sensing	–	–	–	✔	low
			room recognition	–	–	–	✔	low
			computer vision	–	–	–	✔	low
Model robustness	Data augmentation	CDA [13]	voice and activity sensing	–	–	–	✔	medium

* The bars represent oracle scales partially based on the reported numbers in respective publications. Fully comparable scales are difficult to obtain because the solutions are designed for different applications. PhyAug is compared with FADA, Mic2Mic, and CDA in the evaluation sections of this paper. Reasons for excluding other approaches from the comparison will be discussed in the case studies.

dataset in the target domain. ❹ With the augmented training dataset, various techniques can be employed to transfer the existing DNN built in the source domain to the target domain. For instance, we can retrain the DNN with augmented data. The retraining can use the existing DNN as the starting point to speed up the process. For instance, for the DeepSpeech2 [16] which is a large-scale ASR model used in §4, the retraining only requires half of the training time compared with the training from scratch using the augmented data.

For sensing DNN design, the source domain is in general the design dataset. In such case, the source domain cannot be excited any more for data pair sampling in both domains simultaneously. However, we can recreate the excitation to collect the corresponding target-domain samples. For instance, we can use a speaker to play voice samples from the source-domain dataset and collect the corresponding samples from a target-domain microphone. Similarly, we can display image samples from the source-domain dataset and collect the corresponding samples from a target-domain camera that may have optical distortions.

2.2 Related Work

The applications of deep learning in embedded sensing systems have obtained superior inference accuracy compared with heuristics and conventional machine learning. Various approaches have been proposed to address the domain shift problems in embedded sensing [1, 5, 12, 13] and image recognition [15, 30, 32]. Table 1 summarizes the categorization, used techniques, and requirements of these approaches. In what follows, we discuss the important details of these approaches and their differences from PhyAug.

■ **Domain adaptation:** Few-shot Adversarial Domain Adaptation (FADA) [15] transfers the model with limited amount of target-domain training data. It uses the *supervised adversarial learning* technique to find a shared subspace of the data distributions in the source and target domains. FADA requires labeled and paired data samples from both the source and target domains. Adversarial Discriminative Domain Adaptation (ADDA) [30] uses *unsupervised adversarial learning* to learn a feature encoder for the target domain. Although ADDA requires neither class labels nor data pairing, it demands substantial unlabeled target-domain data. TransAct in [1] considers sensor heterogeneity in human activity recognition and uses unsupervised adversarial learning to learn stochastic features for both domains. It requires hundreds of unlabeled target-domain data samples.

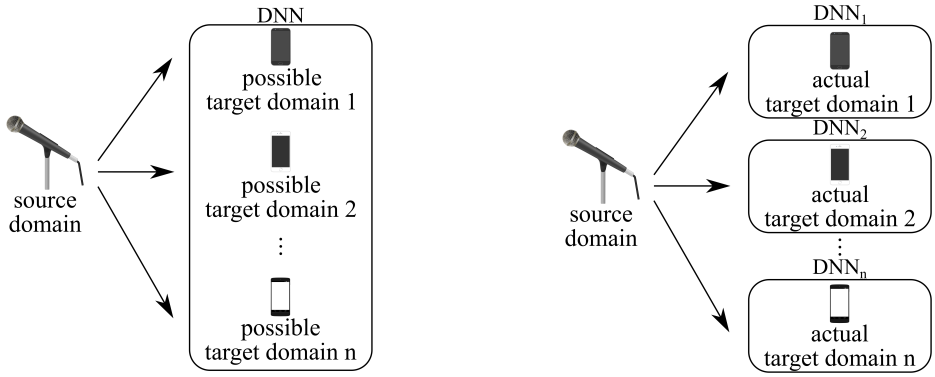
Mic2Mic [12] applies CycleGAN, which is also an adversarial learning technique, to map the target-domain audio recorded by a microphone "in the wild" back to the source-domain microphone for which the DNN is trained. Mic2Mic requires about 20 minutes of speech recording from both microphones, which represents an overhead. It performs one-to-one translations. Our experiment results in §3 and §4 show that CycleGAN may underperform when the source domains are publicly available speech datasets collected using numerous microphones in diverse environments. Cross-domain contrastive learning (CDCL) exploits contrastive self-supervised learning for domain adaptation. Specifically, it minimizes the distance of the cross-domain data samples from the same classes via the contrastive loss, such that the trained model is invariant to the domain difference. We show in the §3 this approach requires a significant amount of target domain data to achieve a satisfactory result.

PhyAug is a domain adaptation approach. Compared with ADDA [30], TransAct [1], and Mic2Mic [12] that are based on unsupervised adversarial learning and thus require substantial target-domain training data, PhyAug exploits the first principle governing the domain shift to reduce the demand on target-domain data. Although FADA [15] aims at reducing the demand for target-domain data, it requires extensive other information such as class labels in both domains. In contrast, PhyAug requires unlabeled data only. Different from Mic2Mic [12] that requires the source domain to be a single microphone, PhyAug admits a source-domain dataset collected via many (and even unknown) microphones in the KWS and ASR case studies. This enlarges the application scope because the datasets used to drive the design of DNNs for real-world applications often consist of recordings from diverse sources.

MetaSense [5] employs meta-learning [10] to rapidly adapt model to the target user' condition with few shots. It uses data collected from multiple source domains to train a base model that can adapt to a target domain related to the source domains. However, it requires substantial training data from both domains and class labels from each source domain. For voice sensing, MetaSense cannot use a source-domain dataset collected via many unlabeled microphones. But PhyAug can.

As an extension to the previous work [11], we apply our proposed PhyAug to two new case studies. First, the new fisheye recognition (FIR) case study extends the application scope of PhyAug from audio sensing to more complex image recognition. FIR and keyword spotting (KWS) share some similarities in adopting the convolutional neural networks (CNNs) for classification. However, FIR is more challenging compared with KWS in terms of task complexity. This extension shows that PhyAug can be applied to different multimedia sensing tasks with varying sensing complexities. Second, the new acoustic room recognition (ARR) case study extends the application scope of PhyAug from multimedia data to the Internet of Things (IoT) sensing data. IoT sensing such as ARR can be more challenging compared to multimedia sensing. The reasons are two-fold. First, IoT sensing data are generally uninterpretable by humans, whereas multimedia data are human-interpretable and often involve humans for labeling. Second, the subtle differences among the classes of IoT sensing data also render more challenges for deep learning models to learn effective representations. Apart from the case studies, we also strengthen the experiments and provide more in-depth analysis.

■ **Model robustness via data augmentation:** Data augmentation has been widely adopted for enhancing model robustness. As illustrated in Fig. 2a, a conventional scheme presumes a number of domain shifts (e.g., scaling, rotation, noise injection, etc) and follows them to generate augmented training samples. Then, the original and the augmented data samples are used to train a single DNN. During the serving phase, this DNN remains robust to the domain shift resembling the presumption. However, should the actual domain shift be out of the presumption, the robustness is lost. The study [13] adopts the above conventional data augmentation (CDA) approach to mitigate the impact of sensor heterogeneity on DNN's accuracy. Specifically, it estimates the probability distribution of



(a) Data augmentation for model robustness (e.g.,[13]).

(b) Data augmentation for model transfer (PhyAug).

Fig. 2. Different purposes of data augmentation illustrated using voice sensing. Note that the source domain may contain many microphones used to collect training samples.

sensors' heterogeneity characteristics from a *heterogeneity dataset* and then uses the characteristics sampled from the estimated distribution to generate augmented training data. As the dataset needs to cover heterogeneity characteristics, its collection in practice incurs a considerable overhead. The heterogeneity dataset in [13] consists of 2-hour recordings of 20 different microphones placed equidistant from an audio speaker. If the characteristic of a microphone "in the wild" is out of the estimated characteristic distribution (i.e., a missed catch), the enhanced DNN may not perform well. Since CDA uses sensor characteristics, we view it as an approach directed by first principles. Different from CDA's objective of enhancing model robustness, PhyAug uses data augmentation to transfer a model to a specific target domain (i.e., sensor). Fig. 2b illustrates this in the context of voice sensing, where microphones' unique characteristics create domains. PhyAug constructs a dedicated DNN for each target domain. Thus, PhyAug is free of the missed catch problem.

2.3 Methodology

As this paper proposes PhyAug which is a domain adaptation approach, it is desirable to show PhyAug's applicability to multiple applications and its scalability to address different levels of pattern sophistication. Therefore, we apply PhyAug to four applications, i.e., KWS, ASR, ARR and FIR. Although KWS and ASR are two specific human voice sensing tasks, they have significantly different complexities. ARR is a mobile sensing application. The benchmark results presented in this paper show that ARR performance is greatly affected by smartphone heterogeneity. FIR is a visual sensing application where the fisheye camera produces non-linear distortion on the captured images, causing domain shift from the pinhole camera. For each case study, we compare PhyAug with multiple existing approaches to show the advantages and performance gains of PhyAug.

3 CASE STUDY 1: KEYWORD SPOTTING (KWS)

Human voice sensing is important for human-computer interactions in many Internet of Things (IoT) applications. At present, the DNN for a specific human voice sensing task is often trained based on a *standard dataset*. However, as IoT microphones are often of small form factors and low cost, their recordings often suffer degraded and varied voice qualities. In addition, the environment that an IoT microphone resides in can also affect its recording. For instance, the echo patterns in

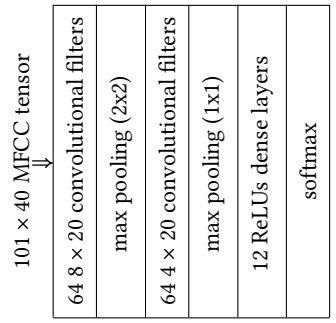


Fig. 3. CNN structure used in KWS case study.

Fig. 4. Microphones & experiment setup.

indoor spaces of different sizes can be distinct. Such run-time variations may be poorly captured by the standard dataset. As a result, the DNN yields reduced accuracy after the deployment.

In this paper, we consider two human voice sensing functions: KWS and ASR. We apply PhyAug to address the domain shift problem. Specifically, we start from a swift process of profiling the IoT microphone's frequency response curve (FRC) with the help of a smartphone. Then, we use the FRC to transform the standard dataset. Finally, we retrain the DNN using the transformed dataset to obtain a personalized DNN for the IoT microphone.

In the case studies of KWS (§3) and ASR (§4), **source domain** is the standard dataset originally used to train the DNN; **target domain** is the dataset of voice samples captured by a specific deployed microphone; **first principle** is the microphone's FRC induced by the microphone hardware and its ambient environment.

3.1 Problem Description

We conduct a set of preliminary experiments to investigate the impact of diverse microphones on the KWS accuracy. Based on the results, we state the problem.

3.1.1 Standard dataset and DNN. We use Google Speech Commands Dataset [33] as the standard dataset in this case study. It contains 65,000 one-second utterances of 30 keywords collected from thousands of people. Audio files are sampled at 16 kilo samples per second (ksps). We pre-process the voice samples as follows. First, we apply a low-pass filter (LPF) with a cutoff frequency of

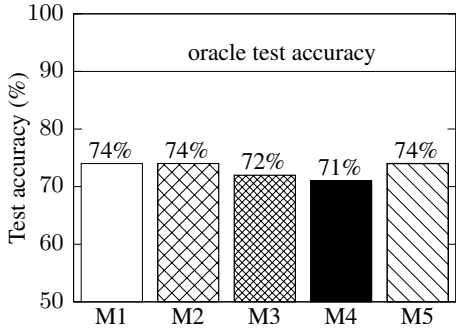


Fig. 5. KWS accuracy on microphones. Horizontal line is accuracy on standard dataset.

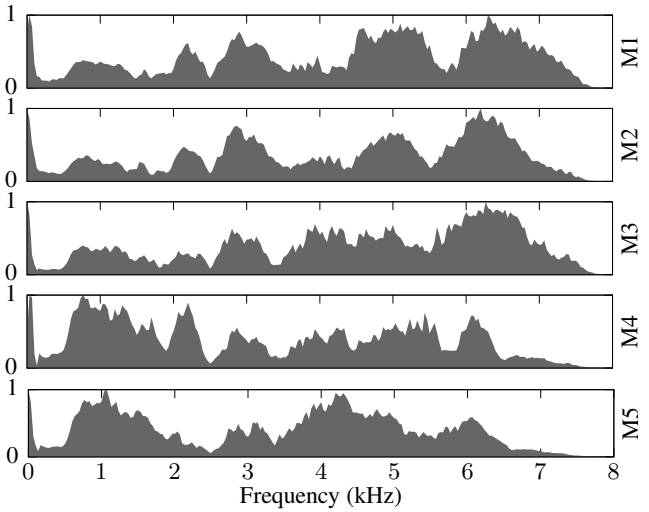


Fig. 6. The five microphones' FRCs. The y-axis of each sub-figure is normalized amplitude.

4 kHz on each voice sample, because human voice's frequency band ranges from approximately 0.3 kHz to 3.4 kHz. Then, for each filtered voice sample, we generate 40-dimensional Mel-Frequency Cepstral Coefficients (MFCC) frames using 30-millisecond window size and 10-millisecond window shift. The z-score normalization is applied on each MFCC frame. Eventually, each voice sample is converted to a 101×40 MFCC tensor. The dataset is randomly split into training, validation, and testing sets following an 8:1:1 ratio.

We implement a CNN to recognize 10 keywords, i.e., "yes", "no", "left", "right", "up", "down", "stop", "go", "on", and "off". We also add two more classes to represent *silence* and *unknown keyword*. Fig. 3 shows the structure of the CNN. It achieves 90% test accuracy, which is similar to that in [36] and referred to as the *oracle test accuracy*.

3.1.2 Impact of microphone on KWS performance. In this section, we demonstrate that the CNN has performance degradation as a result of microphone heterogeneity. We test the CNN on samples captured by five different microphones named M1, M2, M3, M4, and M5 as shown in Fig. 4 that have list prices from high ($80) to low ($3.5). M1 and M2 are two high-end desktop cardioid condenser microphones, supporting sampling rates of 192 ksps at 24-bit depth and 48 ksps at 16-bit depth,

effective frequency responses of $[30\,\text{Hz}, 16\,\text{kHz}]$ and $[30\,\text{Hz}, 15\,\text{kHz}]$, respectively. M3 is a portable clip-on microphone with an effective frequency response range of $[20\,\text{Hz}, 16\,\text{kHz}]$. M4 and M5 are two low-cost mini microphones without detailed specifications. Fig. 4 shows the placement of the microphones. For fair comparison and result reproducibility, we use an Apple iPhone 7 to play the original samples of the test dataset through its loudspeaker, with all microphones placed at equal distances away.

The samples recorded by each microphone are fed into the KWS CNN for inference. Fig. 5 shows the test accuracy for each microphone. Compared with the oracle test accuracy of 90%, there are 14% to 19% absolute accuracy drops due to domain shifts. By inspecting the spectrograms of the original test sample and the corresponding ones captured by the microphones, we can observe the differences. This explains the distinct accuracy drops among microphones. From the above experiment results, the research questions addressed in this case study are as follows. First, how to profile the characteristics of individual microphones with low overhead? Second, how to exploit the profile of a particular microphone to recover KWS's accuracy?

3.2 PhyAug for Keyword Spotting

PhyAug for KWS consists of two procedures: *fast microphone profiling* and *model transfer via data augmentation.*

3.2.1 Fast microphone profiling. A microphone can be characterized by its frequency response consisting of magnitude and phase. We only consider the magnitude component, because the information of a voice signal is largely represented by the energy distribution over frequencies, with little/no impact from the phase of the voice signal in the time domain. Let $X(f)$ and $Y(f)$ denote the frequency-domain representations of the considered microphone's input and output. The FRC to characterize the microphone is $H(f) = \frac{|Y(f)|}{|X(f)|}$, where $|\cdot|$ represents the magnitude.

We propose a fast microphone profiling approach that estimates $H(f)$ in a short time. It can be performed through a factory calibration process or by the user after the microphone is deployed. Specifically, a loudspeaker placed close to the target microphone emits a band-limited acoustic white noise $n(t)$ for a certain time duration. The frequency band of the white noise generator is set to be the band that we desire to profile. Meanwhile, the target microphone records the received acoustic signal $y_n(t)$. Thus, the FRC is estimated as $H(f) = \frac{|\mathcal{F}[y_n(t)]|}{|\mathcal{F}[n(t)]|}$, where $\mathcal{F}[\cdot]$ represents the Fourier transform. As $n(t)$ has a nearly constant power spectral density (PSD), this approach profiles the microphone's response at all frequencies in the given band.

In our experiments, we use the iPhone 7 shown in Fig. 4 to emit the white noise. We set the frequency band of the noise generator to be $[0, 8\,\text{kHz}]$, which is the Nyquist frequency of the microphone. Fig. 6 shows the measured FRCs of the five microphones used in our experiments. Each FRC is normalized to $[0, 1]$. We can see that the microphones exhibit distinct FRCs. In addition, we observe that the two low-end microphones M4 and M5 have lower sensitivities to the higher frequency band, i.e., 5 kHz to 8 kHz, compared with the microphones M1, M2, and M3.

3.2.2 Model transfer via data augmentation. We augment training samples in the target microphone's domain by transforming the original training samples using FRC. The procedure for transforming a sample $x(t)$ is as follows: (1) Apply the pre-processing LPF on $x(t)$ to produce $x'(t)$; (2) Conduct short-time Fourier transform using 30-millisecond sliding windows with an offset of 10 milliseconds on $x'(t)$ to produce 101 Fourier frames, i.e., $X_i(f)$, $i = 1, 2, \ldots 100$; (3) Multiply the magnitude of each Fourier frame with the FRC to produce $|Y_i(f)| = H(f) \cdot |X_i(f)|$; (4) Generate the MFCC frame from each PSD $|Y_i(f)|^2$; (5) Concatenate all 101 MFCC frames to form the MFCC

tensor. Lastly, PhyAug retrains the CNN with augmented data samples for the microphone using the pre-trained CNN as the starting point.

3.3 Performance Evaluation

3.3.1 Alternative approaches. Our performance evaluation employs the following alternative approaches.

■ **Data calibration:** At run time, it uses the measured FRC to convert the target-domain data back to the source-domain data and then applies the pre-trained CNN on the converted data. Specifically, let $Y_i(f)$ denote the ith Fourier frame after the microphone applies the LPF and short-time Fourier transform on the captured raw data. Then, it estimates the corresponding source-domain PSD as $|X_i(f)|^2 = \left(\frac{|Y_i(f)|}{H(f)}\right)^2$ and generates the MFCC frame from $|X_i(f)|^2$. The MFCC tensor concatenated from the MFCC frames over time is fed to the pre-trained CNN.

■ **Conventional data augmentation (CDA) [13]:** This alternative captures the essence of the approach in [13] following the conventional data augmentation scheme illustrated in Fig. 2a. Specifically, one out of the five microphones, e.g., M1, is designated as the testing microphone. The remaining four, e.g., M2 to M5, are used to generate a *heterogeneity dataset* [13]. The *heterogeneity generator* [13] is constructed as follows. For each microphone in the heterogeneity dataset, FRC is measured multiple times with the fast profiling process. At any frequency f, the FRC value is modeled by a Gaussian distribution. A Gaussian mixture is formed by the four heterogeneity-dataset microphones' Gaussian distributions with equal weights. The Gaussian mixtures for all frequencies form the heterogeneity generator. Then, each source-domain training sample is transformed by an FRC sampled from the heterogeneity generator into an augmented sample. Lastly, the DNN is retrained with the augmented training samples and tested with the samples captured by the testing microphone.

■ **CycleGAN (essence of [12]):** Mic2Mic [12] trains a CycleGAN using unlabeled and unpaired data samples collected from two microphones A and B. Then, CycleGAN can translate a sample captured by A to the domain of B, or vice versa. Following [12], we train a CycleGAN to translate the samples captured by a target microphone to the source domain of Google Speech Commands Dataset. To measure the test accuracy, a test sample collected by a microphone is converted by the corresponding CycleGAN to the source domain and fed into the pre-trained CNN.

Compared with PhyAug that requires a single 5-second profiling data collection process for each microphone, CDA repeats the profiling process many times for each heterogeneity microphone to construct the heterogeneity generator; the training of CycleGAN requires 15 minutes of data collected from each target microphone. Thus, both alternative approaches have higher overheads.

■ **FADA [15]**: It trains a feature encoder and classifier in the source domain. Then, it combines source-domain and target-domain data to train a domain-class discriminator. Finally, the weights of the feature encoder and classifier are updated to the target domain through adversarial learning using the domain-class discriminator. To apply FADA for KWS, we follow the architecture in [15] and modify the KWS model in Fig. 3 by adding a fully-connected layer before the last dense layer. Thus, the model has a feature encoder (CNN layers) and a classifier (fully-connected layers).

■ **CDCL [32]**: CDCL comprises three steps for domain adaptation. First, we train a domain-invariant feature encoder to minimize the distance between the source-domain data and the target-domain data via the contrastive learning. We follow the procedure in [32] and construct the positive and negative data samples as follows. The data samples from the different domains but in the same class are viewed as the positive samples. The data samples that are in the different classes from the same or different domains are viewed as negative samples. Second, we freeze the trained feature encoder and apply it on the labeled source-domain data to train a classifier. Third,

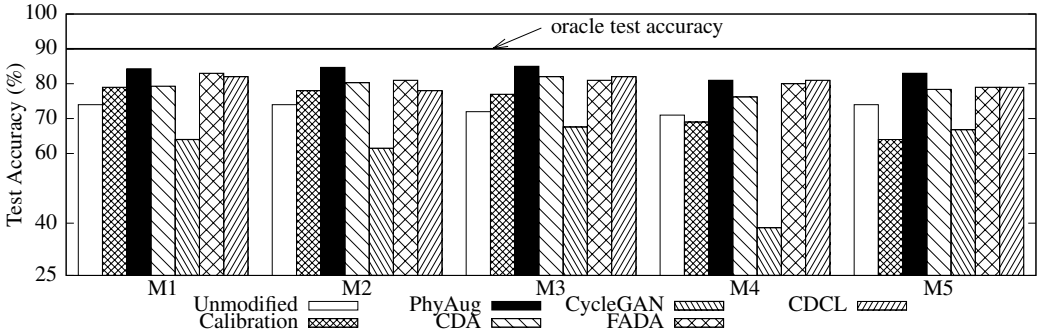


Fig. 7. KWS test accuracy using various approaches on tested microphones. Compared with the unmodified baseline, PhyAug recovers the accuracy losses by 64%, 67%, 72%, 53%, and 56% respectively for the five microphones toward the oracle test accuracy.

we apply the trained feature encoder and the classifier on the target-domain data to evaluate the domain adaptation performance. In this paper, the used feature encoder is a ResNet-18 model and the classifier is a multi-layer perceptron (MLP) consisting of 4 layers. The numbers of neurons in four layers of the MLP are 512, 1024, 1024 and 12.

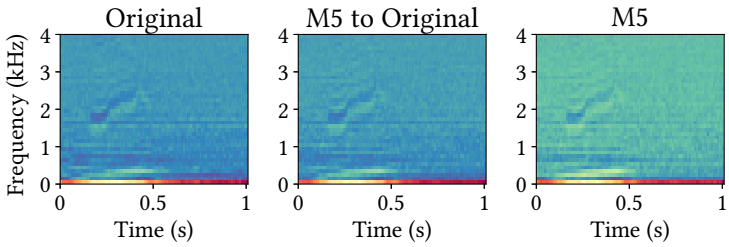
We exclude the MetaSense, ADDA and TransAct reviewed in §2.2 from the baselines for the following reasons. MetaSense cannot be applied to a source-domain dataset collected via many unlabeled microphones. We obtain unsatisfactory results for ADDA in the adversarial training with hours' target-domain training data and extensive hyperparameter tuning. We suspect that the amount of target-domain training data is still insufficient for ADDA. Note that PhyAug only requires five seconds' unlabeled target-domain data as shown shortly. TransAct is customized for activity recognition that differs from human voice sensing.

3.3.2 Evaluation results. We apply PhyAug and the alternatives for the five microphones in Fig. 4. The test accuracies are shown in Fig. 7. The bars labeled "unmodified" are the results from Fig. 5, for which no domain adaptation technique is applied. We include them as the baseline. The results are explained in detail as follows.

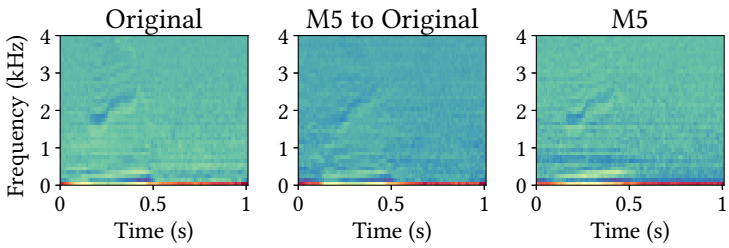
■ **Data calibration:** It brings test accuracy improvements for M1, M2, and M3. The average test accuracy gain is about 4%. For the cheap microphones M4 and M5, it results in test accuracy deteriorations. The reason is as follows. Its back mapping uses the reciprocal of the measured FRC (i.e., $1/H(f)$), which contains large elements due to the near-zero elements of $H(f)$. The larger noises produced by the low-end microphones M4 and M5 are further amplified by the large elements of $1/H(f)$, resulting in performance deteriorations. Thus, although this approach may bring performance improvements, it is susceptible to noises.

■ **PhyAug:** The black bars in Fig. 7 show PhyAug's results. Compared with the unmodified baseline, PhyAug recovers the test accuracy losses by 64%, 67%, 72%, 53%, and 56% for the five microphones. PhyAug cannot fully recover the test accuracy losses. This is because PhyAug only addresses the deterministic distortions due to microphones; it does not address the other stochastic factors such as the environmental noises and the microphones' thermal noises.

■ **CDA:** It recovers certain test accuracy losses for all microphones. This is because for any target microphone, there is at least one heterogeneity dataset microphone giving a similar FRC as the target microphone. Specifically, from Fig. 6, M1, M2, and M3 exhibit similar FRCs; M4 and M5 exhibit similar FRCs (i.e., they have good responses in lower frequencies). However, PhyAug consistently



(a) M5 ⟶ M1

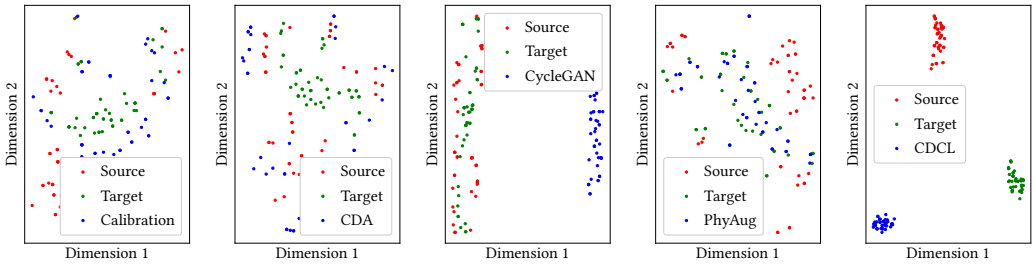


(b) M5 ⟶ Original

Fig. 8. CycleGAN translation results (mid column). (a) Translation from M5 to M1. High similarity between first and second columns shows effectiveness of CycleGAN. (b) Translation from M5 to the domain of Google Speech Commands Dataset. Dissimilarity between first and second columns shows ineffectiveness of CycleGAN.

outperforms CDA. In addition, CDA introduces larger overhead than PhyAug as discussed in §3.3.1.

■ **CycleGAN:** It leads to test accuracy deteriorations for all five target microphones. Although CycleGAN is effective in translating the domain of a microphone to that of another microphone, which is the basis of Mic2Mic [12]. Howerver, CycleGAN is ineffective in translating a certain microphone to the source domain of a dataset that consists of recordings captured by many microphones. We illustrate this using an example of CycleGAN translated audio spectrogram. First, we train a CycleGAN to translate M5 to M1. The first and the third columns of Fig. 8a show the spectrograms captured by M1 and M5 for the same sample played by the smartphone in the setup shown in the paper. We can see that there are discernible differences. The mid column shows the output of the CycleGAN, which is very similar to the first column. This result suggests that CycleGAN is effective for device-to-device domain translation. Then, we apply the same approach to train a different CycleGAN to translate M5 to the domain of Google Speech Commands Dataset. Fig. 8b shows the results. The third column is the spectrogram captured by M5 when a dataset sample shown in the first column is played by the smartphone in the setup shown in the paper. The mid column is the CycleGAN's translation result, which has discernible differences from the first column, suggesting the ineffectiveness of CycleGAN. An intuitive explanation is that the CycleGAN shown with samples captured by many microphones during the training phase is confused and caters into no single microphone. Due to the discrepancy between CycleGAN's output and the dataset, the pre-trained CNN fed with CycleGAN's outputs yields low test accuracy.



(a) r = 7 (b) r = 0.2 (c) r = 1 (d) r = 10 (e) r = 0.4

Fig. 9. t-SNE visualization of different domain data. The r value reported in the sub-figure caption characterizes the effectiveness of the approach. It is the ratio of the source-translation and target-translation distances.

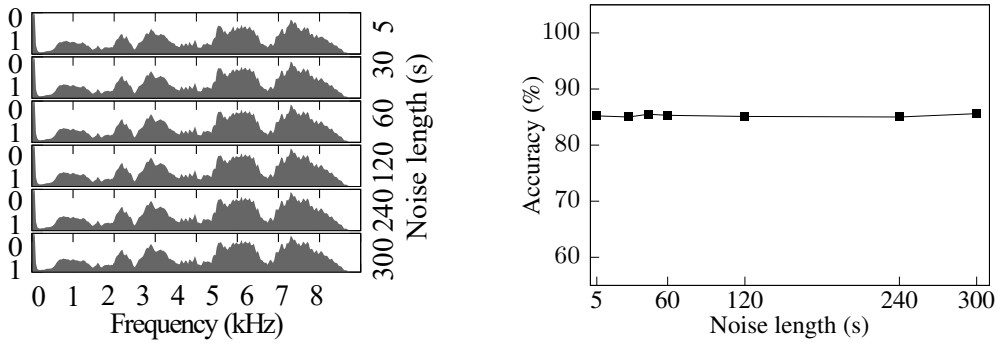
■ **FADA:** When we set the number of labeled target-domain samples per class (LTS/C) to 10 for FADA training, it recovers the accuracy loss for the five microphones by 56%, 38%, 47%, 47%, and 37%, respectively, as shown in Fig. 7. The performance of FADA increases with LTS/C. When we increase LTS/C to 20, PhyAug still outperforms FADA. Note that PhyAug requires a single unlabeled target-domain sample only. In addition, from our experience, FADA is sensitive to hyperparameter settings.

■ **CDCL:** The amount of the used target-domain data per class for contrastive learning is 200. As shown in Fig. 7, CDCL recovers the accuracy loss for the five microphones by 50%, 25%, 56%, 53%, and 27%, respectively. However, the performance of CDCL is sensitive to the amount of target-domain data used for contrastive feature learning. PhyAug outperforms CDCL even when the number of the used target-domain data samples per class increases up to 400.

3.4 In-depth Analysis

3.4.1 Data translation performance of different approaches. We investigate the data translation performance for each approach using the T-distributed Stochastic Neighbor Embedding (t-SNE) [31]. t-SNE is a dimensionality reduction technique to effectively visualize the high-dimensional data in the low-dimensional feature space. As shown in Fig. 9, the red dots labeled "Source" represent the source-domain data, i.e., the original keyword spotting data. The green dots labeled "Target" represent the target-domain data. The target-domain data presented is collected by the microphone M4. The blue dots in each subplot represent the translated data using different approaches. Ideally, the data samples of the same color should cluster together. Moreover, the data translated by an approach from the source-domain data should be close to the target-domain data. To simplify the characterization of the translation effectiveness, we define a metric $r = \frac{d_s}{d_t}$, where d_s is the average distance between the source-domain data points and the corresponding translated data points in the t-SNE space and d_t is the average distance between the target-domain data points and the corresponding translated data points. If the value of r is less than 1, the translated data is closer to the source-domain data; otherwise, the translated data is closer to the target domain. Fig. 9d shows the data translation performance for PhyAug. PhyAug applies the learned FRC to translate the source-domain data to the target domain. Thus, the translated data via PhyAug is expected to be closer to the target-domain data. The distance ratio r for PhyAug is 10, indicating that the translated data via PhyAug is closer to the target-domain data. Thus, when we apply the model trained using the translated data on the target-domain data, we obtain performance improvement. Fig. 9a shows the data translation performance for the Data calibration approach. Data calibration

uses the learned FRC to calibrate the target-domain data back to the source domain. Thus the translated data via Data calibration is expected to be closer to the source domain. The distance ratio r for Data calibration is 7, indicating that the calibrated data are closer to the target-domain data. When we apply the trained DNN from the source-domain data on the calibrated data, the performance improvement is limited. Fig. 9b shows the data translation performance for CDA. CDA uses a *heterogeneity generator* to construct the augmented data that can contain the target-domain data. Thus the augmented data is expected to be close to the target-domain data. The distance ratio r for CDA is 0.2, indicating that the augmented data are closer to the source-domain data. When we apply the DNN trained using the augmented dataset on the target-domain data, the improvement is limited. Fig. 9c shows the data translation performance for CycleGAN. CycleGAN trains a data translation model that tries to map the data between the source domain and the target domain. As the data is translated from the target domain, the translated data is expected to be closer to the source domain. However, we observe that the translated data are far from both the source-domain and the target-domain data. Thus, CycleGAN is observed the performance drop on microphone M4 in the KWS case study. Fig. 9e shows the data translation performance for CDCL. This approach applies the contrastive learning to learn the feature representation such that the domain distance between the source-domain data and the target-domain data is minimized. The translated data is expected to be close to the source-domain data. The distance ratio r for CDCL is 0.4, indicating that the translated data is closer to the source-domain data. Thus, the learned feature representation can reduce the domain difference to a certain extent. However, CDCL requires substantial target-domain data in order to train the DNN model. In summary, PhyAug outperforms the competing baselines in terms of the data translation quality, and it achieves the best results for domain adaptation.



(a) FRCs of M1 measured with various noise emission times.

(b) PhyAug test accuracy with various noise emission times.

Fig. 10. The impact of white noise emission time on the effectiveness of PhyAug.

3.4.2 Amount of target-domain data needed by each approach. In this section, we investigate the amount of target-domain data needed by each approach in order to achieve satisfactory performance.

CDA, Data calibration and PhyAug use white noise to profile the microphones. They do not require the target-domain data. In the previous experiments, the microphone profiling uses a 5-minute noise. We conduct experiments to investigate the impact of shorter noise emission durations on the performance of CDA, Data calibration and PhyAug. Since they use the same FRC to perform data translation, we focus on the PhyAug on a specific microphone, M1. Fig. 10 shows the test accuracy of PhyAug using the M1's FRCs measured with various noise emission times. We can

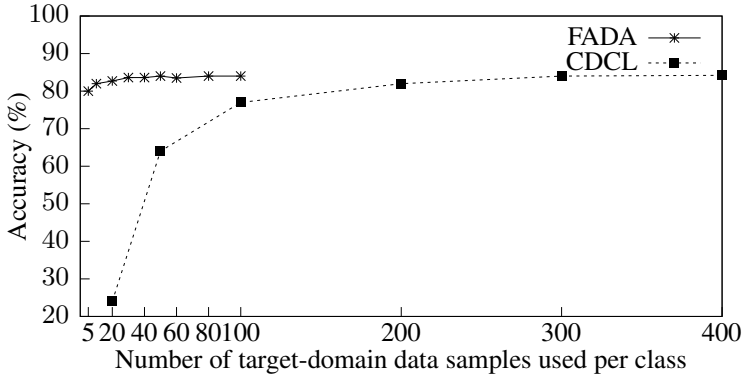


Fig. 11. Evaluation of the amount of the target-domain data needed for CDCL and FADA.

see that a noise emission time of five seconds is sufficient. This result shows that a minimum of 5-second white noise is sufficient to profile a microphone. Thus, Data calibration and PhyAug incur little overhead. CDA is different from Data calibration and PhyAug as it requires a *heterogeneity generator* to generate augmented training data. The construction of the *heterogeneity generator* requires 10-minute white noise from each microphone.

CDCL, FADA and CycleGAN require both the source-domain and the target-domain data for model training. We investigate the performance of CDCL and FADA when the amount of used target-domain data varies. The plot labeled "CDCL" in Fig. 11 shows the CDCL's test accuracy with respect to the used target-domain data. The horizontal axis represents the number of the target-domain data samples used per class; the vertical axis shows the test accuracy. We observe that CDCL's performance increases when the used target-domain data amount increases. Its performance stabilizes when the number of used data samples per class in the target domain is greater than 200, which is around 5% of the available training data. The plot labeled "FADA" in Fig. 11 shows the FADA's test accuracy with respect to the used target-domain data. We can see that FADA achieves good performance with 20 data samples used in each class, which is around 0.5% of the available training data. Despite that CDCL and FADA only require a small portion of target-domain data to achieve good results. PhyAug is preferred for model transfer as it only requires a short noise emission time and does not require the target-domain data.

3.5 Application Considerations

From the above results, PhyAug is desirable for KWS on virtual assistant systems. We envisage that more home IoT devices (e.g., smart lights and smart kitchen appliances, etc.) will support KWS. To apply PhyAug, the appliance manufacturer can offer the microphone profiling function as a mobile app and the model transfer function as a cloud service. Thus, the end user can use the app to obtain the FRC, transmit it to the cloud, and receive the customized KWS DNN. As the KWS DNN is not very deep and PhyAug is a one-time effort for each device, the model retraining in the cloud is an acceptable overhead to trade for better KWS accuracy over the entire device lifetime.

4 CASE STUDY 2: AUTOMATIC SPEECH RECOGNITION (ASR)

ASR models often have performance degradation after deployments. This section shows the impact of the microphone on ASR and applies PhyAug to mitigate the impact.

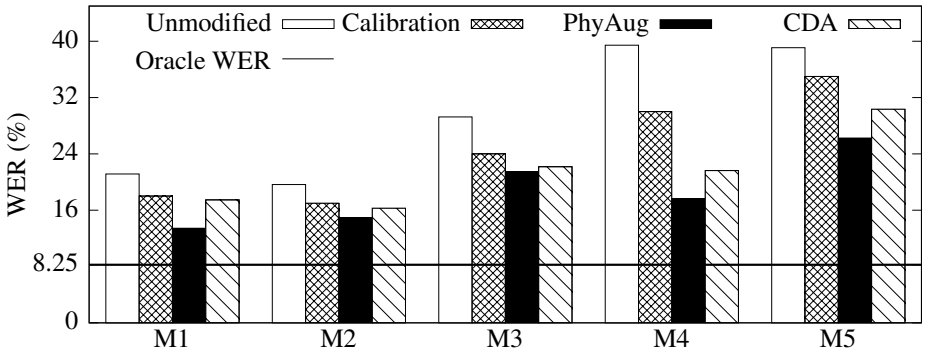


Fig. 12. WERs using various approaches on tested microphones. Compared with the unmodified baseline, PhyAug reduces WER by 60%, 41%, 37%, 70%, and 42% respectively for the five microphones toward the oracle WER. As CycleGAN gives high WERs (about 90%), it is not shown.

4.1 Impact of Microphone on ASR

We use LibriSpeech [19] as the standard dataset in this case study. It contains approximately 1,000 hours of English speech corpus sampled at 16 ksps. Each sample is an utterance for four to five seconds. We use an implementation [16] of Baidu DeepSpeech2, which is a DNN-based end-to-end ASR system exceeding the accuracy of Amazon Mechanical Turk human workers on several benchmarks. The used DeepSpeech2 model is pre-trained with LibriSpeech training dataset and achieves an 8.25% word error rate (WER) on LibriSpeech test dataset. This 8.25% WER is referred to as *oracle WER*. Note that the input to DeepSpeech2 is the spectrogram of a LibriSpeech sample, which is constructed from the Fourier frames using a 20-millisecond window size and 10-millisecond window shift.

DeepSpeech2 has 11 hidden layers with 86.6 million weights. It is far more complicated than the KWS CNN. Specifically, DeepSpeech2 is 175 times larger than the KWS CNN in terms of the weight amount. All the existing studies (e.g., Mic2Mic [12], MetaSense [5], and CDA [13]) that aimed at addressing domain shift problems in voice sensing only focused on simple tasks like KWS and did not attempt a sophisticated model such as DeepSpeech2.

We test the performance of the pre-trained DeepSpeech2 on the five microphones M1 to M5 used in §3. We follow the same test methodology as presented in §3.1.2. In Fig. 12, the histograms labeled "unmodified" represent the WERs of the pre-trained DeepSpeech2 on the test samples recorded by the five microphones. The horizontal line in the figure represents the *oracle WER*. We can see that the microphones introduce about 15% to 35% WER increases. In particular, the two low-end microphones M4 and M5 incur the highest WER increases. This result is consistent with the intuition. From the above test results, this section investigates whether PhyAug described in §3 for KWS is also effective for ASR. Different from the KWS CNN that takes MFCC tensors as the input, DeepSpeech2 takes the spectrograms as the input. Thus, in this case study, PhyAug does not need to convert spectrograms to MFCC tensors in the data augmentation.

4.2 Performance Evaluation

4.2.1 Comparison with alternative approaches. We use data calibration, CDA [13], and CycleGAN (i.e., essence of [12]) described in §3.3.1 as the baselines. FADA [15] cannot be readily applied to DeepSpeech2, because FADA requires class labels while DeepSpeech2 performs audio-to-text conversion without the concept of class labels. Differently, PhyAug and the three used baselines transform data without needing class labels.

■ **Data calibration:** Its results are shown by the histograms labeled "calibration" in Fig. 12. Compared with the unmodified baseline, this approach reduces some WERs.

■ **PhyAug:** Among all tested approaches, PhyAug achieves the lowest WERs for all microphones. Compared with the unmodified baseline, PhyAug reduces WER by 60%, 41%, 37%, 70%, and 42%, respectively, for the five microphones toward the oracle WER.

■ **CDA [13]:** It performs better than the data calibration approach but worse than PhyAug. As PhyAug is directed by the target microphone's actual characteristics, it outperforms CDA that is based on the *predicted* characteristics that may be inaccurate.

■ **CycleGAN:** We record a 3.5-hour speech dataset and use it to train a CycleGAN to translate samples captured by a target microphone to the source domain of LibriSpeech dataset. Unfortunately, DeepSpeech2's WERs on the data translated by CycleGAN from the microphones' samples are higher than 90%. A possible reason is as follows. Unlike the KWS task studied in Mic2Mic [12] and §3 of this paper, which discriminates a few target classes only, end-to-end ASR is much more complicated. CycleGAN may require much more training samples beyond we use to achieve good performance.

4.2.2 Impact of various factors on PhyAug. We also evaluate the impact of the following three factors on PhyAug: the indoor location of the microphone, the distance between the microphone and the sound source, and the environment type. We evaluate the impact of the following three factors on PhyAug: the indoor location of the microphone, the distance between the microphone and the sound source, and the environment type. We adopt an evaluation methodology as follows. When we evaluate the impact of a factor, the remaining two factors are fixed. For a certain factor, let X and Y denote two different settings of the factor. We use PhyAug(X,Y) to denote the experiment in which the microphone profiling is performed under the setting X and then the transferred model is tested under the setting Y. Thus, PhyAug(X,X) evaluates *in situ* performance; PhyAug(X,Y) evaluates the sensitivity to the factor.

■ **Impact of microphone location:** Microphones at different locations of an indoor space may be subject to different acoustic reverberation effects. We set up experiments at three spots, namely, A, B, and C, in a $7 \times 4\,\mathrm{m}^2$ meeting room. Spot B is located at the room center; Spots A and C are located at two sides of B, about 1 m apart from B along the room's long dimension. Fig. 13 shows the results of the unmodified baseline approach tested at three spots, as well as PhyAug's *in situ* performance and location sensitivity. PhyAug's *in situ* WERs (i.e., PhyAug(A, A), PhyAug(B, B), PhyAug(C, C)) are consistently lower than those of the unmodified baseline. The WERs of PhyAug(A, B) and PhyAug(A, C) are slightly higher than PhyAug(B, B) and PhyAug(C, C), respectively.

Similarly, we evaluate the impact of the microphone locations on CDA and Data calibration approaches. Fig. 14 and Fig. 15 show the results of CDA and Data calibration, respectively. Similar to PhyAug, we observe that the WERs are consistently lower than the unmodified results at three tested locations for both approaches, and the WERs of (A, B) and (A, C) are slightly higher than (B, B) and (C, C). These results show that location affects the performance of a certain ASR model transferred by all evaluated approaches, but not much. Thus, CDA and Data calibration also exhibit similar robustness trends as PhyAug.

■ **Impact of microphone-speaker distance:** The distance affects the signal-to-noise ratio (SNR) received by the microphone and thus ASR performance. With the setup at the aforementioned Spot C, we vary the distance between the microphones and the iPhone 7 used to play test samples to be 75 cm, 45 cm, and 15 cm (referred to as D_1, D_2, and D_3). Fig. 16 shows the results. The unmodified baseline's WERs become lower when the microphone-speaker distance is shorter, due to the increased SNR. PhyAug's *in situ* WERs (i.e., PhyAug(D_1,D_1), PhyAug(D_2,D_2), and PhyAug(D_3,D_3))

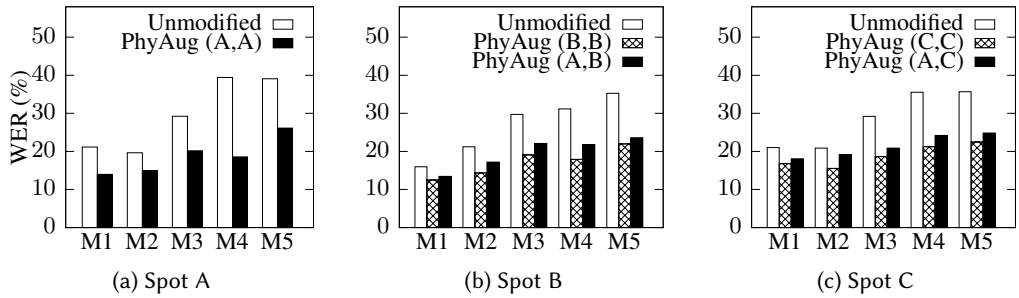


Fig. 13. PhyAug's *in situ* performance and location sensitivity evaluated at three spots in a $7 \times 4\,\mathrm{m}^2$ meeting room.

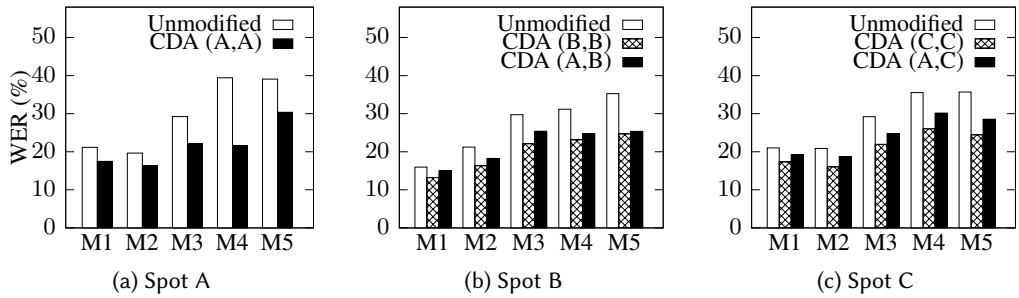


Fig. 14. CDA's *in situ* performance and location sensitivity evaluated at three spots in a $7 \times 4\,\mathrm{m}^2$ meeting room.

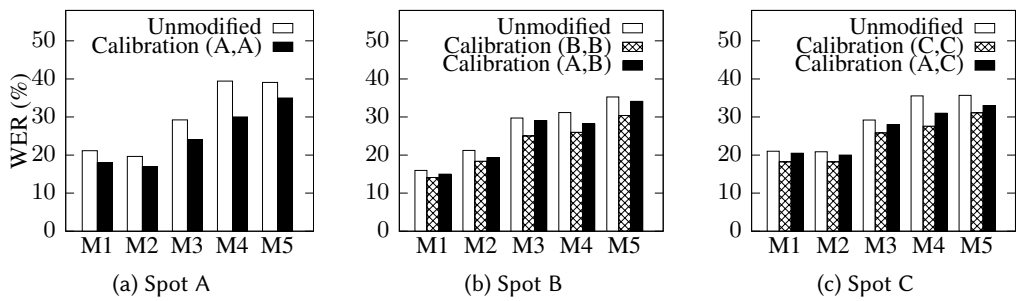


Fig. 15. Data calibration's *in situ* performance and location sensitivity evaluated at three spots in a $7 \times 4\,\mathrm{m}^2$ meeting room.

are consistently lower than those of the unmodified baseline. The performance gain is better exhibited when the distances are longer. This suggests that *in situ* PhyAug improves the resilience of DeepSpeech2 against weak signals. In most cases, the WERs of PhyAug(D_1,D_2) and PhyAug(D_1,D_3) are slightly higher than those of PhyAug(D_2,D_2) and PhyAug(D_3,D_3), respectively. This shows that the microphone-speaker distance affects the performance of a certain model transferred by PhyAug, but not much. Thus, PhyAug for DeepSpeech2 is insensitive to the microphone-speaker distance.

Another related factor is the speaker's azimuth with respect to the microphone that can affect the quality of the recorded signal due to the microphone's polar-pattern characteristic. For a certain

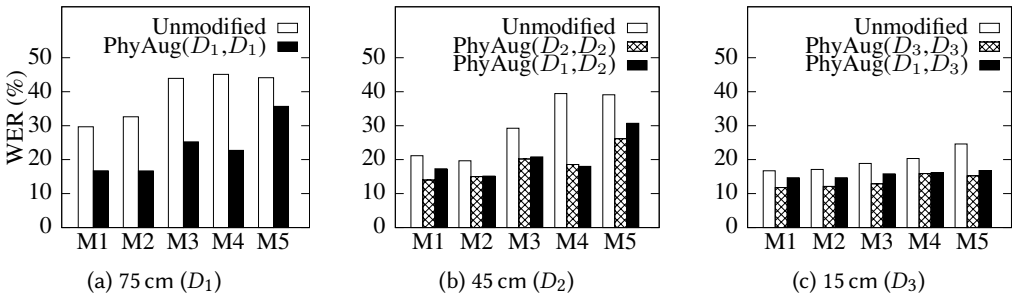


Fig. 16. PhyAug's *in situ* performance and microphone-speaker distance sensitivity evaluated with three distances.

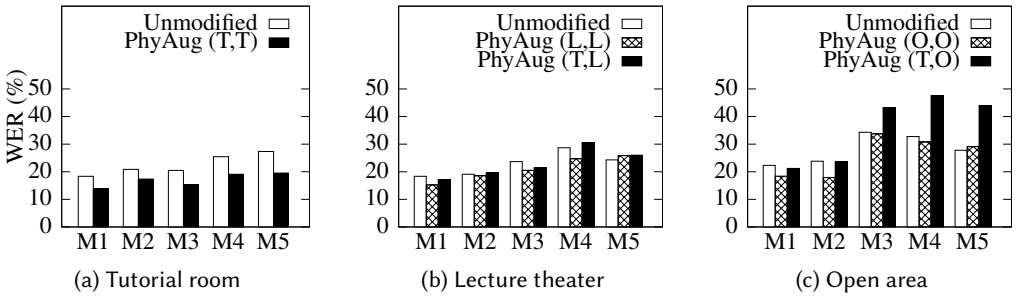


Fig. 17. PhyAug's *in situ* performance and environment sensitivity evaluated in three types of environment, namely, small tutorial room (T), large lecture theater (L), and outdoor open area (O).

microphone, the different azimuths of the speaker create multiple target domains. If the speaker's azimuth can be sensed (e.g., by a microphone array), PhyAug can be applied. However, as the five microphones used in this paper lacks speaker azimuth sensing capability, we skip the application of PhyAug to address the domain shifts caused by the speaker's azimuth.

■ **Impact of environment:** Different types of environments in general have distinct acoustic reverberation profiles, which may affect the microphone's signal reception. We deploy our experiment setup in three distinct types of environments: a small tutorial room (T), a large lecture theatre (L), and an outdoor open area (O). Fig. 17 shows the results. The unmodified baseline approach has similar results in T and L. Its WERs become higher in O, because O has a higher level of background noise. PhyAug's *in situ* WERs in T, i.e., PhyAug(T,T), are consistently lower than those of the unmodified baseline. PhyAug(L,L) and PhyAug(O,O) reduce WERs compared with the unmodified baseline, except for the low-quality microphone M5. As M5 has higher noise levels, the microphone profiling process may not generate fidelity FRCs for M5, leading to increased WERs. As shown in Figs. 17b and 17c, the WERs of PhyAug(T,L) and PhyAug(T,O) are higher than those of the unmodified baseline. The above results show that PhyAug for DeepSpeech2 may have degraded performance on low-quality microphones. In addition, PhyAug for DeepSpeech2 is sensitive to various environments.

4.3 Application Considerations

From results presented in §4.2, PhyAug suits ASR systems deployed at fixed locations, such as residential and in-car voice assistance systems, as well as minutes transcription systems installed

in meeting rooms. PhyAug can also be applied to the *ad hoc* deployment of ASR and automatic language translation for a multilingual environment.

5 CASE STUDY 3: ACOUSTICS-BASED ROOM RECOGNITION (ARR)

Smartphone indoor localization without using extra sensors and infrastructure is desirable. Recent studies exploit the smartphone's built-in audio system for infrastructure-free room-level indoor localization [25, 28]. Specifically, they use a smartphone to sense a room's acoustic background spectrogram (ABS) [28] or the room's reverberation in response to a probe sound emitted by the smartphone [25]. They follow supervised learning to train a model using labeled data samples collected from multiple rooms. Then, the smartphone with the model can recognize which room it is located in using the ABS or room reverberation sensed by the smartphone. Different from the previous two case studies (KWS, ASR) that aim at interpreting the voices, ARR uses acoustic signals to sense the environment. Since ARR uses a smartphone microphone as the sensor, presumably, its performance can be affected by the heterogeneity of the smartphones' microphones. Specifically, if the target smartphone deployed with the trained ARR model is different from the smartphones or specialized acoustic devices used to collect the training data samples, the performance of the ARR model may drop. Conventional data augmentation approaches [13] may fail to capture such device variability because of a lack of target sensors' domain knowledge. Data translation approaches, e.g., mic2mic [12] only address the single device-to-device data translation. In addition, it requires a translation module installed on each device, hindering the generality of the approach.

In this section, we validate that the main cause of the ARR model performance drop is microphone variability. To address this issue, we apply PhyAug to recover the performance degradation of the ABS-based ARR model when being applied on a specific smartphone. Specifically, we exploit the smartphone's ABS profile to perform data translation from a source smartphone to the target smartphone. Then, we apply the transfer learning technique to obtain a domain-adapted ARR model for the target smartphone.

In this case study, **source domain** is the dataset collected from the smartphone's microphone used to train the ARR model; **target domain** is the dataset captured from a different smartphone; **first principle** is the microphone's FRC.

5.1 Problem Description

In this section, we describe the procedures for ABS feature extraction and DNN model used for room recognition. Then, we measure the impact of smartphone microphone variability on the pre-trained ARR model's accuracy. Finally, we apply PhyAug to recover the model accuracy loss and compare PhyAug with baseline approaches.

5.1.1 ABS feature extraction and DNN model design. We follow the acoustic signal preprocessing steps described in [28] to extract ABS features. An ABS feature is extracted as follows. First, the smartphone records a 1-second long background sound within a room at a sampling rate of 44.1 kHz. Second, we apply short-time Fourier transform (STFT) on the signal by sliding a 1,024-point hamming window with 512 points of overlap to obtain the ABS. Third, we discard the frequencies that are greater than 7kHz and sort the values in each frequency bin. Lastly, we select the 5th percentile of the sorted values in each frequency bin to form a one-dimensional vector with 163 elements as the ABS feature. The sorting and selection make sure the feature characterizes the background sound, rather than transient foreground sound.

Different from the ABS-based ARR system in [28] that uses nearest-neighbor classification, we adopt a 5-layer MLP model that takes the ABS feature as input to perform room recognition. The number of neurons in the five layers are 163, 256, 512, 1024, and N, where N represents the number

of rooms. During training, a 0.4 dropout rate is adopted between any two hidden layers to prevent overfitting. We implement the model using Pytorch [20].

5.1.2 Impact of smartphone microphone variability on ARR. We investigate the impact of the smartphone microphone on a pre-trained ARR model. We use three smartphones of different models (Samsung Galaxy S7, Motorola Moto Z, and Google Pixel 4) to collect 20 rooms' ABS features. For each room, we collect 10-minute training data and 2-minute testing data using each smartphone. We train the DNN model with data collected using a specific smartphone and then test the trained model with data collected using all smartphones. The results are shown by the histograms labeled "unmodified" in Fig. 18a, 18b, and 18c, respectively. Taking Fig. 18a as an illustration, the source device used to train the DNN model is Galaxy S7. The oracle accuracy numbers reported in the sub-figure captions are obtained by training and testing the model on the data collected from the same smartphone, which are 98% for Galaxy S7 and Pixel 4, and 99% for Moto Z. Thus, the ABS-based ARR can achieve high accuracy on recognizing different rooms if the source and target devices are identical. However, from Fig. 18a, when applying the model trained on Galaxy S7 to Pixel 4 and Moto Z, the DNN model's accuracy drops to 17% and 16%, respectively. Similar substantial accuracy drops can be observed when the source smartphone is Pixel 4 or Moto Z. These results show that the ABS-based ARR is highly sensitive to smartphone microphone variability.

To visualize the differences between the acoustic traces collected from different smartphones in the same room, Fig. 18d plots the spectrograms of the phones' 1-second data traces collected at the same time. It shows that different smartphones record different ABSes, which is caused by the microphone heterogeneity. From this observation, the research question is how to exploit the microphone characteristics to recover the ARR DNN's accuracy loss?

5.2 PhyAug for Acoustics-based Room Recognition

Similar to §3.2, PhyAug for ARR consists of *fast microphone profiling* and *model transfer via data augmentation.*

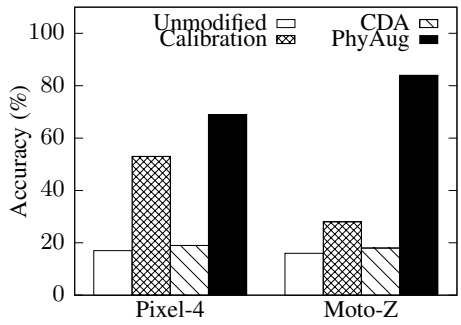
For *fast microphone profiling*, we adopt a new approach that is different from but related to that in §3.2. Specifically, instead of using a speaker to playback the white noise, we place the smartphone in a *profiling room* to record a 1-minute ABS for phone characterization. The profiling room can be different from those to be recognized. We follow the procedure in §5.1.1 to obtain the spectrogram over a 1-minute window. Then, we sort the values in each frequency bin and compute the average of the values from the first percentile to the 20th percentile. The resulted averages for all the frequency bins form the smartphone's *ABS profile* that is specific to the profiling room. We obtain the ABS profiles of the source smartphone and the target phone, which are denoted by $ABS_s(f)$ and $ABS_t(f)$, respectively.

For *model transfer via data augmentation,* we transform the source-domain training data into the target domain. The source-target transfer function is $H(f) = \frac{ABS_t(f)}{ABS_s(f)}$. Then, we multiply the labeled source-domain ABS features with $H(f)$ to generate augmented target-domain ABS features. Finally, we re-train the ARR model with the augmented data.

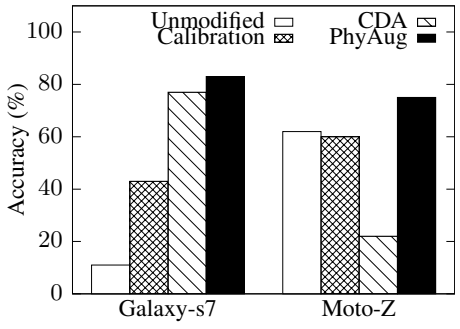
5.3 Performance Evaluation

We compare PhyAug with two alternative approaches: data calibration and CDA. The evaluation results are shown in Fig. 18a, 18b and 18c, which use Galaxy S7, Pixel 4, and Moto Z as the source device, respectively.

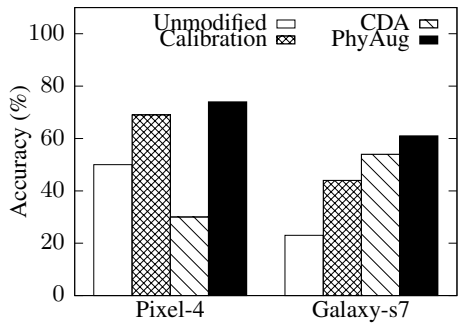
■ **Data calibration:** This approach converts the target domain data back to the source domain, then applies the pre-trained model to the converted data. The histograms labeled "calibration" show



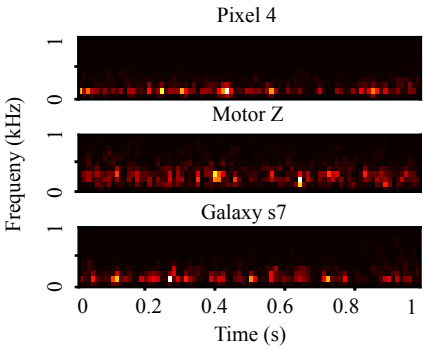
(a) Source device: Galaxy S7 (oracle accuracy: 98%)



(b) Source device: Pixel 4 (oracle accuracy: 98%)



(c) Source device: Moto Z (oracle accuracy: 99%)



(d) ABSes captured by different phones at the same time

Fig. 18. Impact of smartphone microphone variability on ABS-based ARR and comparison of different approaches.

the results of this approach. Compared with the "unmodified" results, this approach recovers a certain amount of the accuracy loss for most source-target phone combinations. However, it leads to lower accuracy when the source and target phones are Pixel 4 and Moto Z. The accuracy drops from 62% to 60%.

■ **CDA:** We follow the scheme presented in §3.3.1 and use target smartphones' ABS profiles to generate a *heterogeneity dataset*. We train a DNN model on this dataset and evaluate on target smartphones. The histograms labeled "CDA" show the results. Compared with the "unmodified" results, we observe considerable accuracy recovery when transferring from Pixel 4 to Galaxy S7 and from Moto Z to Galaxy S7. However, CDA underperforms when transferring between Pixel 4 and Moto Z. The reason is as follows. From the "unmodified" results, any source-target pair involving Galaxy S7 has a poor result. For example, the DNN model's absolute accuracy drops between Pixel 4 with Moto Z is around 30% - 40%, whereas the accuracy drops between Galaxy S7 with Pixel 4 or Galaxy S7 with Moto Z are more than 70%. This implies that Galaxy S7's ABS profile is significantly different from those of Pixel 4 and Moto Z. Under the CDA approach, when we transfer between Pixel 4 and Moto Z, the Galaxy S7 is used as one of the two phones in the heterogeneity dataset.

As a result, the heterogeneity dataset has a complex pattern, which adversely affects the dataset's representativeness.

■ **PhyAug:** PhyAug can recover the accuracy loss for any source-target smartphone pair. In addition, PhyAug achieves the best performance recovery among all evaluated approaches. Specifically, PhyAug recovers 24% to 72% absolute accuracy degradations on different smartphone pairs. In particular, when the source device is Galaxy S7, the "unmodified" accuracies on Pixel 4 and Moto Z are 17% and 16%. PhyAug can recover 52% and 68% absolute accuracy losses, outperforming significantly over the data calibration and CDA approaches.

5.4 Summary

The DNN-based mobile application generally suffers from performance degradation due to the heterogeneity of mobile sensors. This case study applies PhyAug to recover the ARR performance loss caused by smartphone microphone variations. PhyAug only requires smartphones to record 1-minute ABS profiles in a certain room and achieves significant accuracy recovery. This case study shows that PhyAug can be used to address the sensor heterogeneity issue in DNN-based mobile sensing applications.

6 CASE STUDY 4: FISHEYE IMAGE RECOGNITION

DNN-based visual sensing can be found in many IoT applications, including video surveillance [9], augmented reality [24], and autonomous driving [7]. Many such applications use fisheye cameras. The fisheye camera is different from the normal pinhole camera with rectilinear mapping. The fisheye camera produces images with a wide field of view (FOV) while creating strong distortions due to the non-linear mapping of optical lens systems.

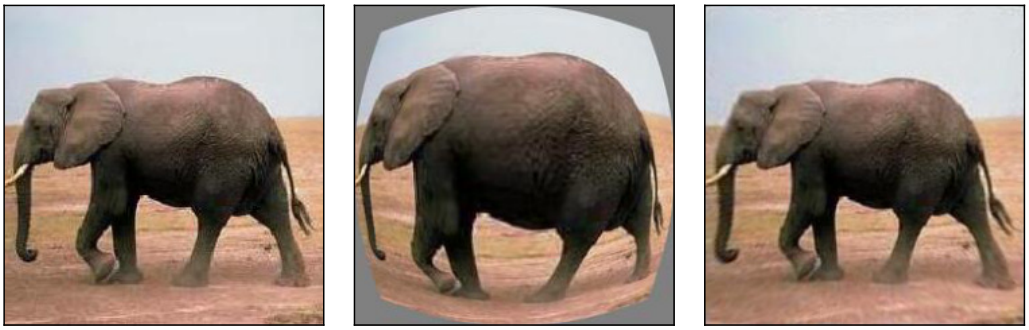
Prevalent image datasets consist of samples obtained using pinhole cameras. Standard DNNs that achieve state-of-the-art performance are also trained and tested on such pinhole camera datasets [2, 34]. They may not perform well on the images collected by fisheye cameras. Image rectification is a conventional approach that applies inverted fisheye models to fix distorted images. However, image rectification has two main limitations [34]. First, as fisheye images contain greatly distorted peripheries, mapping the limited pixels from the image periphery to a larger region leads to information loss. Second, the image rectification requires additional processing time for every image before the image classification. The processing time grows drastically as the image resolution increases. Thus, the image rectification approach is not suitable for applications that impose both deadline and high-resolution requirements, e.g., visual sensing-based pedestrian detection on a moving vehicle.

In this section, we apply PhyAug to recover DNN's performance degradation caused by fisheye camera distortion without performing image rectification. First, we utilize a non-linear polynomial model to augment the original dataset to the images with fisheye distortions. Then, we apply a transfer learning technique to obtain a domain-adapted DNN model for fisheye images.

In this case study, **source domain** is the image dataset that is captured by pinhole cameras and used to train the DNN; **target domain** is the fisheye camera dataset for specific IoT applications; **first principle** is the fisheye camera model described by a non-linear polynomial function.

6.1 Problem Description

In this section, we first introduce the original dataset captured by pinhole cameras. Then, we describe the camera model used to generate synthesized fisheye images. Finally, we compare the performance of PhyAug, CDA, and the image rectification approach.



(a) Original (b) Synthesized (c) Rectified

Fig. 19. Fisheye image sythesization and rectification using parameterized model. (a) original image; (b) synthesized fisheye image; (c) rectified image.

6.1.1 Fisheye camera model. For a given camera, distortions occur when the scenes deviate from the rectilinear projection. The most common source of image distortions is the radial distortion caused by the camera optical lens system. Fisheye cameras produce strong radial distortions on images. Several models have been proposed to characterize a fisheye camera [8]. In this paper, we adopt a generic fisheye model [29], which is a fourth-order polynomial function:

$$R_{src} = r \cdot \left(A \cdot r^3 + B \cdot r^2 + C \cdot r + D\right), \tag{1}$$

where r is the destination image radius and R_{src} is the source pixel. In this model, image radius is normalized, so that $r = 1$ refers to the half minimum width or height of the input image. A, B, C represent the distortion of the image. When the three values are positive, the image contains barrel distortion. When they are negative, pincushion distortion occurs in the image. D describes the linear scaling of the image. The values of A, B, C, D are fixed for a given camera and are often stored as metadata of the image captured by a fisheye camera.

6.1.2 Dataset and deep neural network model. In this case study, our experiments use Caltech-101 dataset [4]. It consists of image objects in 101 classes. Each class contains 40 to 800 images, with a total of around 9,000 images. We split the dataset into training, validation, and testing sets at 70%, 10% and 20% ratio.

We adopt the ResNet-50 CNN model [6] to perform multi-class classification. ResNet-50 consists of 48 convolutional layers along with one max-pooling and one average-pooling layer. The base model is pre-trained on the ImageNet dataset [3]. We customize the model by reducing the output layer size from 1,000 neurons to 101 neurons, to align with Caltech-101's class number. We use *freeze-and-train* to transfer the pre-trained base model for Caltech-101 dataset. In particular, we freeze the weights in feature encoders and update the weights in the fully connected layers. In this case study, we implement the model training and evaluation using PyTorch.

6.1.3 Impact of image distortion on DNN model. We investigate the performance of a DNN model trained on a standard dataset captured by pinhole cameras and test it on images captured by fisheye cameras. Due to the lack of publicly available fisheye image datasets, we use a synthetic dataset for our experiments. We distort the images using the model depicted in Eq. (1). Fig. 19 shows a sample. Fig. 19(a) is the original image. Fig. 19(b) is the corresponding distorted image with a positive parameter set of $A = 0.2, B = 0.2, C = 0.01, D = 0.59$, where a strong radial distortion effect can be observed. We apply the same parameters set on Caltech-101 to generate the synthetic fisheye

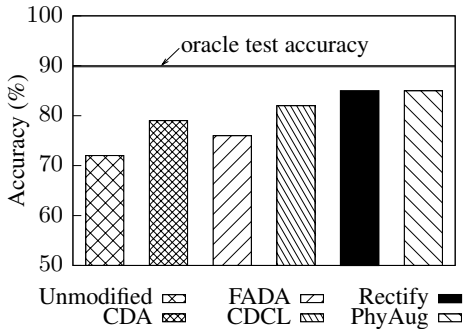


Fig. 20. ResNet-50 test accuracy on Caltech-101 dataset using different approaches.

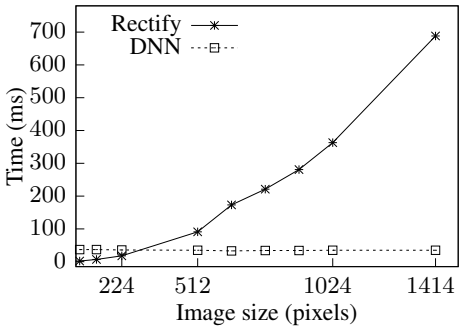


Fig. 21. Execution time vs. image size on an NVIDIA Jetson Nano.

image dataset. The fisheye model is implemented using the Wand package [14] in Python. We test the pre-trained model on both the original and distorted datasets. As shown in Fig. 20, the pre-trained ResNet-50 model achieves 90% oracle accuracy on the original test set. The accuracy on the distorted dataset is 72%. Thus, there is an 18% absolute accuracy drop. Many fisheye cameras can produce images with FOV greater than 180°, resulting in stronger non-linear distortions. Hence, significant accuracy drops can be observed. Based on the experiment results, the research question for this case study is: how to exploit the first principle of fisheye camera models to recover the image classification DNN's accuracy loss?

6.2 PhyAug for Fisheye Image Recognition

We use a generic fisheye model, which is described by a fourth-order polynomial function (cf. Eq. (1)). Note that a fourth-order polynomial function can well represent the model of fisheye cameras, while higher orders provide no additional benefit in terms of accuracy [34]. A system designer can reconstruct the camera model based on lens parameters stored in the image's metadata. In case that the metadata is missing, a standard calibration procedure can be applied to estimate a camera's intrinsic parameters [23]. Specifically, the camera can capture photos of a printed image with a known pattern, e.g., a chessboard, at different angles. In general, 20 to 30 pictures from a fisheye camera are sufficient to obtain distortion parameters. Such calibration tools are available in OpenCV [17] and Matlab [23].

PhyAug for FIR has the following two steps. First, we estimate the parameters of Eq. (1) for a specific fisheye camera and apply the fisheye model on the original images to generate augmented samples in the target domain. Then, we train the DNN model with the augmented data for the fisheye camera. In this case study, we assume that the parameters of the target camera are known.

6.3 Performance Evaluation

6.3.1 Baseline approaches. We evaluate PhyAug performance against following baseline approaches.

■ **CDA:** This approach follows the conventional data augmentation scheme to build a DNN model robust to camera lens distortions. Specifically, it randomly generates a set of potential fisheye camera models using Eq. (1). Subsequently, CDA applies these camera models to augment the training dataset. In our evaluation, we randomly generate 10 sets of parameters and use them to augment the Caltech-101 training dataset.

■ **FADA:** This approach follows the adversarial domain adaptation as presented in §3 to train a domain-invariant feature encoder using paired source-domain and target-domain images. The

feature encoder used is a ResNet-50 model and the discriminator used is a 4-layer MLP with the neurons in the four layers are 1024, 1024, 1024 and 101. The number of images in each class used for feature encoder training is set to 100 due to the relatively complex feature space for this case study.

■ **CDCL:** This approach aims to train a domain-invariant feature encoder by applying the contrastive learning. We follow similar procedures as presented in §3 to construct the positive and negative samples for training. The number of images in each class used for contrastive feature training is 400.

■ **Image rectification:** This approach follows the conventional image rectification scheme to correct image distortions. Once the camera lens distortion parameters are determined, one can tune the same model as in Eq. (1) to rectify the image. Fig. 19(c) shows the rectified result by applying the inverted parameters on the distorted image. In our evaluation, we apply the estimated parameters to rectify distorted images and then evaluate the pre-trained model on rectified images.

6.3.2 Evaluation results. We apply PhyAug and the baseline approaches on the Caltech-101 dataset. The results are presented as follows:

■ **CDA:** As shown in Fig. 20, CDA achieves 79% accuracy on the target fisheye dataset. This approach achieves higher accuracy than the unmodified result, which directly applies the pre-trained model to the target test data. However, there is still an 11% accuracy gap towards the oracle accuracy. The result shows that the model trained with CDA mitigates the impact of fisheye camera distortion.

■ **FADA:** As shown in Fig. 20, FADA achieves 76% accuracy on the fisheye images. It only gives 4% absolute accuracy gain compared with the unmodified result. The performance increase is subtle even we increase the used target-domain images for model training. FADA does not perform well on FIR compared to the KWS. The reasons are two-fold. First, the data complexity of FIR dataset is higher than KWS dataset. The size of a fisheye image is $3 \times 224 \times 224$, whereas the size of KWS MFCC is $1 \times 101 \times 40$. Second, the model complexity used is much higher. We use the ResNet-50 model for FIR and use the ResNet-18 for KWS. Thus, it is more difficult to adapt the DNN for FIR using limited data. We conclude that FADA does not generalize well on the complex tasks for domain adaptation.

■ **CDCL:** As shown in Fig. 20, CDCL achieves 82% test accuracy on the fisheye images, representing a 10% absolute accuracy increase. The result shows that CDCL can effectively learn a feature embedding for both the source-domain and the target-domain data. However, the performance of CDCL is sensitive to the number of the target-domain data used for contrastive feature training.

■ **Image Rectification:** We use pre-defined fisheye model in Eq. (1) to rectify the fisheye image, then apply the pre-trained model on rectified images. As shown in Fig. 20, image rectification achieves 85% test accuracy. This shows that image rectification can effectively recover information loss caused by Eq. (1). However, as the image rectification algorithm is applied on every image, thus will incur extra computing time. The evaluation of image rectification time on a resource-constrained device is presented in the next section.

■ **PhyAug:** We apply re-trained model on the distorted image dataset. The test accuracy is 85%. PhyAug can effectively recover the accuracy loss caused by camera distortions. However, there is still a 5% gap compared with the oracle test accuracy. This is because of the information loss when applying the fisheye camera distortion model on the original image data. As shown in Fig. 19, the occupied region of the distorted image (b) is smaller than the original image (a). The gray area in image (b) represents the amount of lost content caused by the distortion. Therefore, a certain amount of accuracy loss is expected.

We evaluate the execution time of image rectification and DNN inference on an NVIDIA Jetson Nano that can execute DNN models. It is equipped with a quad-core Cortex-A57 CPU, a 128 core

Maxwell GPU and 4GB RAM. Results are shown in Fig. 21. The curve labeled "Rectify" is the time taken to rectify an image with respect to the image resolution. The horizontal axis represents the length of a squared image in pixels, e.g., point 224 refers to a 224 × 224 RGB image. The solid line in Fig. 21 shows that image rectification requires more processing time when the resolution increases. It takes around 18 ms to rectify an image of size 224 × 224. When processing an image of 1414 × 1414 pixels, the rectification time increases up to 688 ms. The curve labeled "DNN" is DNN inference time on images with different resolutions. Note that we run image rectification algorithm on CPU as there is lack of GPU version. In practice, there will incur overhead to create GPU compilable program for every edge device. The dotted line in Fig. 21 shows that the inference time is consistent across all tested images, which is around 36 ms. This is because the floating-point operation for a pre-defined neural network is generally fixed and regardless of input image size. Our experiment shows that ResNet-50 network can work effectively on images with different input sizes. In time-critical applications like autonomous driving, large delays are unacceptable. As PhyAug directly adapts the DNN model to the target domain, it has the advantage of avoiding the time-consuming rectification on resource-constrained devices.

6.4 Summary

This case study applies PhyAug to a visual sensing application. Applying DNNs trained using standard pinhole camera image datasets on fisheye images suffers performance degradation. Despite the prevalent use scenarios of fisheye cameras in visual sensing applications, few publicly fisheye datasets are available for training customized DNNs. We identify that the main contributor to the performance drop is the non-linear mapping of the fisheye camera. We apply parameterized fisheye model to transfer existing DNNs to a specific fisheye camera via guided data augmentation. The results show that PhyAug can significantly recover the accuracy loss; while requiring no data collection effort in the target domain of the fisheye camera. Our experiment on NVIDIA Jetson Nano shows that PhyAug requires less computational overhead than the conventional image rectification approach.

7 DISCUSSIONS

In many sensing systems, the domain shifts are often governed by first principles. The four case studies have demonstrated the advantages of exploiting the first principles in dealing with domain shifts that are often experienced by deployed sensing systems. In practice, the complexity of the identified first principles and the amount of data available for first principle fitting vary from application to application. The quality of the fitted models affects the performance of the domain adaptation. Though it is desirable to develop the theoretical analysis to describe how much the fitted first principle can capture the true relations between the source-domain data and the target-domain data, such analysis will need to be based on certain assumptions that are application-specific. Intuitively, the first principle described with a more complex parametric model will require more data for fitting. The focus of this paper is to establish the steps to exploit the physics governing the domain shifts for domain adaptation and show its applicability to a number of case studies.

For the applications that lack useful first principles, we may fall back to the existing physics-regardless transfer learning approaches. However, the fallback option should not discourage us from being discerning on the exploitable first principles in the pursuit of advancing and customizing deep learning-based sensing in the domain of physics-rich cyber-physical systems.

8 CONCLUSION AND FUTURE WORK

This paper described PhyAug, an efficient data augmentation approach to deal with domain shifts governed by first principles. We presented the applications of PhyAug to four case studies of

keyword spotting, automatic speech recognition, acoustics-based room recognition, and fisheye image recognition. They have distinct objectives and require deep models with quite different architectures and scales. The extensive and comparative experiments show that PhyAug can recover significant portions of accuracy losses caused by sensors' characteristics. In addition, it reduces target-domain training data sampling complexity in dealing with the domain shifts.

The future work may consider applying PhyAug to exploit the following two parametric models. First, room impulse response (RIR) describes indoor audio processes. Voice-based smart appliances can exploit RIR as the first principle for effective adaptations to the deployment environments. Active acoustic sensing-based indoor localization with deep learning [25] can exploit RIR to reduce target-domain training data sampling complexity. Second, computational fluid dynamics (CFD) describes the thermal processes in indoor spaces (e.g., data centers). A trained deep reinforcement learning-based environment condition controller can adapt to new spaces with CFD models and a few data samples in each new space.

REFERENCES

[1] Ali Akbari and Roozbeh Jafari. 2019. Transferring activity recognition models for new wearable sensors with deep generative domain adaptation. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. 85–96.

[2] Zhen Chen and Anthimos Georgiadis. 2018. Parameterized Synthetic Image Data Set for Fisheye Lens. In *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*. IEEE, 370–374.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[4] Li Fei-Fei, Rob Fergus, and Pietro Perona. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*. IEEE, 178–178.

[5] Taesik Gong, Yeonsu Kim, Jinwoo Shin, and Sung-Ju Lee. 2019. Metasense: few-shot adaptation to untrained conditions in deep mobile sensing. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 110–123.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[7] Jonathan Horgan, Ciarán Hughes, John McDonald, and Senthil Yogamani. 2015. Vision-based driver assistance systems: Survey, taxonomy and advances. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2032–2039.

[8] Ciaran Hughes, Martin Glavin, Edward Jones, and Patrick Denny. 2008. Review of geometric distortion compensation in fish-eye cameras. (2008).

[9] Hyungtae Kim, Jaehoon Jung, and Joonki Paik. 2016. Fisheye lens camera based surveillance system for wide field of view monitoring. *Optik* 127, 14 (2016), 5636–5646.

[10] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, Vol. 2. Lille.

[11] Wenjie Luo, Zhenyu Yan, Qun Song, and Rui Tan. 2021. Phyaug: Physics-directed data augmentation for deep sensing model transfer in cyber-physical systems. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021)*. 31–46.

[12] Akhil Mathur, Anton Isopoussu, Fahim Kawsar, Nadia Berthouze, and Nicholas D Lane. 2019. Mic2mic: using cycle-consistent generative adversarial networks to overcome microphone variability in speech systems. In *Proceedings of the 18th international conference on information processing in sensor networks*. 169–180.

[13] Akhil Mathur, Tianlin Zhang, Sourav Bhattacharya, Petar Velickovic, Leonid Joffe, Nicholas D Lane, Fahim Kawsar, and Pietro Lió. 2018. Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 200–211.

[14] Hong Minhee. 2021. *Magic Wand.* https://pypi.org/project/Wand/

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

[16] Sean Naren. 2020. deepspeech2.pytorch. https://github.com/SeanNaren/deepspeech.pytorch.

[17] opencv. 2021. *Fisheye camera model.* https://docs.opencv.org/4.5.2/db/d58/group__calib3d__fisheye.html

[18] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.
[19] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 5206–5210.
[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.
[21] Matthias Pohl, Michael Schaeferling, Gundolf Kiefer, Plamen Petrow, Egmont Woitzel, and Frank Papenfuß. 2014. Leveraging polynomial approximation for non-linear image transformations in real time. *Computers & Electrical Engineering* 40, 4 (2014), 1146–1157.
[22] Dipanjan Sarkar, Raghav Bali, and Tamoghna Ghosh. 2018. *Hands-On Transfer Learning with Python.* Packt Publishing.
[23] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. 2006. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.
[24] Dieter Schmalstieg and Tobias Hollerer. 2016. *Augmented reality: principles and practice.* Addison-Wesley Professional.
[25] Qun Song, Chaojie Gu, and Rui Tan. 2018. Deep room recognition using inaudible echos. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2018).
[26] Russell Stewart and Stefano Ermon. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.
[27] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. 2020. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering* 361 (2020), 112732.
[28] Stephen P Tarzia, Peter A Dinda, Robert P Dick, and Gokhan Memik. 2011. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*. 155–168.
[29] Anthony Thyssen. 2009. *ImageMagick v6 Examples – Distorting Images.* https://legacy.imagemagick.org/Usage/distorts/
[30] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7167–7176.
[31] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[32] Rui Wang, Zuxuan Wu, Zejia Weng, Jingjing Chen, Guo-Jun Qi, and Yu-Gang Jiang. 2022. Cross-domain contrastive learning for unsupervised domain adaptation. *IEEE Transactions on Multimedia* (2022).
[33] Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* (2018).
[34] Senthil Yogamani, Ciarán Hughes, Jonathan Horgan, Ganesh Sistu, Padraig Varley, Derek O'Dea, Michal Uricár, Stefan Milz, Martin Simon, Karl Amende, et al. 2019. Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9308–9318.
[35] Chengcheng Yu, Xiaobai Liu, and Song-Chun Zhu. 2017. Single-Image 3D Scene Parsing Using Geometric Commonsense. In *International Joint Conference on Artificial Intelligence*.
[36] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. 2017. Hello edge: Keyword spotting on microcontrollers. (2017).