Audio intelligence & domain adaptation for deep learning models at the edge

Ng, Linus JunJia

2021

Ng, L. J. (2021). Audio intelligence & domain adaptation for deep learning models at the edge. Master's thesis, Nanyang Technological University, Singapore.
<https://hdl.handle.net/10356/152683>

<https://hdl.handle.net/10356/152683>

<https://doi.org/10.32657/10356/152683>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 19:15:34 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**Audio Intelligence &
Domain Adaptation
for Deep Learning Models
at the Edge**

A thesis submitted to the
School of Electrical and Electronic Engineering
of Nanyang Technological University

by

Linus Ng

In Partial Fulfilment for the Degree of Master of Engineering

Supervisor: Professor Gan Woon Seng

September 2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

September 13, 2021

Date

Linus Ng JunJia

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

September 13, 2021

Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU

Professor Gan Woon Seng

Authorship Attribution Statement

This thesis contains material from one paper published in the following peer-review journal in which I am listed as an author.

Most parts of Chapters 3 and 4 are published as Tan Ee Leng, Furi A. Karnapi, **Linus Ng JunJia**, Kenneth Ooi, and Gan Woon Seng, “Extracting Urban Sound Information for Residential Areas in Smart Cities Using an End-to-End IoT System”, in *IEEE Internet of Things Journal* (Mar. 2021).

The contributions of the co-authors are as follows:

- Karnapi designed the implementation framework of the IoT devices.
- Tan implemented the directional of arrival algorithms.
- I implemented deep learning models for environmental sound classification task for the embedded systems.
- Ooi and I collected and annotated the audio data.
- I setup the preprocessing framework necessary for data preparation.
- I prepared the datasets, trained the model, and collated the results.
- Tan, Karnapi, and I prepared the manuscript together. The manuscript was revised by Ooi and Prof Gan.

September 13, 2021

Date

Linus Ng JunJia

Acknowledgement

The author would like to express sincere gratitude and appreciation to Professor Gan Woon Seng for his enthusiastic encouragement, valuable guidance, and excellent supervision throughout this research.

The author would like to acknowledge Smart Nation Translational Lab (NTU) and the School of Electrical and Electronic Engineering, Nanyang Technological University, for providing the research facilities, which made this research possible.

The author would also like to express his sincere gratitude to his colleagues, Dr Tan Ee Leng and Mr Furi A. Karnapi who provided support and encouragement throughout the course of study.

Finally, the author wishes to express his appreciation to his family and everyone who has supported him in this journey.

Abstract

Identifying urban noises and sounds is a challenging but essential problem in the field of machine listening. It enables and provides a realistic use case for detecting noises in residential areas - from noise complaints to detecting sounds or unusual noises that may indicate possible emergencies. To mitigate noise issues in an estate is not an easy task using machine learning approach due to data scarcity and the lack of labeled data, where the acquisition of labeled data is often difficult, costly, and time-consuming. In this work, we leverage an end-to-end IoT system coupled with deep learning models to detect critical urban sound information at the edge. Wireless acoustic sensor nodes (WASN) are deployed in several residential areas to validate their feasibility in detecting noise events of interest, where real-time edge analytic is performed. We explore methods to address the domain shift caused by novel acoustic conditions that are introduced due to environmental influences in different deployed locations, evaluating the environmental sound classifiers in a WASN setup, and the extent it affects the performance of the sound classifiers in different locations with different microphones. We have collected and annotated audio data set in Singapore for training, validating, and testing purposes. Our experimental results show that the proposed method is able to address the mismatch introduced by the domain shift. The proposed method and future research in this work will enhance model robustness in adapting to new deployed environments and minimize the manpower time required to acquire and annotate audio data.

Contents

Statement of Originality	i
Supervisor Declaration Statement	ii
Authorship Attribution Statement	iii
Acknowledgement	iv
Abstract	v
List of Figures	ix
List of Tables	xii
Acronyms	xiv
1 Introduction	1
1.1 Motivation	1
1.2 Research questions	4
1.3 Thesis Organization	5
2 Literature review	6
2.1 Environmental sound classification	6
2.1.1 Overview of ESC with machine learning approach	8
2.1.2 Audio features and preprocessing of audio data	10
2.1.3 Deep learning neural network model architectures	12

2.1.4	Data augmentation techniques	14
2.1.5	Transfer learning	14
2.1.6	Challenges of transfer learning	16
2.2	Real-world deployment	17
2.2.1	Environmental sound classification with wireless acoustic sensor nodes	18
2.2.2	Challenges of real-world deployment	20
2.2.3	Domain adaptation	20
2.3	Conclusion	22
3	Dataset	23
3.1	Taxonomy	24
3.2	Data collection	25
3.3	Data annotation process	28
3.4	Data pre-processing and preparation for experiment setup	29
3.5	Basis of pre-processing parameters	32
3.6	Conclusion	34
4	Methodology	35
4.1	Model Architecture	35
4.1.1	Feature extractor	36
4.1.2	Inception network architecture	37
4.1.3	Classifier	40
4.1.4	Discriminator	40
4.1.5	Architecture summary	41
4.2	Model Training	42
4.2.1	Pre-training stage	42
4.2.2	Domain-adversarial training stage	43
4.2.2.1	Target domain data-split for adversarial training	44
4.2.2.2	Domain-adversarial training	44
4.2.2.3	Gradient Reversal Layer	45

4.2.3	Exporting the models	48
4.3	Conclusion	48
5	Experiment Results and Discussions	49
5.1	Evaluation metrics	49
5.2	Model parameter pruning	50
5.3	Experiment Results	52
5.4	Discussions	60
5.4.1	Overall performance	61
5.4.2	Class-wise performance	61
5.5	Ablation Study	62
5.6	Conclusion	63
6	Conclusions and Future Work	64
6.1	Conclusions	64
6.2	Future Work	67
	Bibliography	69

List of Figures

2.1	Input and expected output of an ESC system.	7
2.2	Overview workflow for training an acoustic model with machine learning.	9
2.3	Usage of the trained acoustic model for deployment.	10
2.4	Time domain audio signals transformed into magnitude spectra and projected to a reduced set of Mel frequency bands and converted to logarithmic magnitudes.	11
2.5	Downstream workflow for ESC with a convolutional neural network (CNN) architecture	12
2.6	Downstream workflow for ESC with a convolutional-recurrent neural network (CRNN) architecture	13
2.7	Flowchart of a typical workflow to train deep learning models for WASN-ESC tasks	17
3.1	The deployed wireless acoustic sensor nodes. From left to right: Location L1, L2 and L3 with microphones M1 [104] at locations L1 and L2 and microphone M2 [105] at location L3.	25
3.2	Microphone M2 [105] - MEMS microphones on circular array aligned to Cartesian coordinate system. At the data collection stage, only Mic 0 (center microphone) is utilized for recording.	27

3.3	Data annotation process on collected audio recordings - the top-most audio waveform corresponds to the collected raw audio data. The audio waveforms in the middle row are snippets of the top-most audio waveform, edited (trimmed and stitched according to the same class type) and labeled. The remaining audio recordings that do not encompass any sound class events are concatenated and labeled as “Ambient”.	28
3.4	Method for generating data samples from annotated recordings . .	29
3.5	Pre-processing parameters illustrated with 3 unique cases.	33
4.1	Overview of the domain adaptation model - incorporates a feature extractor, classifier and discriminator.	36
4.2	Convolutional block	38
4.3	Inception block	38
4.4	Overview of downstream ESC task	41
4.5	Unsupervised domain adaptation - model architecture, training and testing. Black arrows indicate the downstream forward propagation while backpropagation during training is highlighted in red.	47
5.1	Confusion matrix of source only trained model evaluated on unseen test set (L3). The $(i, j)^{\text{th}}$ entry of each matrix denotes the percentage of tracks w.r.t the ground truth label i that are predicted by the model for class j	55
5.2	Confusion matrix of domain adversarial trained model evaluated on unseen test set (L3). The $(i, j)^{\text{th}}$ entry of each matrix denotes the percentage of tracks w.r.t the ground truth label i that are predicted by the model for class j	56
5.3	t-SNE visualizations without (a) and with (b) domain adversarial training.	58

6.1 Proposed workflow for adapting the environmental sound classifier when deploying an acoustic sensor node at a new location - although ground-truth class labels for the collected target domain data are not required during domain adversarial training, it is recommended to set aside a small subset of labeled target domain dataset for testing. 66

List of Tables

2.1	Overview of the public datasets for acoustic scenes and sound events.	19
3.1	Taxonomy of sound classifier	24
3.2	Locations, descriptions, and common sounds observed of the de- ployed sensor nodes	27
3.3	Recording devices of our WASN with corresponding descriptions .	27
3.4	Class-wise audio length and number of samples in train/val/test splits	30
4.1	Feature extractor (Inception) model architecture	39
4.2	Classifier model architecture	40
4.3	Discriminator model architecture	40
4.4	Number of data samples in test sets for adversarial training and for testing	44
5.1	Model pruning - macro-averaged F1-scores of model variants, with and without domain adversarial training	51
5.2	Class-wise precision and recall comparison of the two train- ing methods evaluated on the unseen test set (target domain, L3). The columns Precision Δ and Recall Δ signify the difference in precision and recall between the two training methods.	53

5.3	Class-wise F1-score and AUPRC comparison of the two training methods evaluated on the unseen test set (target domain, L3). The columns F1-score Δ and AUPRC Δ signify the difference in F1-score and AUPRC between the two training methods.	54
5.4	Class-wise precision and recall comparison of the two training methods evaluated on the validation set (source domain, L1 & L2). The columns Precision Δ and Recall Δ signify the difference in precision and recall between the two training methods.	59
5.5	Class-wise F1-score and AUPRC comparison of the two training methods evaluated on the validation set (source domain, L1 & L2). The columns F1-score Δ and AUPRC Δ signify the difference in F1-score and AUPRC between the two training methods.	60
5.6	Ablation study results show that the pre-training (stage one) of our training method is essential for better performance.	62

Acronyms

AASP	Audio and Acoustic Signal Processing
ASC	Acoustic Scene Classification
ASR	Automatic Speech Recognition
AUPRC	Area Under Precision-Recall Curve
AVC	Audio-Visual Correspondence
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DCASE	Detection and Classification of Acoustic Scenes and Events
DCT	Discrete Cosine Transform
ESC	Environmental Sound Classification
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GPU	Graphical Processing Units
GRL	Gradient Reversal Layer
IoT	Internet of Things
MEMS	Microelectromechanical Systems
MFCC	Mel Frequency Cepstral Coefficients
MIR	Music Information Retrieval
ReLU	Rectified Linear Unit
SED	Sound Event Detection
SGD	Stochastic Gradient Descent
SONYC	Sounds of New York City
STFT	Short-time Fourier Transform
t-SNE	t-Distributed Stochastic Neighbor Embedding
VGG	Visual Geometry Group
WASN	Wireless Acoustic Sensor Nodes
WASN-ESC	Wireless Acoustic Sensor Nodes with Environmental Sound Classification
WGAN	Wasserstein Generative Adversarial Network
WHO	World Health Organization
UN	United Nations
UST	Urban Sound Tagging

Chapter 1

Introduction

This chapter looks into the important applications of audio intelligence at the edge and provide the motivation, along with the research questions that are of interest. This chapter will end off with the organization of the subsequent chapters of the thesis.

1.1 Motivation

In this day and age, it is estimated by the United Nations (UN) that 55% of the world's population lives in urban areas and is projected to reach 68% by 2050 [1]. With the projection of urbanization within societies, a growing concern amongst citizens is the effects of urban noise. Urban noise is usually composed of traffic (roads, air, and rails), construction, industrial, and social activities (parks, markets, playgrounds, eateries, etc.). The effects of noise pollution on urban residents have proven impacts on their physical health and emotional well-being [2, 3, 4]. The World Health Organization (WHO) estimates significant healthy life-years were being lost annually to environmental noise in Western Europe [5]. Health effects due to noise pollution such as cardiovascular disease, sleep disturbance, cognitive impairment, permanent hearing impairment and tinnitus have been highlighted by WHO and other studies [5, 6, 7, 8, 9].

From the perspective of the civil servant, whose work is to handle and manage noise complaints, addressing noise issues and complaints is often tedious and challenging as noise may change over time and usually occurs within a certain time frame, in terms of seconds, minutes, and hours. To help policymakers accurately and efficiently understand the environmental noise and noise types, a system that monitors and profiles noise only by their sound levels would be inadequate [10, 11, 12, 13]. There is a need for continuous monitoring, not only for sound levels but for identifying the noise types and occurrences as well.

The internet of things (IoT) is a rising and promising technology that enables smart solutions that are previously unheard of. Presently, IoT technologies have been adopted by smart cities all around the world in providing better services to their citizens. Whilst there is no formal definition of smart cities [14, 15, 16], the final aim should remain as the effective use of public resources for better communal and municipal services. In recent years, the wide adoption of IoT technologies have been notable in different fields of studies, where several IoT solutions were deployed to optimize management of utilizing public resources, such as parking [17, 18], lighting [19, 20], traffic [21, 22], smart buildings [23, 24] and waste management [25, 26]. The advent of IoT has also sprung out new research areas (such as the sounds of New York City (SONYC) project [27, 28] and the StadtLärm (CityNoise) project [29, 30]). These projects apply IoT technologies with wireless acoustic sensor nodes (WASN) that are running acoustic scene classification (ASC) [30] or sound event detection (SED) [28] predictive models for noise monitoring, detection, and classification.

By identifying the noise types, we can activate the appropriate first responders for timely support, and more importantly, to log records in databases for future reference and analysis. Subsequently, through analytics and studies, we can finally derive different approaches to mitigate noise issues.

In our research, we will focus on the environmental sound classification (ESC) aspect of the deployable WASN system. Although the idea of deploying WASN for audio related solutions have been largely adopted in this age of IoT technologies

and smart city applications, many of the WASN related studies do not incorporate and utilize ESC models and algorithms in their work [31, 32, 33]. To avoid ambiguity for the terms used in this report, the technology, study and system that incorporates ESC with WASN are referred as wireless acoustic sensor nodes with environmental sound classification (WASN-ESC).

The main challenge of WASN-ESC is the scarcity of labeled audio data along with the time-consuming manual annotations of audio data. Like many other machine learning and deep learning solutions, a well-represented and annotated dataset is essential to achieve a robust classifier. Without exception, data acquisition and annotation to generate the training, validation, and test sets are crucial to developing an effective environmental sound classifier. The scarcity of high-quality annotated labels can be concluded to be a common challenge in the majority of machine listening fields. A dedicated international challenge, as part of IEEE Audio and Acoustic Signal Processing (AASP) challenge [34], known as the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge [35] is held annually to target a compilation of different machine listening tasks, some of which, are addressing the issue of data scarcity of high-quality annotations [36, 37].

On top of the data scarcity issue, WASN-ESC introduces a new set of challenges that comprise domain adaptation complications. Environmental sounds differ at different locations due to environmental influences. An acoustic model trained with the data collected in location A would perform well in location A but may not perform well for location B. In other words, a substantial amount of well-represented audio dataset has to be collected and annotated for location B to re-train and test the acoustic model for robust performance at location B. For the human annotators to collect and annotate the audio data would require tremendous time and effort. Naturally, deploying a large-scale wireless acoustic sensor network spanning different locations would take an incredible amount of time and effort during the data collection and annotation phase. From the perspective of operationality and scalability, this would translate to a lengthy, time-consuming,

expensive, and tedious solution.

Currently, few of the more prominent publicly available datasets for ESC, ASC, and urban sound tagging (UST) are the Urban Sound dataset [38] of sounds collected in New York City, UrbanSound8k dataset [39], the ESC-50 [40] and freefield1010 dataset [41]. These datasets are tagged urban sounds acquired from freesound.org (an open-source online collaborative audio database) [42] and the Tampere University of Technology acoustic scenes dataset [43]. Other ESC and UST solutions tend to keep their dataset private as it is a vital asset to their organization [44, 45]. To highlight this research’s distinction, we focus on sounds collected in various deployed locations in Singapore. The dataset used in this research is not publicly available and was compiled by us throughout the past years. The collection and composition of the dataset used in this research will be elaborated on in Chapter 3.

Thus, in this research, we propose a domain adaptation method for the deployable WASNs to achieve robust performance in a local (Singapore) context, with the motivation to minimize the time and effort required for data annotation.

1.2 Research questions

To solve the issue of the scarcity of well-labeled data and transferring knowledge from one sensor node to the next (adaptation), several interesting research questions need to be addressed:

1. What are suitable workflows and methodologies to approach knowledge transfer for location-wise domain adaptation?
2. How much improvement, in terms of the acoustic model’s performance, would the methodologies for knowledge transfer and domain adaptation provide?
3. Can we leverage unlabelled audio data with the adoption of unsupervised learning and adaptation?

4. For repeatability, can we incorporate this proposed workflow to every new sensor node required at new locations?

1.3 Thesis Organization

This thesis shows domain adaptation for optimizing deep learning models at different locations with different microphones. The organization of this thesis is as follows. Chapter 2 summarizes the recent advancements and state-of-the-art in the field of ESC and WASN-ESC. Initially, we introduce existing work for ESC tasks and machine learning techniques such as data augmentation and transfer learning for audio applications. Subsequently, we discuss related works for real-world deployment of WASNs and describe the recent methods for domain adaptation in the audio field.

We present the dataset and methodology used in our study in Chapters 3 and 4. Firstly, we define our taxonomy for the sound classes of interest and describe the data collection and annotation process. Next, we show the data pre-processing processes on the collected audio data before training the model. Subsequently, we present the model architecture and describe the methods employed for model training and the domain adaptation task.

We discuss our experiment results in Chapter 5. Firstly, we describe the evaluation metrics used to evaluate the deep learning models and present the method used for model pruning to derive a model architecture much suited for IoT devices. Subsequently, a comparative performance evaluation is done on the results acquired with and without domain adversarial training. Next, we highlight the importance of our proposed workflow with an ablation study.

Finally, our conclusions and future work are presented in Chapter 6, with the remaining pages for the bibliography.

Chapter 2

Literature review

For the deployment of wireless acoustic sensor nodes with environmental sound classification, the technology can essentially be composed of three subtasks: ESC (on detection, prediction, and classification performed by the acoustic model running in a sensor node), WASN (on hardware and software parts of an acoustic sensor node, as well as the installation and operation of deployable sensor node) and IoT (on the interconnectability of the sensors to a central database, pushing meta-data to the database, and visual analytics of the accumulated meta-data). As our topic's focus is on domain adaptation (adapting machine learning models trained on a source data distribution to perform well on a different target data distribution) for acoustic models running in the acoustic sensor nodes (leaning towards acoustical modeling and ESC), we will not focus on the research and advancements of the IoT and WASN fields that do not incorporate ESC tasks.

2.1 Environmental sound classification

The advancement of deep neural networks, large datasets, and parallel computation in recent years have brought forth the age of deep learning, outperforming previously unattainable performance on traditional methods and shallow networks [46]. This advancement can be categorized into three waves, the perceptron algorithm [47] in 1957, the backpropagation algorithm [48] in 1986, and the success

Figure 2.1: Input and expected output of an ESC system.

of deep learning in speech recognition [49] and image classification [50] in 2012, which led to widespread attention and adoption of deep learning solutions. Complex deep model architectures with numerous parameters are trained to learn from substantial amounts of data, coupled with the recent advances in machine parallelism (such as graphical processing units (GPUs)) to accelerate the training process [51].

Chronologically, computational environmental sound analysis started later as compared to other audio fields such as automatic speech recognition (ASR) and music information retrieval (MIR); thus, numerous methods were primarily inspired and borrowed from ASR and MIR [52]. In acoustical modeling, environmental sounds generally have more variations, in contrast to ASR and MIR, where music and speech can be considered as sound classes of environmental sounds. To capture the intra-class variations for the sound classes in the field of ESC, a wide range of intra-class variation is required to achieve satisfactory results. In addition, variations in the frequency response of different microphones, the microphone’s distance to the audio source, reverberation, and the different designs of microphones, ESC can be considered a much more demanding task than ASR and MIR. Computational environmental sound analysis can be further categorized into environmental sound classification, sound tagging and sound event detection

[52]. In a classification task, only one label is predicted for one segment of audio input in a classification task, as shown in Figure 2.1. An example of classification would be as such: given 4 seconds of audio input, predict one label among a set of pre-defined labels. In sound tagging, several labels are predicted for one segment of sound. For example, given a 10-second audio input, predict if the segment includes one or more of the pre-defined labels. In a sound event detection task, given one segment of audio input, we detect the onset and offset of each sound event, as well as its label. As the classification, tagging, and detection tasks utilize deep learning algorithms and modeling approach, we would like to take the opportunity here to define the term deep learning. Deep learning is a field dedicated to the study and development of machines that can learn underlying feature representations of datasets with the use of deep artificial neural networks. Deep learning is a subset of methods for machine learning [46].

Thus, deep learning, with the ability to learn feature representations given the data, became an attractive solution in the field of ESC, which we will further describe in the subsequent subsections.

2.1.1 Overview of ESC with machine learning approach

The common approach for ESC tasks is based on supervised learning [53]. In supervised learning, each example in the dataset is labeled with the ground truth or the target value. The benefit of having these ground truth labels is that the machine learning algorithm can use them to optimize the learning process. It does this by comparing the target outputs specified in the dataset with its predictions, and computes the difference (or error) and uses the backpropagation algorithm to update its weights iteratively to minimize this error. It is because of this feedback from the target labels in the dataset to the algorithm that this type of machine learning is considered supervised. Figure 2.2 shows a generic overview workflow for training an acoustic model for ESC task. The common workflow essentially leverages on the advancements of the image processing field, by converting the one-dimensional time-series signal of raw audio samples into an image-like two-

Figure 2.2: Overview workflow for training an acoustic model with machine learning.

dimensional time-frequency representation to be used as data samples¹ for model training. When the entire training data samples have been passed into the model, it would be marked as one epoch. Typically, at the end of an epoch, the validation set is passed into the model to gauge the model's performance on an unseen (not trained) validating dataset. This is a common practice in machine learning for monitoring the training progress whilst promoting model generalization and preventing model overfitting. The entire training process repeats itself until one of the few declared conditions are met (e.g. the number of epochs have reached its predefined number, the training loss saturates, the performance of the model on the validation set saturates). Usually, the best performing model on the validation

¹data samples mentioned here are not to be confused with audio samples, but rather samples/examples that form the data set for statistical analysis and machine learning. To avoid any ambiguity, data samples mentioned in this report should be thought of as in statistics discipline rather than audio signal processing

Figure 2.3: Usage of the trained acoustic model for deployment.

set is chosen and exported as the deployable acoustic model. A trained model is then exported and uploaded to the processing unit for inference. At deployment, the process bypasses the training loop and ends by computing an output prediction given an audio segment. Figure 2.3 shows the workflow of the usage of a deployed acoustic model.

2.1.2 Audio features and preprocessing of audio data

Audio applications started experimentations with deep learning methods from the advancements in the image field by representing audio signals as images [54]. By converting one-dimensional time-series audio signals into two-dimensional time-frequency representations with short-time Fourier transform (STFT), we can transform audio signals into image-like data examples. The audio signals are usually divided into overlapping frames between 20 and 60 ms and smoothed using a Hann or Hamming window. Transforming audio frames into the frequency domain is implemented efficiently using the fast Fourier transform algorithm, and finally plotting the changing spectra as a function of time, known as a spectrogram.

Before the deep learning era, hand-crafted features are often coupled with traditional machine learning methods and models such as Gaussian mixture models and hidden Markov models for classification tasks [55, 56, 57]. However, these methods have been outperformed by deep network models in applications where sufficient data is available [58, 59]. One of the more popular hand-crafted features in audio-related fields is the Mel frequency cepstral coefficients (MFCCs)

Figure 2.4: Time domain audio signals transformed into magnitude spectra and projected to a reduced set of Mel frequency bands and converted to logarithmic magnitudes.

[60], where it has previously achieved success in the field of ASR. For decades, MFCCs have been the dominant acoustic feature representation for many audio analysis tasks. However, since MFCCs were initially developed specifically for speech signals, they are not exactly the best tool to extract feature representations that are significantly different from speech. In the era of deep learning, studies in environmental sound moved away from MFCCs and use LogMel spectrogram instead [54, 61]. The compression of the discrete cosine transform (DCT) in MFCCs for environmental sounds was shown to be unnecessary; thus, omitting the DCT compression out of the MFCCs leaves us with the LogMel spectrum - a popular feature across environmental sound fields in deep learning [52, 62]. Since the projection of frequencies of the Mel filter bank is greatly inspired by the human auditory system [60], Mel spectrograms are designed to capture perceptually relevant information more efficiently with fewer frequency bands compared to linear spectrograms. Therefore, it is widespread in the audio machine learning

Figure 2.5: Downstream workflow for ESC with a convolutional neural network (CNN) architecture

literature for environmental sounds to use Mel-frequency Log-magnitude spectrograms as input to CNNs. Figure 2.4 shows the transformation of audio signals from a one-dimensional time-domain into a two-dimensional time-frequency spectrogram, mapped by Mel frequency bands of logarithmic magnitudes. Popular number of Mel bands are 64, 96, and 128 bands [62, 63, 64]. Figure 2.4 shows a LogMel spectrogram with 128 Mel bands.

2.1.3 Deep learning neural network model architectures

Like many other deep learning solutions for audio, convolutional, recurrent and multiple fully-connected feedforward layers are usually stacked to increase the modeling capability. A deep neural network is essentially a neural network with many stacked layers [46]. CNNs are the earliest successful neural networks that were applied to the field of ESC that achieved good results [54, 61, 65, 66, 67, 68]. CNNs are based on convolving the input image with learnable filters (or kernels). A convolutional layer computes multiple feature maps, known as channels or filters, from its corresponding filter. Pooling layers are applied over convolutional layers to downsample the feature maps, where downsampling operations have shown positive results for capturing invariances in image-like data [69]. A CNN often consists of a series of convolutional layers coupled with activation functions and pooling layers. The final pooling layer is often flattened as input to a fully-connected network. The output of the fully-connected network is a vector corresponding to the predictions for each class. Figure 2.5 shows a typical downstream workflow for ESC using a CNN architecture. The majority of the

Figure 2.6: Downstream workflow for ESC with a convolutional-recurrent neural network (CRNN) architecture

CNN models used in ESC tasks are borrowed from the image processing/computer vision literature. Few of the successful and popular CNN architectures in deep learning are the visual geometry group (VGG) architecture [70], the Inception architecture [71], and the ResNet architecture [72]. An extensive comparison of the different popular CNN architectures that achieved good performance for large-scale audio classification task was reviewed in [73]. The Inception architecture is able to achieve better performance for audio classification tasks when compared with other model architectures (such as the VGG and ResNet architectures) with faster training, therefore, cutting down the time and resources required for model training [73]. In general, the architecture of a CNN is largely chosen experimentally based on a validation score, which leads to some general rule-of-thumb and practices in the deep learning field such as using fewer parameters for fewer data and increasing the number of filters while decreasing its size in subsequent convolutional layers [46, 74].

Generally, ESC tasks use CNN based network architectures to classify environmental sounds, resulting in multi-class, single-label prediction. In contrast, SED tasks commonly use CRNN as they focus on a precise detection (on-set and off-set) of overlapping sound events resulting in multi-class, multi-label predictions. Figure 2.6 shows the downstream overview of a CRNN architecture.

2.1.4 Data augmentation techniques

In deep learning and training deep networks, capturing the natural variability within the dataset is often associated with better model performance. The ability to capture the natural variability of the dataset translates to requiring large amounts of training data in order to capture both the inter-class and intra-class variations. Even though the size of machine listening datasets has increased over the last few years, the audio domain lagged behind computer vision datasets such as the ImageNet dataset with 14 million images over 21 thousand classes [50]. Presently, the only machine listening dataset that came close is the AudioSet dataset [75], with over 2.1 million audio excerpts over 527 sound classes. This disparity led deep learning practitioners in the audio field to explore ways in augmenting the data to generate more training data and, at the same time, introduce variations to their dataset.

The simplest and widely practiced augmentation techniques are the ones that augment the existing available data by applying various signal transformations. Basic audio signal augmentation includes time stretching, pitch shifting, dynamic range compression, and adding random noise [61]. Other data augmentation methods focus on the domain-transformed features (e.g. spectrograms) instead of the signals. For example, mixup data augmentation creates new training instances by mixing pairs of features and their corresponding targets on a given mixing ratio [76]. Methods that distort the features, such as random erase, replace random boxes in feature representations by random numbers [77] and SpecAugment, where blocks of features are randomly masked [78].

2.1.5 Transfer learning

The transfer learning approach has seen its popularity increase recently as a solution, allowing the community to use small labeled datasets to train on shallow networks by leveraging pre-trained networks [46, 79]. These pre-trained models are typically trained on a large number of data samples and are utilized for a downstream machine learning task, ideally, on the data classes that are somewhat

related to the features we want to extract [46, 79]. The two ways to apply transfer learning with a pre-trained model are fine-tuning the pre-trained model or by using the pre-trained model as a feature extractor. For fine-tuning, we take the pre-trained model and train it on a task-specific dataset, optimizing the underlying model weights. Hence, instead of randomly initializing the model’s weights and training the model from scratch, fine-tuning initializes the model weights with the pre-trained weights [46, 80]. When utilizing a pre-trained model as a feature extractor, we do not optimize/change any of the pre-trained weights. Instead, we use the network to extract the salient information of the input data into a low-dimensional feature representation of the input data [46, 80].

Over recent years, the transfer learning approach has been explored in many machine listening tasks as well. Transfer learning using deep audio embeddings extracted from pre-trained models can get to a workable solution with limited data [81, 73, 82, 64]. One of the first and better known pre-trained model in using transfer learning for machine listening tasks is the SoundNet [81]. The SoundNet model used a student-teacher training procedure to teach the network for sound by using visual networks on over 2 million videos from Flickr [83]. Results showed that the SoundNet method outperformed handcrafted features methods over the ESC dataset [40]. Another feature extractor that extracts what is known as the VGGish embeddings [73] employs a pre-trained model trained with the YouTube-100M [73, 84] dataset on the ResNet-50 architecture [72]. The VGGish embeddings were compared to Log-Mel patches for comparative performance. Both of the features were trained and evaluated on a fully-connected network and concluded that the VGGish embeddings outperformed the Log-Mel patches. L^3 -Net [82], the successor of SoundNet, was trained with the same dataset as SoundNet but employed a method that differed from SoundNet. L^3 -Net employed 2 subnetworks, an audio subnetwork and a visual subnetwork, where both of the networks were concatenated into a fusion network which outputs a value whether the 2 inputs (image and sound into their respective networks) corresponds to each other. After training, the audio subnetwork was extracted and used to evaluate the same

datasets SoundNet was evaluated on. The results showed that the L^3 -Net embeddings outperformed SoundNet’s performance. Open L^3 [64] is one of the latest works in deploying the transfer learning method on machine listening tasks. The Open L^3 used the same method as L^3 -Net, known as the audio-visual correspondence (AVC) method, but instead, the Open L^3 was trained with the AudioSet [75] dataset. The Open L^3 embeddings did a comparative performance against SoundNet embeddings and VGGish embeddings, where the former successfully outperformed both the latter [64]. Although many transfer learning methods have shown remarkable results in their comparative performance, it is usually compared to hand-crafted features or Log-Mel features trained on a lightweight fully-connected network.

2.1.6 Challenges of transfer learning

The adoption of transfer learning is practical in any machine learning task when the target domain has few or no labeled data. As discussed previously, the resources required for annotating well represented data is often expensive and time-consuming. However, the effectiveness of transfer learning is not always guaranteed and transfer learning tasks comprise a set of its challenges. Transfer learning works best if the source and the target domain data are similar or if the training and test data are drawn from a similar distribution. Unfortunately, in practical scenarios, it is common that the training and test data experience a covariate shift and are not from the same distribution. The covariate shift issue, also known as domain shift, between the source and target domain give rise to a common challenge in transfer learning known as negative transfer [85, 86] - where the newly trained model (after transfer learning) ends up with a decrease in performance. We will explore domain shift in the subsequent section in greater detail and discuss the challenges in real-world deployment/scenarios. Finally, we will review recent studies in domain adaptation for audio classification tasks and the methods used to address the domain shift issue.

Figure 2.7: Flowchart of a typical workflow to train deep learning models for WASN-ESC tasks

2.2 Real-world deployment

The recent advances in IoT technologies resulted in smaller and cheaper sensors and embedded processors with powerful computational capabilities. Solutions, which were previously difficult to achieve using small form factor and vast deployments, have now been made possible through a combination of IoT and cloud technologies. Many real-world deployed WASN-ESC tasks and studies address acoustic noise problems in urban settings, with interest in identifying the occurrence of a significant sound event and its frequency, the type of sound event, sound pressure level.

This section will review WASN made up of IoT devices, distributed in different locations to perform intelligent audio sensing and the challenges and difficulties it may present.

2.2.1 Environmental sound classification with wireless acoustic sensor nodes

WASN, as the name suggests, are wireless acoustic devices that are usually distributed in various prominent locations to form a network of sensors for data aggregation, visualization, and reporting. In an ideal case of requirements, each acoustic sensing node has to capture sound signal continuously and immediately publish the noise type, sound pressure level, direction-of-arrival of the noise, and the timestamp of the events, and transmit this set of information as metadata. Since only the metadata of detected sounds is uploaded, no sound signal will be uploaded from the WASN, mitigating privacy issues. These requirements, however, differ from projects to projects in regards to different published studies.

The Sounds of New York City (SONYC) project [27, 28], which started in 2013, uses an acoustic sensor constructed using the Raspberry Pi [87] and a customized microelectromechanical system (MEMS) microphone module. A total of 56 sensor nodes are deployed in Greenwich Village, Manhattan, Brooklyn, and Queens as of December 2018. The SONYC project, however, does not include the direction of arrival (DoA). The project focuses on SED tasks, leveraging the Raspberry Pi to run a deep learning model in a WASN set up to detect and classify environmental sounds at the edge. Their model was trained and evaluated using a customized dataset, UrbanSound8K, with a defined sound taxonomy [39]. The UrbanSound8K dataset was constructed using recordings obtained from their sensors combined with audio recording in urban settings downloaded from Freesound [42].

Another project, known as DYNAMAP [88], developed a real-time system to detect and visualize the acoustic impact of road infrastructure using smart sensors. A two-class sound classifier based on the Gaussian mixture model (GMM) used to determine the noise contribution from roads and junctions while ignoring noise events unrelated to traffic noise [89]. There are two pilot areas located in Milan and Rome, where 24 and 19 sensors were installed, respectively. Dynamic noise maps of these areas are generated in real-time, and sound classification of noise

Table 2.1: Overview of the public datasets for acoustic scenes and sound events.

Dataset	No. of Sound Classes	Total duration	Location type
UrbanSound8K [65]	10	approx. 10 hours	Outdoors
TUT Acoustic scenes 2016 [43]	15	13 hours	Indoors & Outdoors
freefield 1010 [41]	7	approx. 21 hours	Outdoors

sources is performed to determine the noise contribution from roads and junctions [88, 89].

The Stadtlärm [29, 30] project focuses on acoustic measurements and sound classification within and around a park along a river located in Jena, Germany. The sound classification is performed by CNN model of the VGG architecture. A total of nine sound classes were selected for this study, and the training set was assembled from the dataset UrbanSound8K [65], Tampere University of Technology database for acoustic scenes 2016 dataset [43], and freefield 1010 dataset [41] as summarized in Table 2.1. The location of this study is surrounded by two streets that connect the inner city to nearby highways, trams, and trains. The research also included sound activities from nearby restaurants and regular open-air music events.

The main differences amongst these WASN projects include: the deployed locations, the number of deployed nodes, and the method used for data collection and training. The sensor network in the DYNAMAP project was deployed over a large area but focused on road noise. On the other hand, the sensor network of the StadtLärm project was deployed within the vicinity of a park and investigated more sound classes. The SONYC and StadtLärm projects leveraged external sound databases to speed up the annotation process of their dataset. One thing in common amongst the aforementioned projects is the application of deep learning models for ESC tasks in a WASN setup. Figure 2.7 shows the typical workflow to train these deep learning models to perform WASN-ESC tasks.

2.2.2 Challenges of real-world deployment

The sensor network’s availability and serviceability are largely dependent on the sensors, which can be increasingly challenging to manage as the sensor network increases in size. A large-scale sensor network translates to challenges in real-world deployment for hardware/software maintenance and the ability to preempt failures. On top of that, the increase in sensor network size (number of physical sensor nodes deployed) presents another issue for the underlying deep learning model in the devices as well. Many of the previously discussed WASN-ESC projects used established open-source datasets to train their deep learning models but did not address how the performance of the models are affected in different deployed locations. Environmental sounds in other deployed sites may be very different. Different environmental influences at other locations also mean that the deployed models in new areas will be prone to and possibly suffer from covariate shift, i.e., a distribution mismatch between the training datasets and test sets (collected and representing the data at the new location). That said, an ideal WASN-ESC should be able to analyze the system performance for multiple sound classes over a large area of deployment. However, disparities in the captured sounds due to different locations, microphones and WASN setups would introduce a mismatch between these domains.

2.2.3 Domain adaptation

As discussed in the supersection, the model’s performance is often susceptible to covariate shift, i.e., a distribution mismatch between training and test datasets. Data samples are used to train the classification models to make predictions for future samples. However, if the incoming predicted samples are not an accurate representation of the training samples’ underlying distribution, the model will not generalize well. When deploying in real-world application scenarios, WASN-ESC systems usually face novel acoustic conditions caused by different recording devices or environmental influences, known as a domain shift [90]. Hence, domain adaption methods were derived to improve the robustness of classification

networks in such scenarios by adapting to data from a target domain [91].

Domain adaptation methods can be categorized into supervised, unsupervised and semi-supervised methods, all of which are defined by the availability of labels for the target domain data. Supervised domain adaptation usually involves fine-tuning a model on new target domain data after pre-trained on the annotated source domain data. Semi-supervised domain adaptation would require few labeled data to perform adaptation. On the other hand, unsupervised domain adaptation methods employ samples without labels to achieve the adaptation task.

Domain adaptation methods are prevalent in image-specific tasks. However, many of the highly cited methods [92, 93, 94, 95] did not apply to datasets in real-world scenarios and are instead used on predefined source-to-target domain datasets such as MNIST [96] to MNIST-M [92] dataset, MNIST to SVN [97] dataset, MNIST to USPS [98] dataset, or other variants of the MNIST dataset.

Studies in audio-specific domain adaptation for different recording devices experimented with various approaches. A spectrum correction method was applied to compensate for different frequency responses of the recording equipment, where estimations on a set of frequency-dependent magnitude coefficients were derived from the source domain data and used for spectrogram equalization of the target domain data [99]. This approach led to the best classification results in Task 1B, “Acoustic Scene Classification with mismatched recording devices”, of the 2019 DCASE Challenge [100]; however, it required time-aligned audio recordings from several microphones during the training phase.

Several audio-related domain adaptation studies explored methods employing adversarial networks [91, 101]. Adversarial training with generative adversarial networks (GANs) [102] was explored by using intermediate feature mappings of an ASC model to follow a similar distribution for both the source and target domain data and was further improved using the Wasserstein generative adversarial networks (WGAN) method [91]. A comparative study [101] has shown that their proposed method in using an unsupervised generative network demonstrated improved target domain results.

2.3 Conclusion

Although many studies were published on the topic of domain adaptation for ASC tasks, it is mainly targeted to address issues caused by different recording devices. Domain adaptation studies in most of the machine listening tasks did not address the issue in domain adaptation for different locations caused by covariate shift due to location-wise environmental influences.

In this research, we will explore methods to address the issue of domain adaptation in WASN-ESC tasks. In particular, the covariate shift introduced by environmental influences due to different deployed locations in Singapore.

Chapter 3

Dataset

Constructing a dataset is essential to any machine learning problem. In this thesis, our first challenge is the lack of well-represented audio data to train the sound classifier. Therefore, we started collecting and annotating audio data at different locations of our deployed sensor nodes, although it did require a considerable amount of time. This chapter encompasses the organization of our dataset. Firstly, the taxonomy for the sound classes of interest is defined. The taxonomy is of great importance in this work as it serves as a prerequisite for data collection, data annotation, and subsequently how well the model is able to perform in relation to the defined sound classes.

Following the taxonomy, we present the processes for data collection and annotation. Next, we define the pre-processing steps to transform the collected and annotated audio data into data samples to fit the model. Finally, we split the data samples into training, validation and test sets for model training, validating and testing. The following subsections present the taxonomy, data collection, data annotation, and data pre-processing in detail.

Table 3.1: Taxonomy of sound classifier

Category	Sound classes	Description
Machinery sound	Vehicle	Sounds from the engines of vehicles, such as motorcycles, cars, buses, lorries, and aircraft. Vehicles aside, this class also includes engine/motor sounds made by non-mobile machines, such as fruits shredder machines, pressure cleaner motor (motor pumps), leaf blower, and screeching brake sounds produced by vehicles.
	Construction	Non-engine sounds produced from construction machinery, such as jackhammers, breakers, and powered saws.
	Car horn	The (air) horn or klaxon of a vehicle, such as a car, van, bus, or lorry.
	Alarm	Any kind of alert signal, such as fire alarms, sirens, car alarms, and alarm clocks.
Human-generated sound	Impact sound	Any transient impact sounds, as well as other loud impact-like sounds which may not be transient such as opening/closing of shutter (shops), trolley rolling wheels and dragging of chairs/tables.
	Music	Sounds originating from live bands, buskers, or recorded music played back through a speaker.
	Shout	Sound of a person or people shouting or screaming.
	Human voice	Sound of a person or people talking. This class also includes coughing and sneezing.
Environmental sounds	Rain	Sound due to precipitation from the sky, such as drizzle or thunderstorm. Also encompasses water splashing, pouring, flushing, spraying sounds.
	Bird	Vocalization by a bird.
	Ambient	Any sound not belonging to any classes. Some of the common sounds found in this class include: dog barking, wind sound, squeaking baby shoes, sweeping floor, footsteps, etc.

3.1 Taxonomy

For our application, there are 11 unique urban sound classes - 10 unique sound classes of interest and an “Ambient” class that encompasses any sound events that are not of interest and cannot be classified into any of the ten classes.

Figure 3.1: The deployed wireless acoustic sensor nodes. From left to right: Location L1, L2 and L3 with microphones M1 [104] at locations L1 and L2 and microphone M2 [105] at location L3.

We follow the taxonomy of our previous work in [103]. These sound classes can be grouped into three categories: machinery sound, human-generated sound, and environmental background sounds, as shown in Table 3.1.

3.2 Data collection

Our WASNs are deployed in different locations near public residential facilities. These deployment locations are chosen based on feedback from local agencies and the public. Table 3.2 summarizes these locations, along with a brief description of the surroundings as well as the common sounds observed at each location. A total of 24 hours, 5 minutes and 1 second of audio data are collected from three sensor nodes deployed in three different locations.

Following Table 3.2, the data samples in the training set are recordings collected from two locations within a 5-meter radius vicinity (locations L1 and L2). The data samples in the validation set are recordings collected from locations L1 and L2 as well. To provide a baseline study for location-wise domain adaptation, we construct the test set to consist of recordings solely from location L3. Both sensor nodes deployed in locations L1 and L2 are in the same residential estate and are approximately 20 meters apart from one another. The sensor node

deployed in location L1 is near a children’s playground, while the sensor node deployed in location L2 is next to a basketball court. Both locations L1 and L2 are separated by an open-space residential car parking space. To establish significant contrast between the source (locations L1 and L2) and target (location L3) domains, the sensor node deployed in location L3 is in another residential estate (apart from locations L1 and L2) at an open-space pavilion located at the rooftop of a residential housing building.

Apart from being in a different location, the target domain data, represented as the test set, is also collected with a different recording device. Table 3.3 summarizes the two different recording devices, denoted as M1 and M2 - where microphone M1, a single channel micro-electromechanical system (MEMS) microphone [104], is employed for data collection at locations L1 and L2, and microphone M2, a 7-channel circular array MEMS microphone [105], is utilized to collect audio data recordings from location L3. Only one channel of the 7-channel array recording device (microphone 0 - center mic, see Figure 3.2) is used to record and construct the test set. The locations (L1, L2 & L3) and microphones (M1 & M2) are summarized in Table 3.4 as part of the training, validation, and test set. Figure 3.1 shows the deployed WASNs in locations L1, L2 and L3 with microphones M1 at locations L1 and L2, and microphone M2 at location L3. The data collected from the separately located sensor nodes coupled with microphones M1 (at location L1 and L2) and M2 (at location L3) are synchronized with the system clock of the sensor nodes in accordance with the setup of our previous work in [103].

The deployed WASN in residential estates are line-powered from the switch rooms of the estate. Each individual sensor node comprises a Raspberry Pi 3 [87] coupled with a recording device (M1 or M2) shown in Table 3.3. The sensor nodes are not designed specifically for low power consumption and has an average power consumption of approximately 6W per node.

Figure 3.2: Microphone M2 [105] - MEMS microphones on circular array aligned to Cartesian coordinate system. At the data collection stage, only Mic 0 (center microphone) is utilized for recording.

Table 3.2: Locations, descriptions, and common sounds observed of the deployed sensor nodes

Location	Description	Common sounds observed
L1- Playground	Children's playground within residential estate. Next to an open-space residential carpark.	Human speech, shouting and screaming (mostly children), vehicle engine sounds, birds chirping, car alarm beeping, car horn, aircraft sound flying by.
L2- Basketball court	Basketball court, next to an open-space residential carpark.	Basketball bouncing, human speech, shouting and screaming, vehicle engine sounds, reversing car alarm beeping, birds chirping.
L3- Rooftop pavilion	Open-space pavilion located at rooftop of a residential housing building in a residential estate.	Human speech and shouting, birds chirping, impact sounds, recorded music playing from speakers, aircraft sounds (mainly military aircrafts such as jet fighters and helicopters).

Table 3.3: Recording devices of our WASN with corresponding descriptions

Recording device	Description
M1- [104]	Single channel unit connected to the embedded processor through I2S connection
M2- [105]	7-channel array connected to the embedded processor through USB connection

Figure 3.3: Data annotation process on collected audio recordings - the topmost audio waveform corresponds to the collected raw audio data. The audio waveforms in the middle row are snippets of the topmost audio waveform, edited (trimmed and stitched according to the same class type) and labeled. The remaining audio recordings that do not encompass any sound class events are concatenated and labeled as “Ambient”.

3.3 Data annotation process

The collected audio data are annotated according to our taxonomy (Table 3.1) to provide ground-truth labels. Ground-truth labels are essential for supervised learning tasks and evaluating the performance of deep models.

The annotation process is illustrated in Figure 3.3. The collected raw audio data is imported into the audio editing software. We use version 3.0.0 of Audacity® recording and editing software [106] for audio playback and audio trimming. The annotator listens through the collected raw audio data and trims out sound events with reference to our taxonomy. We stitch the trimmed audio data of the

Figure 3.4: Method for generating data samples from annotated recordings

same class type into a longer audio file and export it with the corresponding labeled name. The remaining audio data without sound events are stitched as well and is labeled as “Ambient”. At the end of the annotation process, we arrive with sizeable audio files that are representative of the 11 classes.

3.4 Data pre-processing and preparation for experiment setup

This section discusses the setup, pre-processing and preparation of the collected and annotated data for model training and evaluation.

Data in the training, validation, and test sets are mutually exclusive. To generate each data sample in the training, validation, and test sets, each annotated recording is divided into frames of length 4 seconds with an overlap of 2 seconds between frames, shown in Figure 3.4. No audio padding nor augmentation techniques are applied to frames containing less than 4 seconds of audio, and these frames are discarded.

The raw audio data is transformed into Log-Mel spectrograms as input to an InceptionNet-based network for training, validating, and testing. The microphone input signals with sampling rate of 44.1kHz are transformed into the short-time Fourier transform domain. A Hann window with a window length of 1,024 samples (23.2-ms) and 50% overlap (512 samples, 11.6-ms) is used in the computation of

Table 3.4: Class-wise audio length and number of samples in train/val/test splits

Class	Training Set (L1 & L2) (M1)		Validation Set (L1 & L2) (M1)		Test Set (L3) (M2)	
	Data samples	Total length (HH:MM:SS)	Data samples	Total length (HH:MM:SS)	Data samples	Total length (HH:MM:SS)
00-Ambient	6,013	03:21:00	111	00:03:44	583	00:19:41
01-Construction	3,764	02:05:41	276	00:09:23	205	00:06:52
02-Vehicle	2,822	01:34:15	661	00:22:28	332	00:11:12
03-Music	5,610	03:07:10	105	00:03:36	315	00:10:36
04-Shout	1,882	01:02:50	306	00:10:23	202	00:06:51
05-Rain	3,273	01:49:14	453	00:15:14	274	00:09:11
06-Birds	4,584	02:33:01	103	00:03:32	284	00:09:33
07-CarHorn	1,120	00:37:27	127	00:04:25	56	00:01:54
08-Alarm	2,926	01:37:42	240	00:08:11	51	00:01:45
09-ImpactSound	3,097	01:43:21	255	00:08:46	113	00:03:55
10-HumanVoice	2,445	01:21:39	114	00:03:53	495	00:16:36

each STFT with 2048 FFT points. The computation results in 343 time-frames for every 4-second segment of audio clip. Hence, each 4-second segment of single-channel raw audio in each data sample is converted to a Log-Mel spectrogram of dimension (343, 64, 1). 128 time-frames (approx. 1.5-second) are randomly cropped within the 343-point spectrogram to form a shape of (128, 64, 1), where 128 represents the number of time-frames in the STFT, 64 represents the number of filters in the Mel filter bank, and 1 represents the number of image channels. Finally, the data samples of shape (128, 64, 1) are fed into the model for training, validating and testing.

To speed up the training process, all incoming data samples are normalized before feeding into the model as input. The Log-Mel spectrograms are normalized with the mean standard deviation standardization method $\bar{x} = \frac{x-\mu}{\sigma}$ where x is the input, μ is the mean and σ is the standard deviation, and $\bar{x}$ is the normalized data sample. The mean and standard deviation values are calculated with all data samples of the training dataset.

Table 3.4 summarizes the number of data samples in the training, validation, and test sets, as well as their total length. The column “Data samples” in Table 3.4 refers to the total number of transformed LogMel spectrogram samples for each class. The term “Total length” in Table 3.4 refers to the total raw audio length (in terms of hours, minutes, and seconds) per class before processing into LogMel spectrograms. The dataset is imbalanced due to the natural difference in intra-class variation of each sound class in our taxonomy and the varying rates of these occurrences at the deployment sites. A class weighted random sampler is added to the data loader to address the imbalanced dataset issue. A data loader is a predefined data loading utility that facilitates loading the training samples into the model during model training. The class weighted random sampler rebalances our training data classes by oversampling the minor classes during model training. Although oversampling increases the likelihood of overfitting classes with lower data sample count due to making exact copies of existing examples during model training, it utilizes all the available annotated data, when compared to undersampling classes with higher data sample count. By utilizing all available collected and annotated data, our training dataset encompasses better representations for intra-class variations that are reflected in the pre-defined taxonomy. Therefore, oversampling is chosen to address the imbalance dataset when representations of intra-class variations are considered. As a result, our model will not see significantly more for one class over another during model training, and therefore, reduces the risk of overfitting to classes with a higher data sample count.

3.5 Basis of pre-processing parameters

Our proposed system performs environmental sound classification, where the data annotation task is solely to provide the class labels. Unlike sound event detection tasks, the onset and offset points of a sound event were not annotated nor used for our model training. At deployment, the model is predicting in real-time given the incoming buffer. Therefore, our model must be familiar with all the onset, midst and offset of the sound event to predict the sound event given the LogMel spectrogram converted incoming audio buffer. Accordingly, data samples (segments from the annotated data) used for model training should contain examples representing the onset, midst and offset of the sound event and the temporal changes over time to achieve robust performance at real-time inference when deployed.

The time interval of the data sample is optimized for the input size required by the classification model, and the preprocessing parameters for audio segment length and overlapping audio segments chosen are primarily dependent on the model’s input shape. For our classification model training, samples of shape 128 by 64 by 1 are randomly extracted from a LogMel spectrogram converted from an audio clip. Therefore, each audio clip converted LogMel spectrogram should be sufficiently greater than 128 time-frames of the LogMel spectrogram (longer than 1.5 seconds) because the input shape of our classifier is set as 128 by 64 by 1 (128 time-frames, 64 Mel-bands). We chose the best-fit parameters of 4 seconds with 2 seconds overlap to generate adequate data samples for model training and diversify the data representation at different temporal points for our classification model with input shape 128 by 64 by 1. We illustrate the basis of our choice with three cases shown in Figure 3.5.

(a) Case 1: 10s segment, 2s overlap - 20s audio length utilized for 2x (128, 64, 1) training samples - Discarding substantial amount of audio segments for model training.

(b) Case 2: 2s segment, 1s overlap - Multiple duplicated segments in training samples.

(c) Case 3: 4s segment, 2s overlap - 8s audio length utilized for 3x (128, 64, 1) training samples.

Figure 3.5: Pre-processing parameters illustrated with 3 unique cases.

In case 1 (Figure 3.5a), a segment size of 128 time-frames for model input is randomly taken per 10-second audio clip. Consequently, selecting the audio length at 10 seconds with 2 seconds overlap would significantly lose crucial data representations because we are discarding the remaining time-frames when we crop 128 time-frames out of a 10-second (857 time-frames) LogMel spectrogram for model input. Selecting a shorter pre-processing parameter of 2 seconds with 1-second overlap (case 2, Figure 3.5b) will result in multiple duplicated segments in our dataset for training the model because we are cropping 128 time-frames out of the 2-second audio clip (171 time-frames) that has a 1-second (85 time-frames) overlap of the previous segment. Thus, by selecting a time interval of 4 seconds with 2-second overlap (case 3, Figure 3.5c), we are able to maximize the number of critical representations from the annotated audio data while retaining a good compromise between discarding valuable data representations and minimizing duplicate or overlapping inputs to the classification model.

3.6 Conclusion

The collected audio data are recorded with two unique microphones in three deployed locations. Annotations on class labels/ground truths for the collected audio data are according to our defined taxonomy. The collected and annotated datasets are pre-processed and split into mutually exclusive training, validation and test sets.

Chapter 4

Methodology

We address the domain shift issue introduced by environmental influences and non-identical hardware responses at different locations with different microphones. We also have to ensure that the sound classifier has an architecture that is lightweight and is capable of accurate predictions - considering that the classifier is often running concurrently with other software components on an IoT device.

This chapter presents the methodology used in this work on domain adaptation for ESC in a WASN setup.

4.1 Model Architecture

In this section, we propose an adversarial domain adaptation model. The model comprises three parts, a feature extractor, a classifier and a discriminator for adversarial training. The InceptionNet is chosen as the model architecture for our feature extractor according to the literature review of the thesis. Both the classifier and discriminator networks are fully-connected linear layers. The training method used is an unsupervised domain adaptation method which requires no labeled data from the target domain to train the model. We follow the general framework proposed in [92]. Figure 4.1 shows an overview of the domain adaptation model.

During the adversarial training process, the feature extractor's flattened out-

Figure 4.1: Overview of the domain adaptation model - incorporates a feature extractor, classifier and discriminator.

put is fed to two fully-connected networks, a classifier and a discriminator, as shown in Figure 4.1. In contrast, the downstream ESC task would only comprise both the feature extractor and the classifier. Given a data sample, the discriminator’s role is to discriminate between the domains, whereas the classifier’s role is to provide a sound class prediction. The gradient reversal layer is inserted between the discriminator and the feature extractor to introduce domain confusion within the feature extractor. During backpropagation, the gradient reversal layer inverses the gradient backpropagated from the discriminator (domain classifier), and impairs the feature extractor’s ability to discriminate between the domains. Domain confusion with gradient reversal causes the feature distributions to be much similar, resulting in domain-invariant features that can be extracted by the adversarial-trained feature extractor network. Hence, the downstream ESC task only requires the adversarial-trained feature extractor and the classifier network, given data samples from the different domains. The training methods are described in greater detail in the next section, Section 4.2.

4.1.1 Feature extractor

The feature extractor consists of the most trainable parameters in the entire model. Given a data sample, a trained feature extractor outputs extracted features (also known as embeddings or feature embeddings) - a much more compact representation of the data sample. The extracted features are then fed into a fully-connected network (classifier) for the prediction output.

Model architectures such as ResNet [72], InceptionNet [71], and VGG16 [70] having millions of parameters are computationally expensive to run on our WASN since computational cost is also incurred by other software components. We use a CNN architecture for our feature extractor to minimize the required computation for classification without the inclusion of the recurrent module in the network. A model pruning study is also conducted and discussed in the next chapter; to keep the model parameters minimal and reduce the required computational power whilst providing accurate predictions.

The proposed model architecture of the feature extractor used in this study is a shallower variant of the InceptionNet [71] - where the original number of filters for the InceptionNet is reduced by a factor of 4 (at a reduction of 93.66% in terms of number of model parameters). An analysis is conducted and discussed for the factor used for parameter pruning in Section 5.2.

4.1.2 Inception network architecture

Our proposed feature extractor network retains the depth but reduces the number of filters used in each convolutional block of the original InceptionNet architecture. The proposed variant Inception network consists of a 93.66% reduction in the number of parameters relative to the original InceptionNet architecture, as elaborated in the model pruning section (Section 5.2) of the thesis. In this section, we describe the details of our feature extractor network.

The main body of an inception network comprises a succession of inception blocks. The construct of an inception block utilizes multiple convolutional blocks with different kernel sizes and returns a concatenated output of the convolutional blocks. Each convolutional block consists of a convolutional layer, followed by batch normalization [107] and the rectified linear unit (ReLU) activation function, as shown in Figure 4.2.

As illustrated in Figure 4.3, the inception block takes the output from the previous layer as input and forwards it into a maxpooling layer and a convolutional block with a kernel size of 1x1. The output of the first convolutional block (kernel size 1x1) is fed to two convolutional blocks with kernel sizes 3x3 and 5x5, while

Figure 4.2: Convolutional block

Figure 4.3: Inception block

the output of the maxpooling layer is fed to a convolutional block of kernel size 1x1. At the final stage of an inception block, the output of convolutional blocks with kernel sizes 1x1, 3x3, 5x5, and 1x1 (after maxpool) are concatenated as a consolidated output.

Table 4.1 summarizes the feature extractor inception-variant network. In Table 4.1, *in-c* represents input channel and *out-c* for output channel. *ks*, *pad*, and *s* are kernel-size, padding and strides, respectively. Each inception block in Table 4.1 has a concatenated output channel *out-c* of *out-1x1*, *out-3x3*, *out-5x5*, and *out-1x1-MP*. *out-1x1* is the output of the convolutional block with kernel size 1x1, *out-3x3* is the output of the convolutional block with kernel size 3x3, *out-5x5* is the output of the convolutional block with kernel size 5x5, and *out-1x1-MP* is the output of the convolutional block with kernel size 1x1 after the maxpooling layer. The feature extractor network has an input shape of (128, 64, 1) and outputs a flattened 256-long vector as the embedding/extracted feature. The feature extractor network has a total of 378,244 parameters.

Table 4.1: Feature extractor (Inception) model architecture

Layer type	No. of filters	Layer properties	Output shape	No. of parameters
Input	-	-	(128, 64, 1)	
Conv. Block	16	$in-c=1, out-c=16, ks=(7,7), pad=(3,3), s=(2,2)$	(16, 32, 64)	832
MaxPool2D	-	$in-c=16, out-c=16, ks=(3,3), pad=(1,1), s=(2,2)$	(16, 16, 32)	-
Conv. Block	48	$in-c=16, out-c=48, ks=(7,7), pad=(3,3), s=(2,2)$	(48, 16, 32)	7,056
MaxPool2D	-	$in-c=48, out-c=48, ks=(3,3), pad=(1,1), s=(2,2)$	(48, 8, 16)	-
Inception Block	64	$in-c=48, out-1x1=16, out-3x3=32, out-5x5=8, out-1x1-MP=8, out-c=64$	(64, 8, 16)	10,484
Inception Block	120	$in-c=64, out-1x1=32, out-3x3=48, out-5x5=24, out-1x1-MP=16, out-c=120$	(120, 8, 16)	24,736
MaxPool2D	-	$in-c=120, out-c=120, ks=(3,3), pad=(1,1), s=(2,2)$	(120, 4, 8)	-
Inception Block	128	$in-c=120, out-1x1=48, out-3x3=52, out-5x5=12, out-1x1-MP=16, out-c=128$	(128, 4, 8)	23,940
Inception Block	128	$in-c=128, out-1x1=40, out-3x3=56, out-5x5=16, out-1x1-MP=16, out-c=128$	(128, 4, 8)	28,518
Inception Block	128	$in-c=128, out-1x1=32, out-3x3=64, out-5x5=16, out-1x1-MP=16, out-c=128$	(128, 4, 8)	32,338
Inception Block	132	$in-c=128, out-1x1=28, out-3x3=72, out-5x5=16, out-1x1-MP=16, out-c=132$	(132, 4, 8)	38,320
Inception Block	208	$in-c=132, out-1x1=64, out-3x3=80, out-5x5=32, out-1x1-MP=32, out-c=208$	(208, 4, 8)	54,976
MaxPool2D	-	$in-c=208, out-c=208, ks=(3,3), pad=(1,1), s=(2,2)$	(208, 2, 4)	-
Inception Block	208	$in-c=208, out-1x1=64, out-3x3=80, out-5x5=32, out-1x1-MP=32, out-c=208$	(208, 2, 4)	65,920
Inception Block	256	$in-c=208, out-1x1=96, out-3x3=96, out-5x5=32, out-1x1-MP=32, out-c=256$	(256, 2, 4)	91,124
AdaptiveAvgPool2D	-	$in-c=256, out-c=256, ks=(7,7), pad=(1,1), s=(1,1)$	(256, 1, 1)	-
Flattened Output	-	-	(256)	-

Table 4.2: Classifier model architecture

Layer type	No. of neurons	Activation function	Output shape	No. of parameters
Input	-	-	(256)	
Linear (Fully-connected)	128	ReLU	(128)	32,896
Linear (Fully-connected)	11	Softmax	(11)	1,419

Table 4.3: Discriminator model architecture

Layer type	No. of neurons	Activation function	Output shape	No. of parameters
Input	-	-	(256)	
Linear (Fully-connected)	128	LeakyReLU	(128)	32,896
Linear (Fully-connected)	64	LeakyReLU	(64)	8,256
Linear (Fully-connected)	1	Sigmoid	(1)	65

4.1.3 Classifier

The classifier network comprises two fully-connected layers and takes in the flattened output from the feature extractor of input size 256. The first fully-connected layer connects the 256 size input units into 128 neurons. The second fully-connected layer connects 128 neurons to 11, where 11 represents the predictions for the 11 sound classes of interest.

The ReLU activation function is used for the intermediate hidden layer, whereas the last layer employs a softmax function that outputs the predictions of the sound classes. The classifier has a total of 34,315 parameters. Table 4.2 summarizes the fully-connected classifier network.

4.1.4 Discriminator

The discriminator network consists of three fully-connected layers and takes in the flattened output from the feature extractor of input size 256. The first fully-connected layer connects the 256 size input units into 128 neurons, where

Figure 4.4: Overview of downstream ESC task

the second fully-connected layer has an input of 128 to 64 neurons. The last fully-connected layer then connects 64 neurons into 1, where 1 represents the prediction for the domain (source and target) classes, given that source labels were set to 0 and target labels as 1.

The LeakyReLU activation function is used for all of the activation functions within the discriminator, with a negative slope set to 0.2, except the output, which is a sigmoid function that outputs the prediction of the domains (source or target domain). Table 4.3 summarizes the fully-connected discriminator network.

4.1.5 Architecture summary

The sole purpose of the discriminator is to perform adversarial training on the feature extractor. The domain adapted, adversarial-trained feature extractor would then be coupled with only the classifier to perform the downstream ESC task. For the downstream ESC task, we combine stacks of inception blocks at the feature extractor and flattens the final output of the feature extractor before propagating into the fully-connected classification block that outputs the class probabilities, given the data sample, as shown in Figure 4.4.

The composition of each inception block is preserved from [71], but the number of filters is reduced for the different sizes of convolutional filters in each of the Inception blocks. This approach results in a much lighter model (feature extractor (*378,244 parameters*) + classifier (*34,315 parameters*)) with a total of 412,559 parameters.

4.2 Model Training

The training of an adapted network comprises two stages. The first stage is a pre-training stage with only the source domain data. The second stage of training is a domain adversarial training stage, where we would require the discriminator as described in the previous section. The following subsections will describe the training methods in detail with illustrations of the training stages shown in Figure 4.5. The term “source-only” domain data represents data collected from the source domain (at locations L1 and L2 with microphone M1) and is commonly referred to as the domain that the model is familiar with (performs well). The term “target” domain data represents data collected from the target domain (at location L3 with microphone M2) and is commonly referred to as the domain that the model is unfamiliar with (does not perform well) before adaptation.

4.2.1 Pre-training stage

We train the feature extractor G_f and the classifier G_c at the pre-training stage, only with the source domain data. The classifier G_c uses a softmax function, $\text{softmax}(\vec{z}) = \left[\frac{\exp(z_i)}{\sum_{j=1}^{|z|} \exp(z_j)} \right]_{i=1}^{|z|}$ (given $|z|$ as the number of classes in the multi-class classifier) to denote the conditional probability for each component of vector $G_c(G_f(\mathbf{x}_i))$. For each i^{th} example, given an example $\mathbf{x}_i$ and the class label y_i , the negative log-likelihood classification loss L_y can be written as: $L_y(G_c(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_c(G_f(\mathbf{x}_i))_{y_i}}$. Equation (1) describes the loss calculated per i^{th} sample and training is done by optimizing (2) over the entire training set. As our method is an unsupervised domain adaptation method, y describes labels for all source-only domain data. Figure 4.5a illustrates the pre-training stage over the training set, consisting of only labeled source-only domain data (X_s, y) .

$$L_y^i(\theta_f, \theta_c) = L_y(G_c(G_f(X_i; \theta_f); \theta_c), y_i) \quad (1)$$

$$J(\theta_f, \theta_c) = \frac{1}{n} \sum_{i=1}^n L_y^i(\theta_f, \theta_c) \quad (2)$$

A total of 37,536 training data samples is used to pre-train the model. The pre-trained model is trained over 50 epochs, with 1,173 mini-batches of batch-size 32 per epoch. We use stochastic gradient descent (SGD) updates with a learning rate of 10^{-3} and momentum of 0.9 coupled with a scheduler for learning rate decay of 10^{-1} at a step size of 10. The training is done on a graphical processing unit (GPU) using the PyTorch [108] framework. Note that the pre-trained model is also the “source only” model discussed in the results chapter (Chapter 5) since the pre-trained model is only trained with data from the source domain. Equations (3) and (4) describe the parameter updates for the parameters in the feature extractor θ_f and the classifier θ_c , w.r.t L_y^i , during training, where α denotes the learning rate.

$$\theta_f \longleftarrow \theta_f - \alpha \left(\frac{\partial L_y^i}{\partial \theta_f} \right) \quad (3)$$

$$\theta_c \longleftarrow \theta_c - \alpha \left(\frac{\partial L_y^i}{\partial \theta_c} \right) \quad (4)$$

To attain a good saddle point during adversarial training is challenging because both the classifier and the discriminator are minimizing and maximizing (respectively) the overall loss simultaneously. Therefore, the motivation of the pre-training stage is to initialize the parameters θ_f with a pre-trained feature extractor G_f for effective adversarial training. An ablation study is conducted in Section 5.5 on the effectiveness of the pre-training stage.

4.2.2 Domain-adversarial training stage

At the domain adversarial training stage, the architecture includes the discriminator. At this stage, the feature extractor is connected to both the classifier and the discriminator and the extracted feature embeddings are fed into both networks.

4.2.2.1 Target domain data-split for adversarial training

To ensure that the target domain data used for adversarial training and model testing are mutually exclusive, we split the test set described in Table 3.4 into two parts (approximately halved per class). The first subset is used for adversarial training and the next subset as an unseen test set. Table 4.4 summarizes the two test set splits (for adversarial training and model testing). The term “unseen test set” denotes the unseen test set split from this point of the thesis onwards.

Table 4.4: Number of data samples in test sets for adversarial training and for testing

	Test Set - Adversarial Training (Target domain - L3)	Test Set - Testing (Unseen) (Target domain - L3)
Data samples	1492	1486

4.2.2.2 Domain-adversarial training

The discriminator G_d uses a sigmoid function, $\text{sigmoid}(\vec{z}) = \left[\frac{1}{1+\exp(z_j)} \right]_{j=1}^{|z|}$ to denote the domain probability for each component of vector $G_d(G_f(\mathbf{x}_i))$. The binary cross-entropy (BCE) loss for L_y can be written as:

$L_d(G_d(G_f(\mathbf{x}_i)), d_i) = d_i \log \frac{1}{G_d(G_f(\mathbf{x}_i))} + (1 - d_i) \log \frac{1}{1-G_d(G_f(\mathbf{x}_i))}$ where d_i denotes the binary domain label for the i^{th} sample, which indicates whether $\mathbf{x}_i$ is from the source domain ($d_i = 0$) or the target domain ($d_i = 1$). Equation (5) describes the discriminator loss per i^{th} sample. Adversarial training is done by optimizing (6) over both source domain samples and the target domain samples (set aside for adversarial training, Section 4.2.2.1). n denotes the total source domain examples and N , a superset of n , includes the target examples n' that are required for adversarial training. The description for the classifier loss L_y is identical as in (1).

$$L_d^i(\theta_f, \theta_d) = L_d(G_d(G_f(X_i; \theta_f); \theta_d), d_i) \quad (5)$$

$$J(\theta_f, \theta_c, \theta_d) = \frac{1}{n} \sum_{i=1}^n L_y^i(\theta_f, \theta_c) - \lambda \left[\frac{1}{n} \sum_{i=1}^n L_d^i(\theta_f, \theta_d) + \frac{1}{n'} \sum_{i=n+1}^N L_d^i(\theta_f, \theta_d) \right] \quad (6)$$

The hyperparameter λ is used to tune trade-offs between losses from the classifier and the discriminator during adversarial training [92]. At the beginning of adversarial training, the domain loss would be extremely noisy, which means that backpropagating a large penalty for domain confusion to the feature extractor layers could restrict the feature extractor’s ability to learn. Therefore, we start the adversarial training with a smaller λ value for penalizing w.r.t the discriminator (domain classification) and we increase the λ value iteratively throughout adversarial training. Here we define lambda as $\lambda = \left[\frac{2 \times 10^{-1}}{1 + \exp(-10 \times \frac{epoch}{maxepoch})} \right] - 1$ where *epoch* is an integer of which the epoch number is at during training and *maxepoch* is the total number of epochs pre-defined before training.

A total of 39,028 data samples (comprising both source domain (37,536 samples) and target domain (1,492 samples)) is used for adversarial training. The model is trained over 50 epochs, with 1,219 mini-batches of batch-size 32 per epoch. The optimizer, the scheduler for the optimizer and their corresponding hyperparameters remains the same as the pre-training stage. Equations (7) and (8) describe the parameter updates for the parameters in the feature extractor θ_f and the discriminator θ_d , w.r.t L_d^i (and L_y^i for the case of θ_f), during training. The parameter updates for the classifier θ_c remains the same as in (4).

$$\theta_f \leftarrow \theta_f - \alpha \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right) \quad (7)$$

$$\theta_d \leftarrow \theta_d - \alpha \lambda \left(\frac{\partial L_d^i}{\partial \theta_d} \right) \quad (8)$$

4.2.2.3 Gradient Reversal Layer

The gradient reversal layer (GRL) proposed in [92] is used to achieve domain confusion within the feature extractor. The GRL has no parameters and acts as an identity transformation during forward propagation, but inverses the gradient

(multiply by -1) during backpropagation. Effectively, at backpropagation, the partial derivatives $\frac{\partial L_d}{\partial \theta_f}$ sent back to update parameters of the feature extractor, θ_f , gets multiplied by -1, hence, replacing $\frac{\partial L_d}{\partial \theta_f}$ with $-\frac{\partial L_d}{\partial \theta_f}$. We treat the GRL as pseudo function, $\mathcal{R}$, where (9) and (10) describe the GRL during forward propagation and backpropagation, respectively.

$$\mathcal{R}(\mathbf{x}) = \mathbf{x} \quad (9)$$

$$\frac{d\mathcal{R}}{d\mathbf{x}} = -\mathbf{I} \quad (10)$$

$$\begin{aligned} J(\theta_f, \theta_c, \theta_d) = & \frac{1}{n} \sum_{i=1}^n L_y(G_c(G_f(X_i; \theta_f); \theta_c), y_i) \\ & - \lambda \left[\frac{1}{n} \sum_{i=1}^n L_d(G_d(\mathcal{R}(G_f(X_i; \theta_f)); \theta_d), d_i) \right. \\ & \left. + \frac{1}{n'} \sum_{i=n+1}^N L_d(G_d(\mathcal{R}(G_f(X_i; \theta_f)); \theta_d), d_i) \right] \end{aligned} \quad (11)$$

In Equation (11), with the introduction of the GRL, $\mathcal{R}$ (inserted between the feature extractor, G_f and the discriminator, G_d), we maximize the loss, L_d , of the discriminator (domain classifier), during adversarial training and simultaneously, minimize classification loss, L_y , on the source domain using the same feature extractor (same convolutional layers).

By impairing the feature extractor's ability to classify between domains, we influence the feature extractor to extract embeddings proficient in classifying the sound classes, but not the domains, thus, creating domain confusion within the extractor network. The GRL method has led to improved results in image tasks that require adaptation to different domains [92].

(a) Stage one of model training - pre-training the model with source domain data, which includes a feature extractor and a fully-connected classifier.

(b) Stage two of model training - adversarial training with both source domain and target domain data. The architecture consists of a feature extractor coupled with a classifier, forming a standard feed-forward architecture. Adversarial training is done with an additional discriminator (domain classifier), which predicts between the source and target domain, where the domain loss is calculated. The discriminator is connected to the feature extractor through a gradient reversal layer that multiplies the gradient by a negative constant during backpropagation.

(c) Model at testing and deployment, which includes an adversarial trained (domain adapted) feature extractor and a fully-connected classifier.

Figure 4.5: Unsupervised domain adaptation - model architecture, training and testing. Black arrows indicate the downstream forward propagation while back-propagation during training is highlighted in red.

4.2.3 Exporting the models

After the domain adversarial training stage, only the classifier and feature extractor are exported. The discriminator G_d is not exported. Two differently trained models $G_c(G_f(X; \theta_f); \theta_c)$ are exported, one with only the source domain data as described in Section 4.2.1 and the other with domain adversarial trained as described in Section 4.2.2. Both exported models are evaluated on samples from both source (validation set, locations L1 and L2) and target domains (unseen test set, location L3). Figure 4.5c illustrates the downstream classification task of the exported models. Results in comparison of the two models (“source only” and “domain adversarial” trained) are discussed in the next chapter.

4.3 Conclusion

To set up a WASN-ESC domain adaptation study, we set our training and validation sets as the source domain data and the test set as the target domain data. Our model architecture for domain adversarial training consists of three parts: a feature extractor network, classifier network and discriminator network. We elaborate on the two training methods - training with only the source domain data and an unsupervised learning domain adaptation method that employs adversarial training. The experiment results of the methods are presented in the next chapter.

Chapter 5

Experiment Results and Discussions

This chapter presents the experiment results of our proposed training methodology for the domain adaptation model, compared with a model trained with only source domain data. Both models are evaluated on both the validation set collected from locations L1 & L2 and the unseen test set from location L3. The unseen test set is annotated for evaluation purposes (required ground-truth).

5.1 Evaluation metrics

The precision, recall, F1-score, and area under precision-recall curve (AUPRC) for each sound class, as well as the macro-averaged scores of the trained model on the validation and test sets, are summarized in the subsequent subsections. Given M data samples, each classified into exactly one of C classes, the precision (P), recall (R), F1-score (F_1), and macro-averaged score (A_{macro}) are, respectively, defined as

$$P = \frac{M_{\text{TP}}}{M_{\text{TP}} + M_{\text{FP}}} \quad (12)$$

$$R = \frac{M_{\text{TP}}}{M_{\text{TP}} + M_{\text{FN}}} \quad (13)$$

$$F_1 = \frac{2PR}{P + R} \quad (14)$$

$$A_{\text{macro}} = \frac{1}{C} \sum_{k=1}^C \frac{M_{\text{TP},k}}{M_k} \quad (16)$$

where M_{TP} denotes the number of true positives, M_{FP} denotes the number of false positives, M_{FN} denotes the number of false negatives, $M_{\text{TP},k}$ denotes the number of true positives belonging to class k , and M_k denotes the number of data samples with ground truth label being class k . By this definition, we have $M_{\text{TP}} = \sum_{k=1}^C M_{\text{TP},k}$ and $M = \sum_{k=1}^C M_k$.

The AUPRC is calculated as a discrete estimate of the area under a curve generated by plotting the precision and recall of a model at various confidence thresholds.

The metrics described above are in the range from 0 to 1, where an ideal classifier should have a maximum score of 1. We use segment-based scoring with the segment size of 128 time-frame and 64 Mel bands for each data sample of shape (128, 64, 1).

It is important to note that both the validation and test sets do not represent the entirety of the sounds that could potentially occur at the deployed locations of the WASN. The datasets should be thought of as a subset of all possible environmental sounds at the deployed location.

5.2 Model parameter pruning

Deploying an extensive neural network with millions of parameters would be computationally expensive to run on a WASN, considering that the neural network is often running concurrently with other software components at the edge. Therefore, it is consequential to prune the model to achieve a lightweight model capable

Table 5.1: Model pruning - macro-averaged F1-scores of model variants, with and without domain adversarial training

Model	Pruned Ratio	No. of Params.	Reduction in Params. (%)	F1-Score (macro-avg)	
				Source Domain	Target Domain
Inception (Source-only)	1	6,507,995	0%	0.835	0.547
Inception (Adaptive)				0.817	0.833
Inception (Source-only) /2	2	1,634,739	74.88%	0.812	0.560
Inception (Adaptive) /2				0.809	0.810
Inception (Source-only) /4	4	412,559	93.66%	0.809	0.561
Inception (Adaptive) /4				0.772	0.857
Inception (Source-only) /8	8	105,081	98.38%	0.785	0.614
Inception (Adaptive) /8				0.793	0.815

of accurate predictions. In this section, we employ a simple model parameter minimization technique - by cutting down the number of filters and fully-connected nodes of the model by a factor of 2, 4 and 8 from the original Inception network. The composition of each inception block is preserved to maintain the integrity of the original Inception network, where the factor reduces only the number of filters used per block. The method results in three additional network variants with fewer parameters, of which the original model, and the models cut by a factor of 2, 4 and 8, consist of 6,507,995, 1,635,739, 412,559, and 105,081 parameters, respectively. All of the networks are evaluated with two different training methods, one with source domain data only training - denoted with “Source-only”, and the other with domain adversarial training, denoted with “Adaptive”, as shown in Table 5.1.

Table 5.1 shows the different network architecture size results for source only training and domain adaptive training. The “reduction in params. (%)” column in Table 5.1 shows the reduction in the number of parameters after pruning (in terms of percentage), relative to the original number of parameters of the InceptionNet. From Table 5.1, we can observe that even when reducing the number of parameters by 4-folds, we could achieve comparable results attained by the larger networks.

It is also observed, from Table 5.1, that the Inception /4 variant displayed the best domain adaptive results. Therefore, as a compromise between performance and computation costs at the WASN, the Inception /4 model is chosen as the default model architecture for deployment.

Note that all of the results are derived from the variant Inception /4 (Total parameters: 412,559) model architecture from here on. The subsequent sections will cover the performance of the Inception /4 model architecture in detail.

5.3 Experiment Results

In this section, we report the performance of our domain adversarial trained network in comparison with a source-only trained network. With regards to the figures and tables presented in this section, “Source-Only” represents the non-adapted Inception-variant network, trained with only source domain data and “Domain adversarial” represents the adapted Inception-variant network, trained with domain adversarial training methods. The source domain described in this section represents data collected from locations L1 and L2, while the target domain would comprise data collected only in location L3. The metrics presented are in the sequence of precision, recall, F1-score, AUPRC, followed by confusion matrices.

First, we compare both training methods on the unseen test set (data samples from the target domain at location L3). Next, we compare both training methods on the validation set (data samples from source domain at locations L1 and L2).

For an objective review, all models are evaluated on the same datasets (validation set (*as described in Table 3.4*) and unseen test set (*as described in Section 4.2.2.1*)) throughout all experiments.

We present Tables 5.2, 5.3, 5.4, and 5.5 with additional Δ columns to indicate the difference between the two training methods. Values under the Δ columns are denoted with + (positive) if the results acquired from domain adversarial training is greater than the results obtained from source-only training, and – (negative) if otherwise. We express with ± 0.000 in the case of identical results derived from

Table 5.2: Class-wise precision and recall **comparison of the two training methods** evaluated on the **unseen test set** (target domain, L3). The columns Precision Δ and Recall Δ signify the difference in precision and recall between the two training methods.

Class	Source-only training		Domain adversarial training		Precision Δ	Recall Δ
	Precision	Recall	Precision	Recall		
00-Ambient	0.581	0.309	0.920	0.825	+ 0.339	+ 0.516
01-Construction	0.663	0.676	0.887	0.922	+ 0.224	+ 0.246
02-Vehicle	0.848	0.807	0.826	0.771	− 0.022	− 0.036
03-Music	0.323	0.064	0.746	0.917	+ 0.423	+ 0.853
04-Shout	0.674	0.574	0.716	0.673	+ 0.042	+ 0.099
05-Rain	0.910	0.664	0.971	0.993	+ 0.061	+ 0.329
06-Birds	0.569	0.838	0.835	1.000	+ 0.266	+ 0.162
07-CarHorn	0.371	0.821	0.900	0.964	+ 0.529	+ 0.143
08-Alarm	0.116	1.000	0.676	1.000	+ 0.560	± 0.000
09-ImpactSound	0.773	0.607	0.939	0.821	+ 0.166	+ 0.214
10-HumanVoice	0.646	0.753	0.889	0.777	+ 0.243	+ 0.024
Overall Performance (Macro Average)	0.589	0.674	0.846	0.879	+ 0.257	+ 0.205

the two training methods.

A comparison on precision and recall between source-only training and domain adversarial training is shown in Table 5.2. Observations show that the domain adversarial training method outperforms the source-only training method with an overall (macro average) improvement of + 0.257 and + 0.205 reflected in both “Precision Δ ” and “Recall Δ ” columns, respectively on the unseen test set (target domain, location L3). A general positive inclination is observed in the Δ columns of Table 5.2 for class-wise precision and recall except for a slight degradation in the “Vehicle” class - the only class that underperformed after domain adversarial training. However, the slight degradation of the “Vehicle” class can be considered minimal (− 0.022 and − 0.036 for Precision Δ and Recall Δ) when compared to the improvement achieved in the other classes such as the “Music” class (+ 0.423 and + 0.853 for Precision Δ and Recall Δ).

Next, the source-only training and domain adversarial training methods are

Table 5.3: Class-wise F1-score and AUPRC **comparison of the two training methods** evaluated on the **unseen test set** (target domain, L3). The columns F1-score Δ and AUPRC Δ signify the difference in F1-score and AUPRC between the two training methods.

Class	Source-only training		Domain adversarial training		F1-score Δ	AUPRC Δ
	F1-score	AUPRC	F1-score	AUPRC		
00-Ambient	0.404	0.318	0.870	0.794	+ 0.466	+ 0.476
01-Construction	0.670	0.472	0.904	0.823	+ 0.234	+ 0.351
02-Vehicle	0.827	0.707	0.798	0.663	- 0.029	- 0.044
03-Music	0.106	0.122	0.823	0.693	+ 0.717	+ 0.571
04-Shout	0.620	0.417	0.694	0.505	+ 0.074	+ 0.088
05-Rain	0.768	0.636	0.982	0.965	+ 0.214	+ 0.329
06-Birds	0.678	0.493	0.910	0.835	+ 0.232	+ 0.342
07-CarHorn	0.511	0.308	0.931	0.869	+ 0.420	+ 0.561
08-Alarm	0.208	0.116	0.806	0.676	+ 0.598	+ 0.560
09-ImpactSound	0.680	0.484	0.876	0.778	+ 0.196	+ 0.294
10-HumanVoice	0.695	0.528	0.829	0.729	+ 0.134	+ 0.201
Overall Performance (Macro Average)	0.561	0.418	0.857	0.757	+ 0.296	+ 0.339

compared on evaluation metrics F1-score and AUPRC. Table 5.3 shows F1-score and AUPRC results for the two training methods on the unseen test set (target domain, location L3). The same trend is observed in Table 5.3 as with Table 5.2 since F1-score and AUPRC are derived from precision and recall values. The class-wise F1-score and AUPRC metrics improved for all classes except for the “Vehicle” sound class after domain adversarial training. Similarly, the slight degradation for the “Vehicle” sound class can be considered minimal when compared for other classes such as the “Music” class (+ 0.717 and + 0.571 for F1-score Δ and AUPRC Δ) and the “Alarm” class (+ 0.598 and + 0.560 for F1-score Δ and AUPRC Δ). Conclusively, the overall performance (macro average) for F1-score and AUPRC metrics (Table 5.3) have improved by + 0.296 and + 0.339 (F1-score Δ and AUPRC Δ columns) for domain adversarial training when compared to source-only training, when evaluated on the unseen test set (target domain, location L3).

		Predicted										
		Ambient	Constr.	Vehicle	Music	Shout	Rain	Birds	C. Horn	Alarm	Impact S.	Human V.
Actual	Ambient	30	0	6	0	0	3	3	0	27	0	27
	Construction	0	65	3	0	0	0	3	2	22	0	0
	Vehicle	0	7	75	3	0	0	0	0	7	0	5
	Music	0	16	0	9	0	0	36	11	22	0	3
	Shout	8	0	0	0	59	0	0	3	17	0	8
	Rain	28	0	0	0	0	67	0	0	2	1	0
	Birds	0	0	0	0	1	0	85	2	8	0	1
	CarHorn	0	0	0	21	0	0	0	67	0	10	0
	Alarm	0	0	0	0	0	0	0	0	99	0	0
	ImpactSound	19	0	0	0	0	0	3	0	0	64	12
	HumanVoice	0	0	0	3	13	0	4	0	5	0	72

Figure 5.1: Confusion matrix of **source only** trained model evaluated on **unseen test set** (L3). The $(i, j)^{\text{th}}$ entry of each matrix denotes the percentage of tracks w.r.t the ground truth label i that are predicted by the model for class j

Subsequently, we compare the normalized (out of 100%) confusion matrices of both methods to denote the differences in performance of the two training methods, presented in Figures 5.1 and 5.2. First, we look at the performance of the source-only trained model on the unseen test set shown in Figure 5.1. From Figure 5.1, the worse performing class predicted by the source-only trained model is the “Music” sound class. The source-only trained model only predicted true positives of 9% and misclassified the majority of the “Music” data samples as “Birds” (36% predicted as “Birds”) sound class and as “Alarm” (22% predicted as “Alarm”) sound class. It is also observed that the majority of the data samples from other classes (such as “Ambient”, “Construction”, “Music”, and “Shout”) are predicted as “Alarm”; resulting in many false positives instances at the “Alarm” column (data samples from other classes misclassified as “Alarm” by the source-only trained model). The false positives observed in the “Alarm” column of the confusion matrix (Figure 5.1) correlate with the poor precision score of 0.116 (source-only trained model precision score for “Alarm” class) shown in Table 5.2. Illustrated in the confusion matrix (Figure 5.1), the source-only trained model also indicated difficulties in discriminating the data samples of the “Ambient” sound class. The source-only trained model’s display of poor performance on the “Ambient” sound class elaborates the different environmental influences present

		Predicted										
Actual		Ambient	Constr.	Vehicle	Music	Shout	Rain	Birds	C. Horn	Alarm	Impact S.	Human V.
	Ambient	81	1	10	1	0	1	1	0	1	0	0
	Construction	0	92	0	7	0	0	0	0	0	0	0
	Vehicle	4	4	78	6	0	0	1	0	0	0	2
	Music	0	0	0	91	0	0	5	0	1	0	0
	Shout	0	0	0	7	71	0	0	0	2	0	15
	Rain	1	0	0	0	0	98	0	0	0	0	0
	Birds	0	0	0	0	0	0	99	0	0	0	0
	CarHorn	0	0	0	0	0	0	0	99	0	0	0
	Alarm	0	0	0	0	0	0	0	0	99	0	0
	ImpactSound	23	1	0	1	0	0	0	0	0	73	0
	HumanVoice	0	0	0	6	13	0	2	0	0	0	77

Figure 5.2: Confusion matrix of **domain adversarial** trained model evaluated on **unseen test set** (L3). The $(i, j)^{\text{th}}$ entry of each matrix denotes the percentage of tracks w.r.t the ground truth label i that are predicted by the model for class j

in the new unseen location and indicates how the model would fail to generalize to a new environment/location setting.

Next, we present the confusion matrix of the domain adversarial trained model, evaluated on the same unseen test set (target domain, location L3) in Figure 5.2. The true positive instances reflected in Figure 5.2 (confusion matrix derived from the domain adversarial trained model) indicate improvements for all classes when compared to Figure 5.1 (confusion matrix derived from the source-only trained model). The domain adversarial trained model achieved 91% true positives for the “Music” sound class; a vast improvement when compared to the 9% true positives predicted by the source-only trained model. The results of the domain adversarial model (Figure 5.2) also show that the false positive instances at the “Alarm” sound class (column) are greatly reduced. The improvement on the false positives at the “Alarm” class boosted the precision score from 0.116 (source-only trained model) to 0.676 (domain adversarial trained model), with an improvement of +0.560 (Precision Δ) as shown in Table 5.2. Finally, the domain adversarial model achieved 81% true positives on the “Ambient” class (compared to 30% by the source-only trained model, with an improvement of 51%), exhibiting adaptation to the new unseen location.

We visualize the feature embeddings extracted from the feature extractor of

both the source only and domain adversarial trained models. Data samples from the validation set (locations L1 and L2) and the unseen test set (location L3) are provided as input to both source only and domain adversarial trained models. The embeddings extracted from the feature extractors of the two models are projected in a 2D space using the t-distributed stochastic neighbor embedding (t-SNE) [109] dimensionality reduction technique. The validation set provided is cut by half with the intent for clearer visualization purposes. The total data sample count for both the validation set and unseen test sets used in this section is 1406 and 1486 samples, respectively. Figure 5.3 shows t-SNE visualizations for both source-only and domain adversarial training. The numbers plotted in the t-SNE plots are the indexed class labels (zero-indexed), where each number corresponds to a given sound class as tagged in Table 3.4.

Figure 5.3a shows that the source-only trained model is unable to generalize for certain classes like “Ambient” and “Music” (class indexed 0 and 3, respectively) between the source and target data samples where the embeddings were mapped far apart and non-overlapping in the 2D visualization. The domain adversarial trained model, however, is able to generalize between the domains well. The overlaps between the source and target samples for the classes of interest are much more distinct than the source-only trained model, as illustrated in Figure 5.3b. The overlaps of the dimension-reduced extracted feature embeddings between the source and target data samples signify that the feature extractor of the domain adversarial trained model does not discriminate between the domains and is able to generalize (indexed classes are clustered together) regardless of whether the data sample is from the source or target domain. From Figure 5.3b, the overlaps of the same indexed classes and the wide distance between the different indexed classes indicate that the feature extractor of the domain adversarial trained model is able to generalize between the domains and, at the same time, discriminate between the sound classes. These observations correlate with the scoring metrics when reviewing the models’ scores for both the validation and unseen test sets.

(a) t-SNE visualizations of **source only** trained model extracted feature embeddings for **val set** (L1 & L2) from the source domain (red) and **unseen test set** (L3) from the target domain (blue)

(b) t-SNE visualizations of **domain adversarial** trained model extracted feature embeddings for **val set** (L1 & L2) from the source domain (red) and **unseen test set** (L3) from the target domain (blue)

Figure 5.3: t-SNE visualizations without (a) and with (b) domain adversarial training.

Table 5.4: Class-wise precision and recall **comparison of the two training methods** evaluated on the **validation set** (source domain, L1 & L2). The columns Precision Δ and Recall Δ signify the difference in precision and recall between the two training methods.

Class	Source-only training		Domain adversarial training		Precision Δ	Recall Δ
	Precision	Recall	Precision	Recall		
00-Ambient	0.544	0.883	0.544	0.775	± 0.000	$- 0.108$
01-Construction	0.886	0.989	0.908	1.000	$+ 0.022$	$+ 0.011$
02-Vehicle	0.920	0.832	0.975	0.829	$+ 0.055$	$- 0.003$
03-Music	0.743	0.524	0.703	0.495	$- 0.040$	$- 0.029$
04-Shout	0.858	0.869	0.747	0.840	$- 0.111$	$- 0.029$
05-Rain	0.976	0.881	0.986	0.916	$+ 0.010$	$+ 0.035$
06-Birds	0.742	0.922	0.518	0.971	$- 0.224$	$+ 0.049$
07-CarHorn	0.862	0.984	0.948	0.866	$+ 0.086$	$- 0.118$
08-Alarm	0.900	0.675	0.990	0.421	$+ 0.090$	$- 0.254$
09-ImpactSound	0.938	0.835	0.895	0.933	$- 0.043$	$+ 0.098$
10-HumanVoice	0.495	0.833	0.460	0.851	$- 0.035$	$+ 0.018$
Overall Performance (Macro Average)	0.806	0.839	0.789	0.809	$- \mathbf{0.017}$	$- \mathbf{0.030}$

We evaluate both the source-only trained and domain adversarial trained models on the validation set (source domain, locations L1 and L2). A comparison of precision and recall between source-only training and domain adversarial training is shown in Table 5.4, and a comparison of F1-score and AUPRC metrics is shown in Table 5.5. Observations from Tables 5.4 and 5.5 show a slight degradation in performance on the validation set **after** domain adversarial training ($- 0.017$, $- 0.030$, $- 0.037$ and, $- 0.041$ for overall Precision Δ , Recall Δ , F1-score Δ , and AUPRC Δ). The slight degradation (w.r.t performance at the source domain) is generally due to the introduction of domain confusion at the feature extractor throughout domain adversarial training. However, since the focus is on adaptation to the target domain, the slight degradation at the source domain can be considered negligible because we have the advantage in deploying the model that performs better out of the two training methods for the different locations.

Table 5.5: Class-wise F1-score and AUPRC **comparison of the two training methods** evaluated on the **validation set** (source domain, L1 & L2). The columns F1-score Δ and AUPRC Δ signify the difference in F1-score and AUPRC between the two training methods.

Class	Source-only training		Domain adversarial training		F1-score Δ	AUPRC Δ
	F1-score	AUPRC	F1-score	AUPRC		
00-Ambient	0.674	0.485	0.639	0.431	− 0.035	− 0.054
01-Construction	0.935	0.878	0.952	0.908	+ 0.017	+ 0.030
02-Vehicle	0.874	0.806	0.896	0.849	+ 0.022	+ 0.043
03-Music	0.615	0.407	0.581	0.367	− 0.034	− 0.040
04-Shout	0.864	0.760	0.791	0.645	− 0.073	− 0.115
05-Rain	0.926	0.879	0.950	0.917	+ 0.024	+ 0.038
06-Birds	0.823	0.687	0.676	0.504	− 0.147	− 0.183
07-CarHorn	0.919	0.849	0.905	0.828	− 0.014	− 0.021
08-Alarm	0.771	0.636	0.591	0.467	− 0.180	− 0.169
09-ImpactSound	0.884	0.799	0.914	0.841	+ 0.030	+ 0.042
10-HumanVoice	0.621	0.419	0.597	0.397	− 0.024	− 0.022
Overall Performance (Macro Average)	0.809	0.691	0.772	0.650	− 0.037	− 0.041

Hence, the model trained with the “source-only training” method would be favourable and appropriate for deployment at locations L1 and L2, in correspondence to the better performance (out of the two training methods) at the source domain (locations L1 & L2).

5.4 Discussions

We report the performance of the domain adversarial training method in comparison with the source-only training results. We discuss the overall performance and the class-wise performance of the results gathered before and after the applied adaptation method.

5.4.1 Overall performance

When comparing the domain adversarial trained model with the source-only trained model, evaluations on the target domain data show notable improvements on the macro average precision score of 0.589 to 0.846, recall score of 0.674 to 0.879 (Table 5.2), F1-score of 0.561 to 0.857, and AUPRC of 0.418 to 0.757 (Table 5.3).

The overall performance (bottom-most) row of Tables 5.2 and 5.3 show that the overall performance improves + 0.257, + 0.205, + 0.296, and + 0.339 for overall precision, recall, F1-score, and AUPRC respectively on the unseen test set, after domain adversarial training.

5.4.2 Class-wise performance

Comparing the confusion matrices (Figures 5.1 and 5.2), our proposed method shows consistent improvement for all sound classes from the collated true positive instances. Notably, the “Music” sound class is able to achieve significant improvement from 9% of true positive instances to 91% after adaptation. Other sound classes such as “Ambient” and “CarHorn”, with previous true positive instances at 30% and 67%, have improved to 81% and 99%, respectively, after adaptation.

The Δ columns of Tables 5.2 and 5.3 indicate improvement for all classes except for the “Vehicle” sound class. However, the degradation in performance for “Vehicle” sound class is minimal (-0.017 , -0.030 , -0.037 , and -0.041 for Precision Δ , Recall Δ , F1-score Δ , and AUPRC Δ) in comparison to the increment in performance for the other classes, notably the “Music” sound class (+ 0.423, + 0.853, + 0.717, and + 0.571 for Precision Δ , Recall Δ , F1-score Δ , and AUPRC Δ) and the “Alarm” sound class (+ 0.598 and + 0.560 for F1-score Δ and AUPRC Δ).

5.5 Ablation Study

Table 5.6: Ablation study results show that the pre-training (stage one) of our training method is essential for better performance.

Model	No. of Params.	F1-Score
		Target Domain
Inception /4 (<i>Source-only</i>)		0.561
Inception /4 (<i>Adaptive, without pre-training</i>)	412,559	0.788
Inception /4 (<i>Adaptive, with pre-training</i>)		0.857

We analyze the importance and effectiveness of our proposed stage one pre-training component of our training method and compare the results with and without such component as our ablation study.

First, we include the results evaluated by the “source-only” trained model as the baseline. Next, we compare the performance of adversarial-trained models (with and without the pre-training stage) as shown in Table 5.6. We observe that the performance drops from an F1-Score of 0.857 to 0.788 when pre-training is **not** applied.

Although applying domain adversarial training without the pre-training stage has shown an improvement in performance when compared to the baseline “source-only” trained model, our ablation study shows that incorporating stage 1 (pre-training stage) of our proposed method yield better results when compared to applying domain adversarial training without a pre-training stage. From this ablation study, we conclude that the proposed domain adversarial training method coupled with a pre-training stage would enhance the model’s performance on the target domain.

5.6 Conclusion

We review the performance of the source-only trained model on the unseen test set (target domain, location L3) and show that the model failed to generalize to a new location without domain adversarial training. We present Tables 5.2 and 5.3, indicating significant performance improvement on the unseen test set (target domain, location L3) after domain adversarial training (+ 0.257, + 0.205, + 0.296 and, + 0.339 for overall Precision Δ , Recall Δ , F1-score Δ , and AUPRC Δ). A lightweight, domain adversarial trained model capable of accurate predictions is achieved by performing model parameter pruning. Finally, our ablation study shows that pre-training the model yields better results (than without the pre-training stage) before domain adversarial training.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

WASN-ESC is a significant and momentous area of research with many applications. While there are many published works on machine listening tasks, WASN-ESC is still relatively new research. The WASN aspect of the task extends to a network of sensor nodes, with each sensor node running its own sound classifier. The ESC task can be increasingly difficult to manage as the sensor network increases in size due to novel acoustic conditions at different deployed locations. The effectiveness of a WASN setup is proportionate to the scalability of the sensor nodes - where a WASN large-scale setup would be able to gather much more metadata and information when conducting studies for urban noise. As presented in the previous chapter, the source-only trained model is considerably compromised when ported from one location to another unfamiliar location. The source-only trained model is not able to generalize well to a new location (L3) even though the source and target sound classes are similar - which illustrates the domain shift problem. Due to novel environmental influences, the domain shift problem would constrain the scalability of such WASN technology and restrain its capabilities to achieve a large-scale setup. Current adaptation methods require manual annotations by the data annotators to label newly collected sounds from an unseen deployed location. This process is labour-intensive, mundane and vul-

nerable to inter-annotator variability. Thus, domain adaptation for WASN-ESC tasks is an important area of research as it does not only address the environmental acoustic differences within an estate, but also the capacity to extend out to different cities and different recording devices.

To the best of the author’s knowledge, this is the first research to experiment with a real-world dataset for ESC tasks from a WASN setup for sounds collected in Singapore. In this study, we propose a real-world dataset for ESC tasks from a WASN setup and addressed the issue of domain adaptation when a sensor node (coupled with a deep network for ESC) is deployed to different locations. In terms of scalability, we have shown that our method is able to scale up deep networks for different locations using unsupervised learning for location adaptation purposes, thus, without going through the demanding manual annotation for the new records that are collected in a new deployed location. We show that our Inception variant network is able to keep model parameters at minimal and reduce the required computational power in a sensor node whilst maintaining accurate predictions. Our ablation study shows that the pre-training stage of our methodology is essential for better overall performance. Finally, our results show that the proposed Inception variant network, coupled with the proposed method, improved the performance when evaluated with data in the target domain (at a different location and deployed with a different recording device) - significant performance improvement on the target domain (+ 0.257, + 0.205, + 0.296 and, + 0.339 for overall Precision Δ , Recall Δ , F1-score Δ , and AUPRC Δ) can be attained after domain adversarial training.

Per the research questions presented in Section 1.2, we have proposed a suitable knowledge transfer approach for location-wise domain adaptation by applying our proposed method for domain adversarial training (w.r.t research question 1). The results of domain adversarial training presented in Tables 5.2 and 5.3 showed an increase in overall performance on the target domain for all of the defined evaluation metrics (w.r.t research question 2). Adaptation can be achieved just by labelling the collected data at a new location as the target domain (domain label).

Figure 6.1: Proposed workflow for adapting the environmental sound classifier when deploying an acoustic sensor node at a new location - although ground-truth class labels for the collected target domain data are not required during domain adversarial training, it is recommended to set aside a small subset of labeled target domain dataset for testing.

Therefore, adapting to a new location can be realised without the need to provide class labels for the collected target domain data (w.r.t research question 3). It is still, however, recommended to provide ground-truth labels for a small subset of the target domain data for evaluation purposes. Testing with an independent (unseen) labeled target domain dataset can ensure that the classifier has fulfilled adaptation to the new deployed location after adversarial training. In response to research question 4, we have proposed a workflow for scaling up WASN-ESC setups and presented it in Figure 6.1. Figure 6.1 illustrates the downstream process on how we can adapt the underlying environmental sound classifier when deploying a sensor node at a new location. Finally, following Figure 6.1, our proposed workflow can be repeatable for every new sensor node required at new locations. The following is a summarized list of major contributions of the work and thesis:

- A novel dataset of annotated sound events that are collected from deployed sensor nodes in residential areas in Singapore.
- Proposed a domain-adversarial training method that incorporates additional pre-training method and ablation study that is essential to achieve significant improvements for the target domain.
- Proposed an Inception variant network and pruning method that is able to keep model parameters at minimal and reduce the required computational power in a sensor node whilst maintaining accurate predictions.
- Proposed methodology for scalability of deep networks for different locations using unsupervised learning for adaptation purposes on WASN-ESC tasks.
- Proposed workflow and downstream process for adapting environmental sound classifiers for deploying acoustic sensors at new locations.

6.2 Future Work

We have shown improved performance on the target domain and addressed the domain shift issue influenced by novel environmental acoustics and recording devices; however, there are many future research directions given different problem statements. For example, our method would work well given similar class labels for both source and target domains, but it would not be directly applicable to a target domain that requires a different set of sound classes that are not within the pre-defined taxonomy. For that matter, we would require a semi-supervised domain adaptation approach, where few labeled target domain examples are available and essentially leveraging on the available labeled source domain examples that are similar to the newly defined target label. In this case, defining the taxonomy is crucial for effective adaptation, such that given sound classes that occur in a superset relationship (*i.e.* the relationship of “Vehicle” sound class (*source*) to “Truck” sound class (*target*)) or a subset relationship (*i.e.* the relationship of “Jack Hammer” sound class (*source*) to “Construction” sound class (*target*)) can be identically indexed in the list of source domain sound classes to the list of target

domain sound classes when we approach adaptation problem, given differences in the class label space.

The future work also includes further experimentations with other domain adaptation methods. As reviewed in Chapter 2, many applied spectrum correction techniques in DCASE 2019 Task1B (microphone domain adaptation). The addition of such augmentation in a downstream process pipeline for audio domain adaptation tasks may hold promising results. In recent years, there are different domain adaptation methods experimented in the literature of the computer vision field. Although many applied their methods only on image datasets, it is possible that these methods can translate into the audio field for adaptation tasks as well.

Finally, generative adversarial networks (GANs) can be explored in the situation of insufficient target domain data. In this case, GANs can provide a solution to learn the underlying representations of the target domain data and generate realistic target domain data to be utilized for further adversarial training.

Bibliography

- [1] H. Ritchie and M. Roser. “Urbanization”. In: *Our World in Data* (June 2018). URL: <https://ourworldindata.org/urbanization>.
- [2] M.S. Hammer, T.K. Swinburn, and R.L. Neitzel. “Environmental Noise Pollution in the United States: Developing an Effective Public Health Response”. In: *Environmental Health Perspectives* 122.2 (Feb. 2014), pp. 115–119. DOI: 10.1289/ehp.1307272. URL: <https://doi.org/10.1289/ehp.1307272>.
- [3] A. Bronzaft and G. Van Ryzin. “Neighborhood Noise and Its Consequences: Implications for Tracking Effectiveness of NYC Revised Noise Code”. In: (June 2019).
- [4] M. Basner et al. “Auditory and non-auditory effects of noise on health”. In: *The Lancet* 383.9925 (Apr. 2014), pp. 1325–1332. DOI: 10.1016/s0140-6736(13)61613-x. URL: [https://doi.org/10.1016/s0140-6736\(13\)61613-x](https://doi.org/10.1016/s0140-6736(13)61613-x).
- [5] H. Miedema et al. “Burden of disease from environmental noise – Quantification of healthy years lost in Europe”. en. In: *World Health Organization* (2011).
- [6] M.N. Mead. “The sound behind heart effects.” In: *Environmental health perspectives* 115.11 (Nov. 2007), A536–7.
- [7] K.D. Kryter, A.Z. Weisz, and F.M. Wiener. “Damage Risk Criterion for and Auditory Fatigue from the Audioc”. In: *The Journal of the Acoustical Society of America* 33.6 (1961), pp. 858–859.

- [8] S.A. Stansfeld and M.P. Matheson. “Noise pollution: non-auditory effects on health.” In: *British medical bulletin* 68 (2003), pp. 243–57.
- [9] L. Tzivian et al. “Effect of long-term outdoor air pollution and noise cognitive and psychological functions in adults”. en. In: *International Journal of Hygiene and Environmental Health* 218.1 (2015), pp. 1–11.
- [10] P.H.T. Zannin, F.B. Diniz, and W.A. Barbosa. “Environmental noise pollution in the city of Curitiba, Brazil”. en. In: *Applied Acoustics* 63.4 (Apr. 2002), pp. 351–358.
- [11] J.M.B. Morillas et al. “An environmental noise study in the city of Cáceres, Spain”. en. In: *Applied Acoustics* 63.10 (Oct. 2002), pp. 1061–1070.
- [12] K.T. Tsai, M.D. Lin, and Y.H. Chen. “Noise mapping in urban environments: A Taiwan study”. en. In: *Applied Acoustics* 70.7 (July 2009), pp. 964–972.
- [13] J.S. Garcia et al. “Low-cost alternatives for urban noise nuisance monitoring using wireless sensor networks”. en. In: *IEEE Sensors* 15.2 (Sept. 2014), pp. 836–844.
- [14] B. Ahlgren, M. Hidell, and E. Ngai. “Internet of things for smart cities: Interoperability and open data”. en. In: *IEEE Internet Computing* 20.6 (Nov. 2016), pp. 52–56.
- [15] A. Zanella et al. “Internet of things for smart cities”. en. In: *IEEE Internet of Things Journal* 1.1 (Feb. 2014), pp. 22–32.
- [16] B.N. Silva, M. Khan, and K. Han. “Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities”. en. In: *Sustainable Cities and Society* 38 (Apr. 2018), pp. 697–713.
- [17] F.A. Turjman and A. Malekloo. “Smart parking in IoT-enabled cities: A survey”. en. In: *Sustainable Cities and Society* 49 (Aug. 2019).
- [18] I. Chatzigiannakis, A. Vitaletti, and A. Pyrgelis. “A privacy-preserving smart parking system using an IoT elliptic curve based security platform”. en. In: *Computer Communication Vols.* 89-90 (Sept. 2016), pp. 165–177.

- [19] D. Jin et al. “Smart street lighting system: A platform for innovative smart city applications and a new frontier for cyber-security”. en. In: *The Electricity Journal* 29.10 (Dec. 2016), pp. 28–35.
- [20] Y.C. Chang and Y.H. Lai. “Campus Edge Computing Network Based on IoT Street Lighting Nodes”. en. In: *IEEE Systems Journal* 14.1 (Mar. 2020), pp. 164–171.
- [21] P. Rizwan, K. Suresh, and M.R. Babu. “Real-time smart traffic management system for smart cities by using Internet of Things and big data”. en. In: *2016 International Conference on Emerging Technological Trends (ICETT)*. Oct. 2016, pp. 1–7.
- [22] A. Sharif et al. “Internet of things — smart traffic management system for smart cities using big data analytics”. en. In: *14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. Dec. 2017, pp. 281–284.
- [23] D. Minoli, K. Sohraby, and B. Occhiogrosso. “IoT Considerations, Requirements, and Architectures for Smart Buildings—Energy Optimization and Next-Generation Building Management Systems”. en. In: *IEEE Internet of Things Journal* 4.1 (Feb. 2017), pp. 269–283.
- [24] A.P. Plageras et al. “Efficient IoT-based sensor BIG Data collection–processing and analysis in smart buildings”. en. In: *Future Generation Computer System* 82 (May 2018), pp. 349–357.
- [25] T. Anagnostopoulos et al. “Challenges and Opportunities of Waste Management in IoT-Enabled Smart Cities: A Survey”. en. In: *IEEE Transactions on Sustainable Computing* 2.3 (July 2017), pp. 275–289.
- [26] Z. Wen et al. “Design, implementation, and evaluation of an Internet of Things (IoT) network system for restaurant food waste management”. en. In: *Waste Management* 73 (Mar. 2018), pp. 26–38.

- [27] C. Mydlarz, J. Salamon, and J.P. Bello. “The implementation of low-cost urban acoustic monitoring devices”. en. In: *Applied Acoustics* 117 (Feb. 2017), pp. 207–218.
- [28] J.P. Bello et al. “SONYC: A System for Monitoring, Analyzing, and Mitigating Urban Noise Pollution”. In: *Communications of the ACM* 62.2 (Feb. 2019), pp. 68–77. DOI: 10.1145/3224204.
- [29] J. AbeBer and C. Kuhn. “A distributed sensor network for monitoring noise level and noise sources in urban environment”. en. In: *IEEE 6th International Conference on Future Internet of Things and Cloud*. Aug. 2018, pp. 1–8.
- [30] J. AbeBer et al. “Urban noise monitoring in the Stadtlarm project – A field report”. en. In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. Oct. 2019, pp. 1–4.
- [31] F. Alias et al. “Aggregate Impact of Anomalous Noise Events on the WASN-Based Computation of Road Traffic Noise Levels in Urban and Suburban Environments”. In: *Sensors* 20.3 (2020), p. 609.
- [32] J.A. Fernandez-Prieto, J. Bago, and M.A. Gadeo-Martos. “Wireless Acoustic Sensor Nodes for Noise Monitoring in the City of Linares (Jaén)”. In: *Sensors* 20 (Dec. 2019), p. 124. DOI: 10.3390/s20010124.
- [33] J.S. Garcia et al. “Spatial Statistical Analysis of Urban Noise Data from a WASN Gathered by an IoT System: Application to a Small City”. In: *Applied Sciences* 6 (Nov. 2016), p. 380. DOI: 10.3390/app6120380.
- [34] *IEEE Signal Processing Society - Audio and Acoustic Signal Processing Challenges*. [Online] Accessed: 2021-02-01. URL: <https://signalprocessingsociety.org/community-involvement/audio-and-acoustic-signal-processing/aasp-challenges>.
- [35] *Detection and Classification of Acoustic Scenes and Events*. [Online] Accessed: 2021-02-01. URL: <http://dcase.community/>.

- [36] *Acoustic Scene Classification*. [Online] Accessed: 2021-02-01. URL: <http://dcase.community/challenge2019/task-acoustic-scene-classification/>.
- [37] *Urban Sound Tagging*. [Online] Accessed: 2021-02-01. URL: <http://dcase.community/challenge2019/task-urban-sound-tagging/>.
- [38] M. Cartwright et al. *SONYC Urban Sound Tagging (SONYC-UST): a multilabel dataset from an urban acoustic sensor network*. Version 0.2.0. This work is supported by National Science Foundation award 1544753. May 2019. DOI: 10.5281/zenodo.3233082. URL: <https://doi.org/10.5281/zenodo.3233082>.
- [39] J. Salamon, C. Jacoby, and J.P. Bello. “A Dataset and Taxonomy for Urban Sound Research”. In: *22nd ACM International Conference on Multimedia (ACM-MM’14)*. Orlando, FL, USA, Nov. 2014, pp. 1041–1044.
- [40] K.J. Piczak. *ESC: Dataset for Environmental Sound Classification*. Version V2. 2015. DOI: 10.7910/DVN/YDEPUT. URL: <https://doi.org/10.7910/DVN/YDEPUT>.
- [41] D. Stowell and M.D. Plumbley. “An open dataset for research on audio field recording archives: freefield1010”. en. In: *53rd International Conference: Semantic Audio*. Jan. 2014, pp. 1–6.
- [42] F. Font, G. Roma, and X. Serra. “Freesound Technical Demo”. In: *ACM International Conference on Multimedia (MM’13)*. ACM. Barcelona, Spain: ACM, 21/10/2013 2013, pp. 411–412. ISBN: 978-1-4503-2404-5. DOI: 10.1145/2502081.2502245.
- [43] A. Mesaros, T. Heittola, and T. Virtanen. *TUT Acoustic scenes 2016, Development dataset*. Feb. 2016. DOI: 10.5281/zenodo.45739. URL: <https://doi.org/10.5281/zenodo.45739>.
- [44] B.D. Coensel and D. Botteldooren. “Smart sound monitoring for sound event detection and characterization”. en. In: *Proceedings of 43rd International Congress on Noise Control Engineering*. Nov. 2014.

- [45] *Audio Analytic Company*. [Online] Accessed: 2021-02-01. URL: <https://audioanalytic.com>.
- [46] I. Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.
- [47] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain”. In: *Psychological Review* (1958), pp. 65–386.
- [48] D. Rumelhart, G. Hinton, and R.J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.
- [49] G. Hinton et al. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97.
- [50] A. Krizhevsky, I. Sutskever, and G. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [51] N.P. Jouppi et al. “In-datacenter performance analysis of a tensor processing unit”. In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. 2017, pp. 1–12.
- [52] H. Purwins et al. “Deep Learning for Audio Signal Processing”. In: *CoRR* abs/1905.00078 (2019). arXiv: 1905.00078. URL: <http://arxiv.org/abs/1905.00078>.
- [53] T. Virtanen, M. Plumbley, and D. Ellis. *Computational Analysis of Sound Scenes and Events*. Sept. 2017, pp. 1–422. DOI: 10.1007/978-3-319-63450-0.
- [54] K. J. Piczak. “Environmental sound classification with convolutional neural networks”. In: *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2015, pp. 1–6.
- [55] V. Peltonen et al. “Computational auditory scene recognition”. In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2. IEEE. 2002, pp. II–1941.

- [56] C. Clavel, T. Ehrette, and G. Richard. “Events detection for an audio-based surveillance system”. In: *2005 IEEE International Conference on Multimedia and Expo*. IEEE. 2005, pp. 1306–1309.
- [57] S. Chu, S. Narayanan, and C.C.J. Kuo. “Environmental sound recognition with time–frequency audio features”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.6 (2009), pp. 1142–1158.
- [58] Z. Kons and O. Toledo-Ronen. “Audio event classification using deep neural networks”. In: Jan. 2013, pp. 1482–1486.
- [59] J. Li et al. “A Comparison of deep learning methods for environmental sound”. In: *CoRR* abs/1703.06902 (2017). arXiv: 1703.06902. URL: <http://arxiv.org/abs/1703.06902>.
- [60] S. Davis and P. Mermelstein. “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences”. In: *IEEE transactions on acoustics, speech, and signal processing* 28.4 (1980), pp. 357–366.
- [61] J. Salamon and J.P. Bello. “Deep convolutional neural networks and data augmentation for environmental sound classification”. en. In: *IEEE Signal Processing Letters* 24.3 (Mar. 2017), pp. 279–283.
- [62] M. Huzaifah. *Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks*. 2017. arXiv: 1706.07156 [cs.CV].
- [63] A. Politis, S. Adavanne, and T. Virtanen. *A Dataset of Reverberant Spatial Sound Scenes with Moving Sources for Sound Event Localization and Detection*. 2020. arXiv: 2006.01919 [eess.AS].
- [64] J. Cramer et al. “Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2019, pp. 3852–3856. DOI: 10.1109/ICASSP.2019.8682475.

- [65] J. Salamon, C. Jacoby, and J.P. Bello. “A dataset and taxonomy for urban sound research”. en. In: *22nd ACM International Conference on Multimedia*. Nov. 2014.
- [66] P. Bellucci, L. Peruzzi, and G. Zambon. “LIFE DYNAMAP project: The case study of Rome”. en. In: *Applied Acoustics* 117 (Feb. 2017), pp. 193–206.
- [67] P. Bellucci, L. Peruzzi, and G. Zambon. “LIFE DYNAMAP: An overview of the project after two years working”. en. In: *Inter-Noise and Noise-Con Congress and Conference Proceedings*. Aug. 2016, pp. 6338–6349.
- [68] P. Bellucci, L. Peruzzi, and G. Zambon. “LIFE DYNAMAP: Making dynamic noise maps a reality”. en. In: *Inter-Noise and Noise-Con Congress and Conference Proceedings*. Aug. 2016, pp. 1181–1188.
- [69] D. Scherer, A. Müller, and S. Behnke. “Evaluation of pooling operations in convolutional architectures for object recognition”. In: *International conference on artificial neural networks*. Springer. 2010, pp. 92–101.
- [70] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2015).
- [71] C. Szegedy et al. “Going Deeper with Convolutions”. In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: <http://arxiv.org/abs/1409.4842>.
- [72] K. He et al. “Deep Residual Learning for Image Recognition”. In: (2015). arXiv: 1512.03385 [cs.CV].
- [73] S. Hershey et al. “CNN Architectures for Large-Scale Audio Classification”. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. URL: <https://arxiv.org/abs/1609.09430>.
- [74] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

- [75] J.F. Gemmeke et al. “Audio Set: An ontology and human-labeled dataset for audio events”. In: *Proc. IEEE ICASSP 2017*. New Orleans, LA, 2017.
- [76] H. Zhang et al. “mixup: Beyond Empirical Risk Minimization”. In: *CoRR* abs/1710.09412 (2017). arXiv: 1710.09412. URL: <http://arxiv.org/abs/1710.09412>.
- [77] Z. Zhong et al. *Random Erasing Data Augmentation*. 2017. arXiv: 1708.04896 [cs.CV].
- [78] D.S. Park et al. “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition”. In: *Interspeech 2019* (Sept. 2019). DOI: 10.21437/interspeech.2019-2680. URL: <http://dx.doi.org/10.21437/Interspeech.2019-2680>.
- [79] E.S. Olivas et al. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*. IGI Global, 2009.
- [80] J. Yosinski et al. “How transferable are features in deep neural networks?”. In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.
- [81] Y. Aytar, C. Vondrick, and A. Torralba. “Soundnet: Learning sound representations from unlabeled video”. In: *Advances in neural information processing systems*. 2016, pp. 892–900.
- [82] R. Arandjelovic and A. Zisserman. “Look, listen and learn”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 609–617.
- [83] *Flickr Company*. [Online] Accessed: 2021-02-01. URL: <http://www.flickr.com>.
- [84] S. Abu-El-Haija et al. “Youtube-8m: A large-scale video classification benchmark”. In: *arXiv preprint arXiv:1609.08675* (2016).

- [85] S.J. Pan and Q. Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.
- [86] M.T. Rosenstein et al. “To transfer or not to transfer”. In: *NIPS 2005 workshop on transfer learning*. Vol. 898. 2005, pp. 1–4.
- [87] *Raspberry Pi 3 Model B Plus*. [Online] Accessed: 2021-01-15. URL: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus>.
- [88] J.C. Socoro et al. “Adapting an Anomalous Noise Events Detector for Real-Life Operation in the Rome Suburban Pilot Area of the DYNAMAP’s Project”. en. In: *11th European Congress and Exposition on Noise Control Engineering*. May 2018, pp. 685–692.
- [89] F. Alias and J.C. Socoro. “Description of anomalous noise events for reliable dynamic traffic noise mapping in real-life urban and suburban soundscapes”. en. In: *Applied Science* 2.146 (Feb. 2017), p. 7.
- [90] S. Ben-David et al. “A theory of learning from different domains”. In: *Machine learning* 79.1-2 (2010), pp. 151–175.
- [91] K. Drossos, P. Magron, and T. Virtanen. *Unsupervised Adversarial Domain Adaptation Based On The Wasserstein Distance For Acoustic Scene Classification*. 2019. arXiv: 1904.10678 [cs.SD].
- [92] Y. Ganin et al. *Domain-Adversarial Training of Neural Networks*. 2016. arXiv: 1505.07818 [stat.ML].
- [93] E. Tzeng et al. “Adversarial Discriminative Domain Adaptation”. In: *CoRR* abs/1702.05464 (2017). arXiv: 1702.05464. URL: <http://arxiv.org/abs/1702.05464>.
- [94] G. French, M. Mackiewicz, and M.H. Fisher. “Self-ensembling for domain adaptation”. In: *CoRR* abs/1706.05208 (2017). arXiv: 1706.05208. URL: <http://arxiv.org/abs/1706.05208>.

- [95] P. Russo et al. “From source to target and back: symmetric bi-directional adaptive GAN”. In: *CoRR* abs/1705.08824 (2017). arXiv: 1705.08824. URL: <http://arxiv.org/abs/1705.08824>.
- [96] Y. LeCun and C. Cortes. “MNIST handwritten digit database”. In: (2010). URL: <http://yann.lecun.com/exdb/mnist/>.
- [97] *The Street View House Numbers (SVHN) Dataset*. <http://ufldl.stanford.edu/house-numbers/>. Access: 2021-01-15.
- [98] J. J. Hull. “A database for handwritten text recognition research”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.5 (1994), pp. 550–554. DOI: 10.1109/34.291440.
- [99] M. Kośmider. *Calibrating Neural Networks for Secondary Recording Devices*. Tech. rep. DCASE2019 Challenge, June 2019.
- [100] *Acoustic Scene Classification with mismatched recording devices*. [Online] Accessed: 2021-02-01. URL: <http://dcase.community/challenge2019/task-acoustic-scene-classification-results-b>.
- [101] W. Wei et al. “A-CRNN: A Domain Adaptation Model for Sound Event Detection”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 276–280.
- [102] I. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].
- [103] E.L. Tan et al. “Extracting Urban Sound Information for Residential Areas in Smart Cities Using an End-to-End IoT System”. In: *IEEE Internet of Things Journal* (2021), pp. 1–1. DOI: 10.1109/JIOT.2021.3068755.
- [104] *Adafruit I2S MEMs Microphone - SPH0645LM4H*. [Online] Accessed: 2021-01-15. URL: <https://www.adafruit.com/product/3421>.
- [105] *MiniDSP UMA-8 USB mic array - V2.0*. [Online] Accessed: 2021-01-15. URL: <https://www.minidsp.com/products/usb-audio-interface/uma-8-microphone-array>.

- [106] *Audacity® software*. Audacity® software is copyright © 1999-2021 Audacity Team. Web site: <https://audacityteam.org>. It is free software distributed under the terms of the GNU General Public License. The name Audacity® is a registered trademark.
- [107] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: <http://arxiv.org/abs/1502.03167>.
- [108] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *CoRR* abs/1912.01703 (2019). arXiv: 1912.01703. URL: <http://arxiv.org/abs/1912.01703>.
- [109] L. van der Maaten and G. Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: <http://jmlr.org/papers/v9/vandermaaten08a.html>.