POLYTECHNIC UNIVERSITY OF TURIN
EURECOM UNIVERSITY

Master's Degree in Data Science and Engineering

Politecnico di Torino

Master's Degree Thesis

Low-complexity neural networks for robust acoustic scene classification in wearable audio devices

Academic supervisors

Prof. ANTONIO SERVETTI

Prof. PIETRO MICHIARDI

Industrial supervisors
in Bang & Olufsen

Dr. SVEN EWAN SHEPSTONE

Dr. PABLO MARTINEZ-NUEVO

Candidate

MICHELE PANARIELLO

S276977

April 2022

Abstract

This work concerns the design of a machine learning pipeline to perform acoustic scene classification (ASC) on a pair of headphones by means of a convolutional neural network (CNN). ASC is the task of recognizing which scenery surrounds the device (e.g. bus, park, home) using the audio captured by its microphone; the goal is to make the headphones context-aware to enhance user experience.

A challenging aspect of the task is the lack of recordings coming from the microphone of the headphones, which forces us to resort to external data sources: training on audio acquired from a different microphone may cause a data distribution shift and impact the classification performance (a phenomenon known as *device mismatch*). Moreover, because of the embedded environment, it is only possible to use a CNN of low complexity, which may be limiting in terms of modeling accuracy.

We define the set of acoustic scenes to classify by seeking balance among the neural network's capabilities, the possible use cases of the product, and what labeled data is publicly available. We assess the impact of device mismatch by re-recording a part of our training set and testing our model on it; we establish that no sensible performance degradation occurs. We then devise a technique to simulate the availability of further data from the target microphone.

We test the generalization capabilities of the neural network to various kinds of input perturbations such as wind, unseen acoustic scenes and reverberations. We propose a number of approaches to make the model more robust to said perturbations, including different data augmentation techniques and the integration of a hidden Markov model in the system. We show that there is a fundamental trade-off between perturbation robustness and classification performance.

We attempt to bypass that trade-off by increasing the complexity of the model. Our experiments suggest that merely deepening the network does not lead to sensible improvements; instead, the preprocessing is the real performance bottleneck of the system. With a more fine-grained input audio representation, we are able to enhance classification performance and robustness simultaneously, albeit at the cost of higher memory usage.

Acknowledgements

This thesis was carried out at the Acoustics & Research department of the Bang & Olufsen Innovation Lab in Struer, Denmark. The candidate would like to thank all the members of the department that helped shape this work with their expertise, professionalism and continuous support. Special thanks to Sven and Pablo for their supervision and guidance throughout the project.

Table of Contents

List of Tables

List of Figures

Acronyms

AI artificial intelligence

ALP adversarial logit pairing

ASC acoustic scene classification

B&O Bang & Olufsen

BN batch normalization

CNN convolutional neural network

DCASE detection and classification of acoustic scenes and events

DL deep learning

ERM empirical risk minimization

FFT fast Fourier transform

HATS head and torso simulator

HMM hidden Markov model

IoT internet of things

LTI linear time invariant

MFCC Mel-frequency cepstral coefficient

ML machine learning

OOD out-of-distribution

ReLU rectified linear unit

RMS root mean square

SGD stochastic gradient descent

SNR signal-to-noise ratio

SSN subspectral normalization

Chapter 1

Introduction and fundamentals

The union of linear algebra, data analysis and numerical optimization known as machine learning has tiptoed into several areas of engineering and technology. From a technical perspective, this is due to the apparently ever-increasing availability of two main elements: the computing power of machines, and data about countless aspects of our lives. From a scientific perspective, being able to model complex phenomena simply by showing a certain number of examples to a (mostly) general-purpose algorithm is intriguing, tempting, and potentially time-saving: one can spare the effort of manually assembling a model by means of domain-specific knowledge and just let a machine infer it automatically.

Indeed, machine learning methods are applicable to a wide variety of data types, both structured (tabular data) and unstructured (highly dimensional inputs such as images, audio, time series). For this reason, these algorithms have made their way into an equally wide range of applications: language translation, speech recognition, predictive maintenance would be popular examples. Less common ones may include database query optimization, satellite image analysis, lecture summarization, animal healthcare monitoring, and just about any imaginable niche. This shows how pervasive machine learning has become, and how flexible and powerful those problem-agnostic models can be when combined with the right data.

In this work, we describe how to combine machine learning with one further specific area: context-awareness of wearable audio devices. We present a case study in which we design an algorithm to make a pair of headphones understand what "acoustic scenario" their user is immersed in by analyzing the audio input captured by their microphone: in the scientific literature, this task is known as *acoustic scene classification*. We explore the challenges faced in formalizing the problem; the constraints imposed by the production environment; the iterative process of

optimizing our system, discovering its shortcomings, and optimizing it again. The outcome of our work is a thorough benchmark of the model's capabilities in several scenarios and the investigation of what techniques can be used to improve its robustness. With that, we lay the foundation of what will be further developed into a suitable system to be deployed in a commercial product.

In this Chapter, we give a general introduction to machine learning and the task of acoustic scene classification.

In Chapter 2, we provide a more detailed description of the goal of our work, its main challenges, and the starting point of the project.

In Chapter 3, we define a suitable set of scenes to classify, we provide a baseline measurement of how well the model can distinguish them, and how much data is needed to train it until its predictive power reaches saturation.

In Chapter 4, we describe the differences between the embedded production environment and the prototyping Python-based environment, and what we can do in attempt to unify them.

In Chapter 5, we explore the trade-offs in making the model robust against various kinds of perturbations without modifying the basic steps of the pipeline where the audio input is preprocessed and classified.

In Chapter 6, we relax the requirement of keeping the preprocessing and classifier fixed and assess the consequent impact on the overall model's performance.

In Chapter 7, we draw the conclusions of our work, and systematically suggest what research directions and sub-fields may be worth investigating for future developments of the acoustic scene classification system.

1.1 Machine learning, deep learning, neural networks

1.1.1 The machine learning mathematical framework

Machine learning (ML) broadly describes the task of extracting relevant knowledge from sets of data by means of machine-automated algorithms. The data represents "experience" for the machine, which has to generate a parametrized algorithm that matches the patterns "learned" from the data [83].

Formally, a ML problem can be framed as the union of four main elements: a dataset, a model, a risk function, and an optimization technique. To better contextualize their role, we report a practical example for each of them in the context of designing a system that must recognize the identity of a certain person from a picture of their face. Under this premise, a machine learning problem is composed of the following:

- A **set of data** (or **dataset**) D related to the task at hand (e.g. pictures of human faces and the identity associated to each of them);
- A **model** (or **hypothesis**) h that depends on a set of parameters θ and describes the phenomenon of interest (e.g. the set of operations that map the picture of a face to a certain identity);
- A **risk function** $\mathcal{R}$ that numerically quantifies the divergence between $h(\theta)$ and the reality described by D (e.g. how many faces in the dataset are wrongly identified);
- An **optimization algorithm** $\mathcal{A}$ that tunes the parameters θ in order to minimize the risk $\mathcal{R}$ and make the model more faithful to reality (e.g. perform changes to the algorithm until the amount of misidentified faces is lower than a certain threshold). This is what represents the process of *learning*; for this reason, the set θ is said to contain the *learnable parameters* of the model.

This work operates in the realm of *supervised learning*. Supervised learning is a branch of ML where the dataset D is of the form

$$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \tag{1.1}$$

with $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. The couple $(\mathbf{x}, y)$ represents the pair of a model input $\mathbf{x}$ and a desired model output y. This is often referred to as a *data point*. Input-output pairs are assumed to be related by an unknown function $f(\mathbf{x}) = y$. In supervised learning, the model $h(\theta)$ aims to approximate the unknown function f: thus, it takes on the form of a parametric function $h : \mathcal{X} \to \mathcal{Y}$ such that $h(\theta; \mathbf{x}) \approx f(\mathbf{x})$. The "supervision" comes from the fact that the model is instructed to reproduce a known desired output y; this is not the case in other branches of ML such as *unsupervised learning*.

Both input and output domain $\mathcal{X}$ and $\mathcal{Y}$ may either be numerical or categorical. If $\mathcal{Y}$ is categorical, the task is called *classification* and a value y is called *label*; if $\mathcal{Y}$ is numerical, the task is called *regression* [105]. The model h is selected from a class of functions $\mathcal{H}$ called *hypothesis class*. The risk $\mathcal{R}$ is defined as the expected value of a *loss function* $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that quantifies how much the output of the model (*prediction*) differs from the true expected output (*ground truth*):

$$\mathcal{R} \triangleq \mathbb{E}_{\mathcal{X}\times\mathcal{Y}} \left[\mathcal{L} \left(h(\theta; \mathbf{x}), y \right) \right] \tag{1.2}$$

A theoretically relevant quantity in the presented framework is the Bayes risk $\mathcal{R}^*$, defined as the infimum of the achievable risk over all possible hypotheses:

$$\mathcal{R}^* \triangleq \inf_h \mathcal{R}\left(h\right) \tag{1.3}$$

Since the true risk as defined in (1.2) cannot usually be analytically computed, it is often approximated with the *empirical risk*, i.e. the average loss computed over

all dataset points. It is then possible to formally state a supervised classification task as an *empirical risk minimization* (ERM) problem:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{|D|} \sum_{(\mathbf{x}_i, y_i) \in D} \mathcal{L}\left(h(\theta; \mathbf{x}_i), y_i\right) \tag{1.4}$$

whose solution θ^* is usually employed to parametrize the final model $h(\theta^*, \mathbf{x})$. In many cases, such a solution cannot be computed in closed form, and it is necessary to resort to numerical optimization techniques to estimate it; however, according to the choice of $h(\cdot, \cdot)$ and $\mathcal{L}(\cdot, \cdot)$, the objective function of (1.4) can be highly non-convex, and therefore have several local minima. This makes the optimization procedure more challenging: for this reason, the goal of a machine learning task is often not to find *the* minimizer θ^*, but rather *one* good minimizer among the many existing ones.

1.1.2 Deep learning and neural networks

Among the numerous branches of machine learning, *deep learning* (DL) has seen great success in many different areas of application, including acoustic scene classification. Deep learning is a sub-field of machine learning where the hypothesis class $\mathcal{H}$ is the set of functions represented by *neural networks*. A neural network model with N *layers* can be described as a chain of N parametric transformations applied to an input vector $\mathbf{x} \in \mathbb{R}^D$ to produce an output $\mathbf{u} \in \mathbb{R}^C$:

$$\left(f_{\theta_N}^{(N)} \circ f_{\theta_{N-1}}^{(N-1)} \circ \cdots \circ f_{\theta_1}^{(1)}\right)(\mathbf{x}) = \mathbf{u} \tag{1.5}$$

Each function f represents a *layer*. In *fully-connected neural networks* (also known as *feedforward neural networks*, *dense networks* or *multi-layer perceptrons*), each layer is usually of the form

$$\begin{array}{rccl} f_{\mathbf{W},\mathbf{b}}(\mathbf{x}) : \mathbb{R}^D & \longrightarrow & \mathbb{R}^{D'} \\ \mathbf{x} & \longmapsto & g\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \end{array} \tag{1.6}$$

Where:

- the *weight* matrix $\mathbf{W} \in \mathbb{R}^{D' \times D}$ and the bias vector $\mathbf{b} \in \mathbb{R}^{D'}$ are within the set of the learnable parameters θ;
- $g(\cdot)$ is a nonlinear function, oftentimes the *rectified linear unit* (ReLU), defined as $g(\mathbf{x}) = \max\{\mathbf{0}, \mathbf{x}\}$ and applied pointwise to each vector element.[1]

[1] Other possible activation functions used in practice include $\tanh(\mathbf{x})$ or $\sigma(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$. However, ReLU and its variants are currently the most popular choice [89].

A fully-connected neural network architecture can quickly become cumbersome in terms of memory usage, as a layer $f_{\mathbf{W},\mathbf{b}}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^{D'}$ has $D \times D' + D'$ parameters that need to be stored. This number can rapidly grow if the model must process large amounts unstructured data, such as images or digital audio. Partly because of that, fully-connected neural nets have been outclassed by *convolutional neural network*s (CNNs), a neural network topology that replaces the linear projection in a layer with a convolution operation:

$$f_{\mathbf{W},\mathbf{B}}(\mathbf{X}) = g\left(\mathbf{W} * \mathbf{X} + \mathbf{B}\right) \tag{1.7}$$

where the input $\mathbf{X}$, the $\mathbf{W}$ operator and the output typically take the form of 3-dimensional tensors. $\mathbf{W}$ is usually of much smaller dimension than $\mathbf{X}$, which allows the network to be more lightweight than a normal fully-connected topology. This approach has been a well-established trend in computer vision tasks for quite a long time [65], and has later been experimented with in the audio domain [1] as well. The operation in (1.7) is sometimes followed by a *pooling* operator that reduces the resolution of the output tensor in the first two dimensions by replacing certain regions of the input with a scalar summary of their values. The goal is twofold: first, the network learns to compress the information contained in the input signal into a lower dimensional space; second, this makes the model invariant to small translations in the input signal that happen within the boundaries of the summarized regions. A popular pooling operation is *max pooling*, in which the summarization consists in applying the max operation to $K \times K$ 2-dimensional, non-overlapping patches of the input tensor. This is typically done independently along the third dimension, whose resolution is left untouched. A max pooling operation across $K \times K$ patches may be expressed as:

$$\text{MaxPool}(\mathbf{X})_{ijc} = \max_{\substack{i' \in \{Ki-K+1, Ki-K+2 \ldots Ki\} \\ j' \in \{Kj-K+1, Kj-K+2 \ldots Kj\}}} \mathbf{X}_{i'j'c} \tag{1.8}$$

The max operation in the equation above is sometimes replaced with the arithmetic mean operation, obtaining what is known as *average pooling*.

In the case of neural networks, the empirical risk minimization problem stated in (1.4) is usually solved by means of stochastic gradient descent (SGD), which optimizes the parameters of the neural network iteratively by means of *mini-batches* of the available training data. Let $\mathcal{B} \in D$ be a randomly drawn subset (i.e. a mini-batch) of the training dataset D; let $\bar{\mathbf{W}} = [\mathbf{W}_1 \ldots \mathbf{W}_K]$ be a vector containing the concatenation of the network's convolution operations; let $\text{net}(\bar{\mathbf{W}}; \mathbf{x})$ be the prediction produced by the network for a given input $\mathbf{x}$. Then each step of SGD performs the following update:

$$\bar{\mathbf{W}} \leftarrow \bar{\mathbf{W}} - \frac{\lambda}{|\mathcal{B}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{B}} \nabla_{\bar{\mathbf{W}}} \mathcal{L}(\text{net}(\bar{\mathbf{W}}; \mathbf{x}_i), y_i) \tag{1.9}$$

The bias terms $\bar{\mathbf{B}} = [\mathbf{B}_1 \dots \mathbf{B}_K]$ are optimized in a similar way. This procedure is referred to as "training" and continues until the parameters of the model have converged. The scalar λ is an arbitrary scalar called *learning rate* that controls how much of a numerical impact each iteration has on the optimized parameters.

Occasionally, the loss term $\mathcal{L}$ may be combined with an additional term that applies numerical constraints to the network's weights in order to purposefully limit its complexity. One possible approach is to penalize the optimization algorithm if it leads to the network having high-magnitude weights: this can be done by adding the L^2-norm of the concatenated weights to the normal loss function:

$$\mathcal{L}' = \mathcal{L} + \alpha \left\| \mathbf{W} \right\|^2 \tag{1.10}$$

where α is a scalar hyperparameter that controls how much the penalty impacts the optimization procedure. SGD can then be applied by substituting $\mathcal{L}$ with $\mathcal{L}'$ in (1.9). This operation is known as L^2 *regularization* or *weight decay* and has been shown to improve the generalization capabilities of deep neural networks [66].

In classification tasks, the output of the neural network is usually a number of scalars N equal to the number of classes to classify. This output size is obtained by progressively scaling down the resolution of the input tensor by means of convolutional layers, fully-connected layers and pooling operations. Those N coefficients are often referred to as *logits*. The logits vector $\mathbf{g} \in \mathbb{R}^N$ is often converted to a probability vector $\mathbf{p}$ of the same dimension by applying a *softmax* operation to it:

$$\mathbf{p}_i = \frac{e^{\mathbf{g}_i}}{\sum_{j=1}^{N} e^{\mathbf{g}_j}} \tag{1.11}$$

The class prediction $\hat{y}$ is then usually performed by choosing the predicted class with highest probability:[2]

$$\hat{y} = \underset{i=1\dots N}{\operatorname{argmax}} \, \mathbf{p}_i \tag{1.12}$$

In this setting, a typical loss function for classification tasks is the *negative log-likelihood* loss or *cross-entropy* loss, where one encourages the network to maximize the probability of the true class of a sample by minimizing its negative logarithm:

$$\mathcal{L}\left(\mathbf{x}, y\right) = -\log \mathbf{p}_i \big|_{i=y} \tag{1.13}$$

Assuming that y is a numerical encoding of the true class.[3]

[2]Because of the properties of the softmax operation, it would technically be equivalent to perform the prediction by taking $\hat{y} = \operatorname{argmax}_{i=1\dots N} \mathbf{g}_i$. However, normalizing the logits as probabilities makes them more easily interpretable and is necessary for certain loss functions to be computed.

[3]The term *cross-entropy* derives from the fact that the loss can be interpreted as a cross-entropy

1.2 Acoustic scene classification

Acoustic scene classification (ASC) is the task of automatically establishing which environment produced a given acoustic signal. An environment may correspond to a physical location, such as "office" or "train", or a high-level acoustic description such as "quiet" or "noisy". These environments are denoted *acoustic scenes* [15].

ASC is often compared to the similar task of sound event detection. While the former refers to the identification of a scene that could include multiple sound sources (such as footsteps and chatter in a "public square" scene), the latter usually aims to identify sound coming from a single source (such as a doorbell or the passing of a car), often attempting to obtain specific timestamps that describe when the acoustic event takes place [52].

One of the most promising applications of ASC is the possibility of endowing portable devices with acoustic context awareness and sound-based monitoring of specific environments. This idea dates back to at least two decades ago [108], but has since gained popularity thanks to the rise of IoT [73, 26]. Other potential use cases of ASC include urban noise monitoring [2] and enhancement of scene recognition capabilities of robotic systems [20].

While it surely does not lack potential use cases, acoustic scene classification was shown to be challenging even for humans: the study described in [93] reports that, when given the task of distinguishing between a set of 25 acoustic scenes, human annotators achieved approximately 70% accuracy. By contrast, human accuracy on vast image datasets such as ImageNet was estimated to be around 95% [104]. Thus, in general, it is also reasonable to expect the task of acoustic scene classification to be more challenging than others within a machine learning framework.

1.2.1 Machine learning approaches to acoustic scene classification

Ever since one of its earliest known appearances in scientific literature, ASC was tackled by means of ML-based methods [108, 15]. This is likely due to the difficulty in hand-crafting a set of rules that can accurately describe something as complex as an acoustic scene: as previously mentioned, one of the perks of machine learning is to automatically infer classification criteria by relying on little to no human knowledge. A typical acoustic scene classification pipeline follows the same pattern as many audio-based machine learning systems: it is usually composed of a front-end preprocessing phase and a back-end classification algorithm.

---

operation applied between the probability vector $\mathbf{p}_i$ produced by the network and a ground-truth vector $\mathbf{y}$ that has value 1 in the component whose index matches the true class' index and 0 elsewhere: $\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\sum_{i=1}^{N} \mathbf{y}_i \log \mathbf{p}_i$.

Most typical preprocessing techniques involve time-frequency representations of the input signal such as the magnitude or the power of its short-time Fourier transform [27]. A commonly adopted approach is then to extract informative coefficients from the spectral representation by using filter banks: among those, Mel filter banks seem to be a popular choice [108, 21]. Some systems perform the further step of computing *cepstral* features from the filtered input, usually Mel-frequency cepstral coefficients [96, 52]. Other kinds of front-end preprocessing, unexplored in this work, are based on autoregressive approaches such as linear predictive coding [15].

Once features are computed, a back-end statistcal model is usually applied to perform the classification task. Support vector machines [22] are popular discriminative classifiers that have also been applied to acoustic scene classification [75, 36]. Gaussian mixture models for class density estimation have been widely employed in audio tasks in general [33, 55], including acoustic scene classification [11, 20]. When the temporal evolution of input features is deemed of relevance for the modeling task, hidden Markov models are often used [27, 21].

While fully-connected neural networks have sometimes been experimented with as back-end classifiers [120], most of the recent research efforts in the field of acoustic scene classification have been devoted to the use of convolutional networks, which have been shown to outperform previous classification approaches [106, 19, 118]. Occasionally, the use of recurrent networks has also been investigated [12], albeit not with equal success.

It should be pointed out that the rise in popularity of end-to-end neural network based classification pipelines has impacted the world of audio processing and started to blur the line between back-end and front-end steps. Recent research has stepped into the direction of directly processing "raw" waveform data, usually by means of neural networks employing 1-dimensional convolutions in the first layer. With this approach, the feature representation of the audio signal is directly learned from the input in its original form. This has been common practice for a long time in areas such as computer vision [65, 113]. The learned features are directly processed by the subsequent layers of the network, eventually producing the desired output in an end-to-end fashion. The weights of the 1-dimensional convolutional filters can either be completely learnable [90, 13, 48] or biased in ways that make sense in audio applications: for example, a popular choice is to use parametrized time-domain band pass filters [99, 100]. The end-to-end approach has achieved state of the art in a number of applications, including keyword spotting [5], speech anti-spoofing [35], and voice activity detection [68]. Some research has been done in applying end-to-end neural networks to acoustic scene classification tasks, obtaining decent results [50, 114]; however, most of the top-performing ASC systems still focus on time-frequency handcrafted representations [3].

Chapter 2

Problem statement

2.1 Goals of the research activity

This thesis was carried out within a wider research activity concerning ASC conducted at Bang & Olufsen. The general research direction of the work is to design a low-complexity algorithm that can perform ASC while running on the company's headphones. At the beginning of the thesis work, B&O provided the following:

- a CNN architecture chosen based on the memory and energy constraints of the headphones' hardware;
- a rough baseline of the CNN's capabilities on a number of toy ASC tasks of low complexity;
- a codebase to perform basic model training;
- a pipeline to deploy a set of trained model weights to a *BeoMusic* headphone[1] flashed with an experimental, customized firmware.

When the candidate joined the project, the final use case for providing ASC capabilities to a headphone was not defined; rather, B&O intended to explore the possibilities of acoustic scene classification and establish an ad-hoc application suitable to the degree of effectiveness of the model. While a functioning basic pipeline for training and deployment was already present in the codebase of the company, the model's capabilities had only been experimented with on dummy problems in a prototyping TensorFlow environment. The candidate was then tasked to turn the dummy proof-of-concept into an effective system suitable for a production environment. Specifically, the candidate's contributions included:

[1] *BeoMusic* is an arbitrary working name and is not associated to any commercially available Bang & Olufsen product.

- establishing a set of acoustic scenes that are within the model's capabilities and also make sense from a use case perspective;
- refining the training pipeline to achieve satisfactory performance on the ASC task using the selected scenes;
- designing a testing protocol to ensure that the model's performance measurements obtained in the TensorFlow environment are faithful to the performance in the wild, when the model runs on the headphones;
- assessing the degree of robustness of the model to natural input perturbations that may occur when using it on a product;
- proposing a set of techniques to increase this robustness.

2.2 Challenges of the proposed task

The proposed task inherently presents several challenges, further aggravated by the constraints of the production environment.

Firstly, the deployment of the model on the *BeoMusic* was initially designed ad-hoc for the headphone's embedded platform: as a consequence, some of the operations and data-structures that ensure the functionality of the network were hard-coded. This made it cumbersome to modify components of the architecture of the CNN. Partly due to this reason, B&O's research interest shifted towards data-driven approaches that did not involve model modifications. Therefore, **the company requested that the network topology should have been modified as little as possible, and only if strictly necessary**. This constraint was relaxed in a late stage of the project, which allowed for a more free experimentation with different network architectures.

Secondly, the CNN that the candidate was initially required to use was extremely small, containing about 8000 parameters; moreover, the raw audio input signals underwent a rather heavy compression during the preprocessing step. In other words, **the overall complexity of the initial model was very low**. This represented a potential obstacle in classifying with acceptable accuracy a set of acoustic scenes that was rich enough to be of interest for usage in a product.

Thirdly, **no labeled data was initially available from the microphone mounted on the company's headphones**. As it is widely documented in the ASC literature, training a neural network on recordings produced by a microphone then testing it on recordings from a different microphone (a setting known as *device mismatch* or *microphone mismatch*) can lead to a severe decrease in performance [61, 82]. During the research activity, such an issue needed to be addressed.

Lastly, the task of selecting a set of acoustic scenes which to perform ASC on is notoriously problematic [16]: **the idea of enclosing every possible situation of reality in a limited set of scenes hides several potential pitfalls**. For

example, one may assume that the problem of classifying whether a scene is "indoor" or "outdoor" is a well-posed binary classification problem, since one can either be in an indoor or outdoor setting with no ambiguity; but what if the user stands close to a balcony, or in a structure that is not covered on all sides? Is that "indoor" or "outdoor"? Moreover, what if some scenes of interest are potentially overlapping, such as "chatter" and "train"? In that regard, some argue that ASC is inherently open-set, which makes the modeling task more difficult.

In conclusion, this work concerns an open-ended problem in a low-complexity and low-data setting. The thesis seeks to find a balance between the high-level requirements of a company (the generic intent of the device being scene-aware), the constraints imposed by a production system (small and initially fixed model, low data), and what can reasonably be achieved from a data-scientific perspective. Note that all challenges presented thus far are to some extent intertwined: for example, choosing a small number of simple acoustic scenes for the labels may compensate for the low model complexity, but could also reduce the potential use cases of the algorithm.

2.3 Initial pipeline setup

The project is set as a typical acoustic scene classification task where each acoustic scene constitutes one class. The front-end processing is carried out by computing a time-frequency representation of the raw audio signal; the back-end classification is performed by a convolutional neural network. In this section, we describe the details of the pipeline proposed by the company when the candidate joined the project.

2.3.1 Front-end preprocessing

The initial front-end setup is as follows: the audio signal to be analyzed is split into windows of 3.45 seconds, then preprocessed to produce an input $\mathbf{x}$ for the CNN classifier. When the input clip is longer than 3.45 seconds, it is divided in multiple windows that each become an individual data point associated to a label representing its scene. At training time, we obtain the windows by extracting a 3.45 second portion of the audio, applying a 1 second shift along the time axis, and repeating. Conversely, at test time, the shifts are as long as the input clip, i.e. there is no time overlap between consecutive inputs. The audio signal is re-sampled to 16 kHz and converted to 16-bit depth. At training time, if the clip is in stereo format, the two channels are separated in two distinct mono files. This can be considered a form of data augmentation (see Section 3.3.2), as we produce two slightly different versions of the same input content that are both used at training time. Ideally, at test time, all input signals should already be in mono format with

a 16 kHz sampling rate, since we capture them from a single microphone of the headphones with those characteristics. However, in our experiments, we often test on external sources of data that are in stereo format: in this case, we split all files to two mono tracks as in the training phase. In both cases, a Mel spectrogram is subsequently produced from the raw audio input. We choose Mel spectrograms since, while being a popular choice in audio preprocessing in general [7], studies have shown their greater effectiveness with respect to other traditional front-end choices such as MFCCs in deep learning based environmental sound classification [51]. The spectrograms are computed with fast Fourier transform (FFT) windows of 1024 samples and a hop length of 768; they are then mapped to the Mel scale using 30 Mel filters (normalized according to their energy) in the range 0 to 8000 Hz. The obtained spectrogram is then log-scaled. This specific preprocessing procedure results in an input tensor $\mathbf{x}$ of shape $30 \times 72 \times 1$ for the classifier; formally, we say that $\mathbf{x} \in \mathbb{R}^{F \times T \times C}$, with $F = 30$, $T = 72$, and $C = 1$. Following a widely used nomenclature, we call F the *frequency* dimension, T the *time* dimension, and C the *channels* of the input respectively.

2.3.2 Low-complexity back-end CNN classifier

The back-end classifier is a 5-layer convolutional neural network adapted from the SB-CNN architecture introduced in [107], which was originally used for environmental sound classification instead of acoustic scene classification. The first 3 layers are convolutional and the last 2 are fully-connected; between the convolution operations, max pooling and batch normalization (BN) [53] layers are present. During training time only, the fully-connected layers include dropout [116] as regularizer. This network topology was modified in [86] to adapt it to a low-complexity environment. The main modifications include:

- **Replacing max pooling with strided convolutions.** In a s-strided convolution, the convolutional filters move by s steps instead of 1 when sliding across the input tensor. This is useful to reduce the number of computations the model needs to perform in order to downsample the input features: instead of producing a wider output tensor and scaling it with pooling, a strided convolution directly performs a sparser transformation, reducing the number of needed operations.
- **Replacing convolutions with depthwise-separable convolutions.** In a normal 2-dimensional convolution, an input tensor of shape (F, T, C) is usually directly convolved with N 3D filters of shape (K_1, K_2, C). In a depthwise-separable convolution, the input is first convolved with C flat $(K_1, K_2, 1)$ shaped filters; the resulting intermediate tensor is subsequently convolved with N filters of shape $(1, 1, C)$. The latter step is known as "pointwise" convolution, since its filters only involve a single pixel per position. Given a fixed input

shape, a depthwise-separable convolution performs fewer operations and needs fewer parameters to produce the same output shape. Hence, it has been widely used in deep learning for embedded applications [47, 115].

Starting from the orginal architecture in [107], all max pooling layers but the last one are replaced with 3-strided convolutions, and all convolutions but the first one are made into depthwise-separable convolutions of the same input and output shape. The final neural network architecture has 8156 parameters and is reported in Table 2.1.

Input shape	Operator	Details
$F \times T \times 1$	Conv2d 5×5 + BN + ReLU	Stride 3
$F/3 \times T/3 \times 24$	Conv2d 5×5 + BN + ReLU	Depthwise sep., stride 3
$F/9 \times T/9 \times 48$	Conv2d 5×5 + BN	Depthwise sep., stride 3
$F/27 \times T/27 \times 48$	MaxPool 2×2 + ReLU	-
$F/54 \times T/54 \times 48$	Flatten + dropout	-
Previous shape flattened	Dense + Dropout + ReLU	-
32	Dropout + Dense + Softmax	-
4	-	-

Table 2.1: Initial SB-CNN architecture.

Throughout all the experiments involving the SB-CNN, the hyperparameters and the settings regarding the optimization of the neural network were kept fixed in order to make the effects of each proposed technique comparable. The parameters of the network are optimized using Adam optimizer [60], a variation of the SGD algorithm that has been shown to work well on a variety of deep learning tasks, including acoustic scene classification [44, 17]. We use an initial learning rate of $5 \cdot 10^{-4}$. The mini-batch size is set to 8, and we use L^2 regularization only on the parameters of the fully-connected layers of the network with coefficient 10^{-3}. The training procedure is carried out for 20 epochs and validation is performed at the end of every epoch: the network state achieving the highest validation metric on the validation set is selected as final version to evaluate on the test set. We choose the macro-averaged F1-score as validation metric. Let us define the precision and recall for a specific class i as follows:

$$\mathrm{Pr}_i = \frac{\text{true positives of class } i}{\text{total predictions of class } i} \tag{2.1}$$

$$\mathrm{Rc}_i = \frac{\text{true positives of class } i}{\text{total samples of true class } i} \tag{2.2}$$

The F1-score of a single class i is then the harmonic mean between the precision and the recall of class i; the macro-averaged F1-score is the mean of the F1-scores

of all N classes.

$$\mathrm{F1}_i = \frac{2 \cdot \mathrm{Pr}_i \cdot \mathrm{Rc}_i}{\mathrm{Pr}_i + \mathrm{Rc}_i} \tag{2.3}$$

$$\text{Macro F1} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{F1}_i \tag{2.4}$$

This metric is particularly useful when the goal is to have a classifier that is equally effective on all classes even though not all of them are equally represented within the dataset. For this reason, it will be employed as evaluation score for several experiments described in this dissertation.

Chapter 3

Model benchmarking in a low-data scenario

As previously mentioned, the CNN architecture used for the classification of acoustic scenes was initially considered to be fixed. Thus, a number of diagnostic operations were performed to obtain a baseline assessment of the model's capabilities.

As remarked in Section 2.2, one of the most critical aspects of the project was to establish whether training the network on a microphone and testing it on a different one would negatively impact the model's performance. However, such a verification would require to have at least a minimal amount of data from the microphone of the *BeoMusic* (hereafter referred to as "target microphone"). During the first part of the project, it was not possible to acquire data directly from the target microphone, due to the need of bypassing a set of internal components and deactivating some functionality of the headphones to obtain the raw microphone output: this operation required time and was not immediately executable. Therefore, the initial exploratory investigation had to be carried out by means of external sources of data.

In this Chapter, we describe the diagnostic experiments performed on the initial SB-CNN model to obtain a baseline reading of its capabilities. We start by defining a set of scenes that seems suitable for the complexity level of the model. As previously remarked, it is not yet possible to know whether a microphone mismatch will be an issue: thus, we assume the worst case scenario that it will negatively affect the system, and that the network will need to be trained on data manually gathered by B&O using the target microphone. This means we are in a low-data setting: we investigate the impact of this situation by training the CNN on different amounts of data and inspecting the results. We then propose different approaches to tackle this issue by using data augmentation and transfer learning to improve the performance of the neural network when training it with few data points.

3.1 Defining the acoustic scenes

To define a suitable set of scenes for the neural network to classify, we adopt the following strategy: first, we train the model on a well-known, publicly available ASC dataset; second, we evaluate the CNN on the test set; third, we perform error analysis on the results and establish which configuration of scenes would minimize the classification error of the model.

The employed dataset is the TAU Urban Acoustic Scenes 2019, the 2019 iteration of a dataset used in the popular *Detection and classification of acoustic scenes and events* (DCASE) challenge [82]. In the absence of data recorded from the actual target microphone, we use this as a proxy dataset and assume that the microphone used in it is the one that would be installed in the hypothetical final product.

The DCASE dataset is composed of audio recordings of urban scenery acquired in 10 different European cities. The recordings are grouped in 10 classes according to the specific environment they were taken in: those classes are what we refer to as acoustic scenes. The recording time of each scene is balanced across each city. The original recordings are about 5 minutes long; however, in the dataset, they are split in shorter clips of the duration of 10 seconds each. The scenes contained in TAU Urban Acoustic Scenes 2019 are the following:

- *airport*
- *bus*
- *metro*
- *metro station*
- *park*
- *public square*
- *shopping mall*
- *street pedestrian* (street with pedestrians)
- *street traffic* (mostly sound of cars and vehicles passing by)
- *tram*

The overall duration of the recordings is about 40 hours. All recordings were taken in stereo format using a Soundman OKM II Klassik binaural microphone at a sampling rate of 48 kHz and 24-bit resolution. The authors of [82] claim that, since the microphone is designed to be worn in the ears as a pair of earplugs, the recorded sounds should faithfully reproduce the stimuli that reach the auditory system of the wearer. While the organizers of DCASE provide an official dataset split for the challenge, throughout this work we produce several customized splits that are better suited to our needs.

To find suitable scenes for the model to classify, we start by assessing the capabilities of the network on a broad range of classes. We first preprocess the raw audio from TAU Urban Acoustic Scenes 2019 by re-sampling and splitting the

recordings as described in Section 2.3.1. This results in 26744 audio clips of 10 seconds each, which we partition as follows:

- 70% of the data (18719 clips, ≈ 52 hours of running time[1]) is used for training
- 15% of the data (4012 clips, ≈ 11 hours of running time) is used for validation
- 15% of the data (4012 clips, ≈ 11 hours of running time) is used for testing

We run a training procedure on the SB-CNN using the pipeline presented in Section 2.3.2 and inspect the outcome. The results are shown in Fig. 3.1.

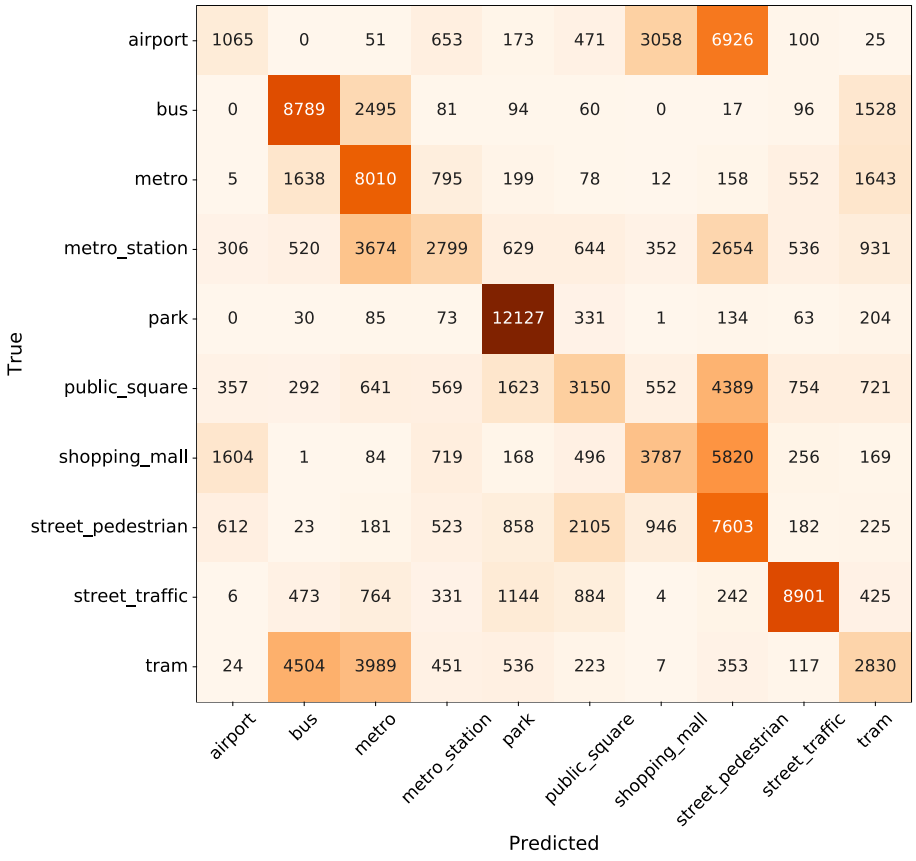


Figure 3.1: Initial model performance on TAU Urban Acoustic Scenes 2019 dataset using all 10 classes.

The current system achieves a macro-averaged F1-score of about 0.45; the score in terms of accuracy is approximately the same. As a comparison, the top performer

[1]Note that this is more than the total running time of the original TAU Urban Acoustic Scenes 2019 dataset as the number of available clips are doubled after splitting the stereo channels.

from the ASC task in the DCASE 2019 challenge scored around 0.85 [19]. Because the model version deployed in the *BeoMusic* on the time of this test supported 4 classes, a merging strategy was designed to group the 10 classes into 4 macro-categories. The categories were chosen to maximize the network's performance a-posteriori by merging together the classes that were most often confused. A semantically meaningful name representing the common aspects of the subclasses was given to the new 4 macro-classes. Specifically:

1. *bus*, *metro* and *tram* were grouped under the class *travel*, since they all represent means of transportation ($\approx$ 11 hrs running time in stereo format);
2. *airport*, *metro station*, *public square*, *shopping mall*, *street pedestrian* were grouped under the class *chatter*, since they all represent scenes where a lot of people are present around the user and chatter noises can generally be heard ($\approx$ 18 hrs running time in stereo format);
3. *park* was renamed *quiet*, with the intent of representing situations of silence. This class was easily recognizable from the very beginning and did not need any merging ($\approx$ 4 hrs running time in stereo format);
4. *street traffic* was renamed *vehicle*, and was not merged with any other class out of exclusion ($\approx$ 4 hrs running time in stereo format).

The choice of classes is in line with the recent trends in the usage of wearable audio devices: this was investigated by the company Audio Analytic in their 2019 survey [6] about consumers' habits regarding "hearables", i.e. computerized wearable electronics whose main purpose is audio reproduction such as earplugs, headphones, etc. The survey was conducted among 3000 people in the UK and 3000 in the US and focuses on what functionalities potential users would desire from combining hearables with artificial intelligence. The survey reports that the four most popular locations where people employ their audio devices are: at home, when traveling, at the gym, and "out and about". It is straightforward to see how such locations could match the 4 macro-classes established for our system: "home" would most likely be *quiet*, "traveling" would generally correspond to *travel*, "gym" might be represented by *chatter*, and "out and about" may include either *chatter* or *vehicle* (in the case of urban areas). We argue that the results from the survey seem to validate our model-driven class choice. We decide to continue our experiments

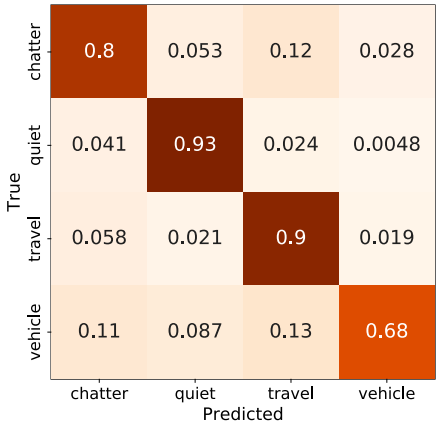


Figure 3.2: Model perfomance on TAU Urban Acoustic Scenes 2019 after grouping up the most often confused classes.

with those 4 new classes, as they seem to reflect a plausible use case scenario of the developed technology.

The same model's performance was evaluated a-posteriori on the new 4 classes after performing the merging operation (i.e. by re-labeling the already performed prediction outputs). The results are illustrated in Fig. 3.2. The model's accuracy now improves to around 0.85, while its macro-averaged F1-score is 0.79. As previously mentioned in Section 2.3.2, we deem the macro-averaged F1-score to be more suitable for our problem because of the class imbalance: the *chatter* and *travel* scenes have a higher number of samples than the *quiet* and *vehicle* scenes. This choice of metric is motivated by the fact that, on the final product, we would like the network to perform equally well on all classes; thus, hereafter we employ the macro-averaged F1-score as evaluation metric.

3.2 Impact of the low-data scenario

As previously mentioned, no labeled data from the target microphone was available at the start of the project. After an initial set of acoustic scenes was defined, we performed an assessment of the model's learning capability as a function of the amount of available training data. This would have been valuable information in the case that new training data would have had to be recorded.

We therefore define a fixed test set and simulate the availability of fewer data points by progressively increasing the size of the training set. For consistency, we also tune the size of the validation set according to that of the training set. More specifically, the setup is the following:

- we reserve at most 15% of the data (4012 clips, $\approx$ 11 hours of running time) for training;
- we reserve at most 15% of the data (4012 clips, $\approx$ 11 hours of running time) for validation;
- the remaining 70% of the data (18719 clips, $\approx$ 52 hours of running time) is fixed and used for testing.

We choose these proportions since, in the case of having to manually gather audio from our target microphone, simulating 24 hours of available training+validation data seems a plausible upper bound to set. Therefore, we keep the proportions of the training and validation partitions to 15% of the total dataset at most: this is equivalent to having slightly more than 11 hours of stereo audio in each set, which in turn are almost 23 hours in mono format. The rest of the data is used for testing.

To simulate the availability of different quantities of training data, we create several possible train and validation subsets of the size of 100, 200, 500, 1000, 2000,

and 4012 samples. We train a different version of the model on each of the given subsets maintaining the same training procedure illustrated in Section 2.3.2: for each split, we report the macro-averaged F1-score across all classes. The results are illustrated in Table 3.1.

Train. and Val. set size	**Running time**	**Macro-averaged F1**
100 samples	33m	16.6%
200 samples	1h 6m	37.4%
500 samples	2h 46m	62.0%
1000 samples	5h 33m	79.2%
2000 samples	11h 6m	79.0%
4000 samples	22h 16m	79.6%

Table 3.1: F1-score on TAU Urban Acoustic Scenes 2019 with increasing amounts of training data.

Our experiments indicate that the network's performance seems to saturate when being trained with about 1000-2000 samples, with an upper bound around 79% F1-score. Naturally, the general goal of the project will be to increase such upper bound as much as possible, i.e. to increase the overall generalization capabilities of the model. However, should the need of gathering training audio from the target microphone arise, we now have a rough indication of how much data may be required to reach a state of the CNN that is reasonably close to the upper bound of its capacity.

At this point of the project, access to the *BeoMusic* microphone was still not possible; thus, we moved forward assuming the worst-case scenario, i.e. a microphone mismatch does take place, it negatively impacts the functioning of the system, and it is necessary to gather new data using the target microphone. In such a case, the most useful effort would be to attempt to optimize the network so that it reaches its saturated state faster; in other words, we would like to improve performance in the low data scenario. The next section investigates this problem.

3.3 Tackling the low-data scenario

In the previous Section, we obtained a benchmark of the performance of the model when it is trained with a few minutes of audio, then gradually increased the amount of training data up to about 22 hours. Of course, there is an evident performance gap between the two settings. In this Section, we examine possible ways to increase the generalization capability of the neural network in the low-data regime and make it as close as possible to the that obtained with 22 hours of training data.

Historically, deep models have been shown to have greater generalization capabilities than shallower ones [113]; however, because the network topology cannot be changed for the reasons illustrated in Section 2.2, it is not possible to tweak the CNN's architecture in attempt to improve performance. Therefore, we adopt training-time strategies.

3.3.1 Transfer learning from a different microphone

Studies have shown how initialization of the weights of a deep neural network can considerably impact the outcome of the optimization procedure [119]; in that regard, a number of initialization techniques have been devised to ease the learning process of neural networks, some of which became standard in modern deep learning libraries [37, 42]. While such techniques have their perks, if training data is scarcely available, it is sometimes the case that not even a theoretically grounded stochastic initialization is sufficient for a successful learning procedure. In those scenarios, one commonly applied technique is *transfer learning*. In general, transfer learning consists in a process of two-steps [125]:

1. knowledge is acquired by training a model on a source dataset $\mathcal{D}_s$ to perform a source task $\mathcal{T}_s$;
2. the knowledge from the previous step is used to ease the learning of a target task $\mathcal{T}_t$ by means of a dataset $\mathcal{D}_t$.

More specifically, network-based transfer learning involves reusing some or all the weights of a network trained on $\mathcal{T}_s$ as initialization values for the training on $\mathcal{T}_t$. This serves as an alternative means of initializing the network's parameters and is one of the most effective and widely employed techniques to speed up training convergence (or allow convergence at all, if the data is scarce) in deep learning models. This approach is also known as *fine-tuning*.

In this instance, we hypothesize that, in the event of a low-data scenario, it would be possible to train the SB-CNN model on a ASC task that uses a different microphone for which a lot of data is available, then fine-tune the network using fewer data points from the target microphone. We perform such an experiment with the aid of a further dataset internally owned by the company that we refer to as Support scenes dataset. This dataset is the result of the composition of two different data sources. The first source is EigenScape [39], a dataset containing recordings of 8 acoustic scenes in fourth-order Ambisonics format [9]. The scenes are: *Beach*, *Busy Street*, *Park*, *Pedestrian Zone*, *Quiet Street*, *Shopping Centre*, *Train Station*, *Woodland*. The scenes were recorded with the Ambisonics microphone MH Acoustics EigenMike at 24-bit resolution and 48 kHz and a gain of +25 dB was subsequently applied to them (except for the *Train Station* class, which had a gain of +5 dB). Outdoors recordings were taken with a windshield, so no wind background noise is

present in them. For each scene, 8 recordings of exactly 10 minutes were taken, i.e. each scene has a total of 1h 20m running time.[2] In the Support scenes dataset, only the scenes *Busy Street*, *Park*, *Pedestrian Zone*, *Shopping Centre* and *Train Station* were included. The selected Ambisonics recordings were played-back in an anechoic chamber (see Section 4.2.1) and re-recorded with a pair of Behringer ECM8000 microphones and a Focusrite Scarlet 2i4 soundcard to obtain a down-mixed stereo version of them. The resulting files were stored in WAV format with a 32-bit precision and a sampling rate of 44.1 kHz. The second data source is a set of airplane and train recordings taken by a B&O employee in first-order Ambisonics format and re-recorded as stereo files the same way as the previous data source. The train recordings are 10 minutes long in total, while the airplane recordings amount to 1 hour and 11 minutes. We re-label the data according to the four classes established in Section 3.1. We obtain the following running times: 3h 50m of *chatter*, 1h 20m of *quiet*, 1h 20m of *vehicle*, 1h 21m of *travel*. The final label mapping and runtime for each class is shown in Table 3.2.

Original scene	**Running time**	**Re-mapped label**
Airplane	1h 11m	*travel*
Busy Street	1h 20m	*vehicle*
Park	1h 20m	*quiet*
Pedestrian Zone	1h 20m	*chatter*
Shopping Centre	1h 20m	*chatter*
Train	10m	*travel*
Train Station	1h 10m	*chatter*

Table 3.2: Original labels and and label re-mapping of the Support scenes dataset.

Armed with two available datasets, we setup the transfer learning process as follows:

- $\mathcal{T}_s$ is learning the four illustrated scenes from the audio representation obtained by the source microphone Behringer ECM8000, using the Support scenes dataset as $\mathcal{D}_s$;
- $\mathcal{T}_t$ is learning the same four scenes as represented by the Soundman OKM II Klassik, using the TAU Urban Acoustic Scenes 2019 dataset as $\mathcal{D}_t$.

Note that our final goal is to make $\mathcal{T}_t$ the learning of the selected four classes according to the characteristics of the microphone of the *BeoMusic*; however,

[2]We remove 10 minutes of audio from *Train Station* as they did not properly represent the label *chatter* indicated in Table 3.2.

when this experiment was being carried out, it was not yet possible to access that microphone. Hence, we use the Soundman OKM II Klassik as a "proxy" target microphone.

We train the model on $\mathcal{D}_s$ as in Section 3.2; then, we apply transfer learning and fine-tune the obtained model for 10 further epochs on $\mathcal{D}_t$ (the remaining hyperparameters are kept the same). The final state of the model is the one that achieves the best macro-averaged F1-score obtained on the validation set. Validation is performed after every epoch. We repeat the experiment while progressively increasing the size of $\mathcal{D}_t$ as described in Section 3.2. We report the macro-averaged F1-score obtained on the test set by each version of the CNN in Section 3.3.3.

3.3.2 Data augmentation: SpecAugment

Data augmentation consists in applying random noise and transformations to the data at training time in order to simulate the availability of more data points. It is an extremely popular technique in several fields of machine learning [111, 71, 126], including acoustic modeling [23]. Recently, a straightforward data augmentation technique for time-frequency signal representations called SpecAugment [91] was proposed in the context of speech recognition; because of its simplicity and effectiveness, it has been since investigated on a wide variety of different audio-related tasks [28, 123, 92, 14]. Given its success, we decide this general-purpose augmentation technique is a good starting point to attempt to improve the generalization capabilities of the model in the low-data scenario.

SpecAugment essentially consists in applying random 0-masking and time warping to an input spectrogram; for ease of implementation, we focus on the former transformation. The masking is applied independently over time and frequency dimensions. More specifically:

- Frequency masking consists in zeroing out the same f consecutive frequency channels in all feature frames of the spectrogram. Let ν be the total number of Mel bins of the input, and let F be a hyperparameter representing the maximum width of a masking band. We sample the actual width f of the masking band from a uniform distribution in $[0, F)$; subsequently, we sample the starting point of the masking band f_0 from a uniform distribution in $[0, \nu - f)$. All the values in the frequency bands in the range $[f_0, f_0 + f)$ are then zeroed out. We repeat this process N_f times, where N_f is a hyperparameter.
- Time masking works the same way as frequency masking, but in the time dimension. Let τ be the number of frames in the spectrogram. We first set hyperparameters T (maximum mask width) and N_t (number of masks). For N_t times, we set to zero the coefficients of all frames in the range $[t_0, t_0 + t)$, where t is uniformly sampled from $[0, T)$ and t_0 is uniformly sampled from $[0, \tau - t)$.

We repeat the same experiments described in Section 3.3.1, but using SpecAugment as online augmentation technique during the training phase. Considering that our input spectrogram is of size (30×72), we use the following hyperparameters: $N_f = N_t = 2$, $F = 4$, $T = 10$. Moreover, we trigger the overall SpecAugment transformation with a probability $p = 0.5$. Results are reported in Section 3.3.3.

3.3.3 Experimental results

Transfer learning greatly improves the CNN's performance in the low-data regime, providing an average increase of 30 percentage points in F1-score when training on 500 data points or below. SpecAugment also seems to be effective in increasing the generalization capabilities of the network in the low-data setting. To a lesser extent, this is also true when the model is saturated (i.e. when training it with 4000 samples): in that scenario, augmentation consistently provides a small performance increase (1-2%) both with and without transfer learning. The results illustrated in this section are reported in Fig. 3.3 and in Table 3.3.

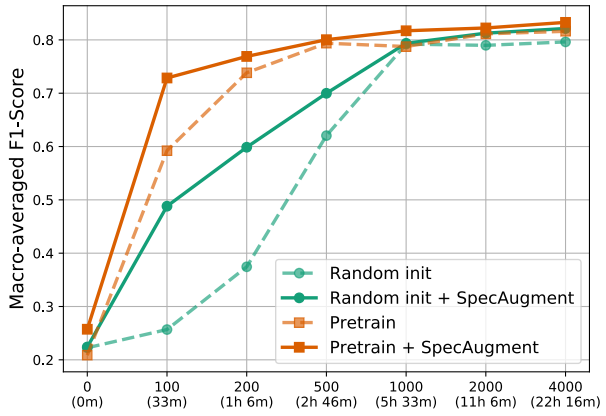


Figure 3.3: Macro-averaged F1 score of the SB-CNN on the TAU Urban Acoustic Scenes 2019 test set with increasing amount of training data and different training configurations.

Training mode	**F1-score with 4000 training samples**
Random init	79.6%
Random init + SpecAugment	81.6%
Pretrain	82.1%
Pretrain + SpecAugment	83.2%

Table 3.3: Test set results by SB-CNN with training at high-data regime.

We conclude that data augmentation is useful in increasing the neural network's generalization capabilities regardless of the training setting: thus, we permanently incorporate it in the the pipeline. However, for such a small model, the effects of transfer learning seem to quickly diminish when the available data increases. The technique will therefore be employed if a microphone mismatch takes place and the available data is scarce; however, in the presence of abundant data points

(i.e. $\gg 4000$), its application may not be necessary. Regardless, it is now possible to establish a baseline training routine. Henceforth, we refer to the SB-CNN trained with SpecAugment (with the hyperparameters listed in Section 3.3.2) as "baseline model". Whether or not transfer learning will be included in the baseline model is yet to be established, as it depends on the effect that the mismatch between source and target microphone will have on the neural network. In Chapter 4, we shall investigate this issue and establish the precise impact of microphone mismatch on the system.

Chapter 4

Closing the gap between research and production

The divergence between research and production systems in the field of machine learning has been remarked a number of times in the literature [10]. This phenomenon is especially true in embedded applications, where models must be deployed on specialized hardware and cannot run in their native environment (often a Python-based engine such as TensorFlow [76] or PyTorch [94]). While frameworks that partially automate the deployment operation to embedded environments did recently emerge [24, 87], a company may sometimes need to design its own proprietary deployment pipeline because of hardware compatibility reasons, as is the case with this project. Because of that, the system needs to be attentively sanity-checked before being installed into a product.

The pervasive spread of ML has fostered the surfacing of a set of best practices to systematically train, test, and maintain ML models in production environments; this broad range of practices has been given the name MLOps [80]. More specifically, some attention has been given to how to properly test a model in a production environment [101, 128, 31].

In this section, we describe the work made to unify the results obtained in a proxy system such as TensorFlow with the actual model running in the company's product. The main steps of our pipeline include:

- **Input capture**: capturing an audio stimulus with a microphone and converting it to a 1-dimensional PCM signal.
- **Front-end preprocessing**: computing Mel spectrograms of the input signal.
- **Back-end CNN classification**: the neural network produces scene probabilities from the Mel spectrogram.

Within the scope of this project, two distinct versions of this pipeline exist: one for prototyping (running in a TensorFlow/Keras environment) and one deployed in

the company's headphones (running on the target embedded platform). While the two serve the same purpose, they effectively are independent systems that employ different technologies: one could say between them is a "gap" that must be "filled" to successfully deploy the DL-based product. Fig. 4.1 illustrates this concept. In order to unify the two pipelines into one, we tackle each step individually.

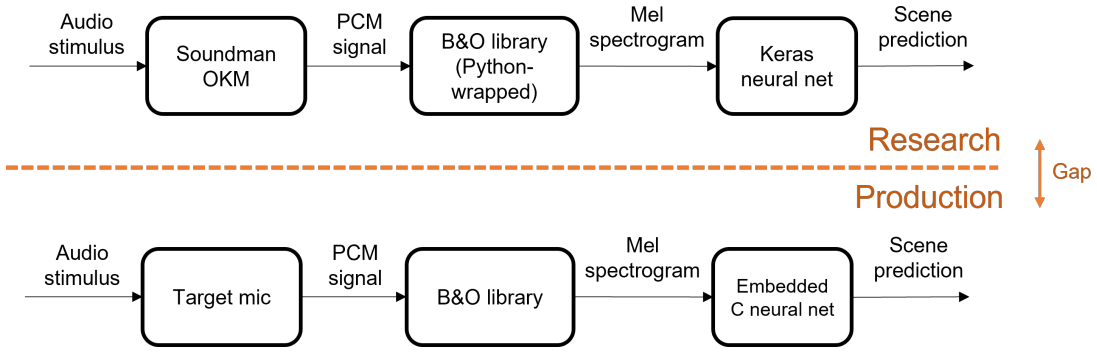


Figure 4.1: The two parallel pipelines of the research and the production environments.

4.1 The model gap: front-end processing and back-end neural network

We begin by addressing the last two steps of the pipeline, since the assessment of their behavior is purely technical and does not involve the study of the data distribution at hand (i.e. the specific audio stimuli input into the system).

In fact, the front-end preprocessing steps of the research and the production pipeline hardly need any assessment operation to be unified, since the generation of Mel spectrograms is handled by a proprietary B&O library. The same C code can be deployed on the headphone platform and also used in a Python environment thanks to a custom wrapper. Thus, this processing step performs the same operations in both platforms, and is guaranteed to yield the same results.

However, the neural network implementation is different for the two pipelines: TensorFlow is used for prototyping in the research environment, and the corresponding C implementation is used on the hardware platform. Therefore, it is necessary to testify the numerical identity of the two CNNs. We establish that the two back-end classifiers are equivalent by forwarding the same set of data in the two versions of the CNN. We sample 30 minutes of audio from the test set described in Section 3.2, making sure that the four established classes are equally represented. The obtained data points are preprocessed and passed through the two versions of the same network. We then assess how many samples obtain the

same prediction from the two classifiers, regardless of the fact that the predicted class matches the ground truth or not. For this reason, the exact version of the CNN weights used for this experiment is not strictly relevant, so long as it is the same for both the TensorFlow and C pipelines. We employ the model version reported in the last row of Table 3.3. The C model is compiled as an executable binary and run on a Linux operating system. The binary is the same that would run on the headphone's hardware.

Our experiment shows that about 99% of the clips extracted from the test set are classified the same way by both versions of the neural network. In all clips that are classified differently between the two pipelines, at least one of the networks has a difference in probability of 5% or less between the most likely and second most likely class: in other words, the classification mismatch between the two implementations only takes place in situations of high uncertainty, when a slight oscillation in the CNN's final activation values can be just enough to cause the maximum argument of the softmax to change. We hypothesize this might be due to slight differences in the numerical representation used by the two platforms. In spite of that, we deem the behavior of the two CNN versions sufficiently similar: it is possible to state that the research-production gap between the last two stages of the pipeline is effectively closed.

4.2 The data gap: microphone mismatch

The last step to unify the research and production pipelines is related to the input data. Because the training and testing microphone are potentially different, it is unreasonable to assume that the result of the input capturing step will be the same for both development and production pipelines. Therefore, what one can do is to ensure that such a difference does not considerably affect the downstream task, and if so, what can be done to mitigate the issue.

As mentioned in Section 2.2, the act of training an ASC model on a set of training data recorded from a specific microphone then testing it on audio coming from a different microphone is known as *device mismatch* or *microphone mismatch*; recent literature suggests that this setup can significantly diminish the classification effectiveness of the model [61]. This issue has received a considerable amount of attention from the ASC community, to the point that the DCASE 2019 challenge has dedicated an entire sub-task to acoustic scene classification with mismatched devices [82]. However, while several techniques have recently surfaced to tackle the issue of device mismatch, they all have been applied in the context of relatively large models and fine-grained feature extraction (usually spectrograms with either 128 or 256 frequency bins [49, 78, 57]); in our scenario, the model is extremely simple, and our feature extraction setup implicitly applies a remarkable level

of compression to the data (using 30 Mel bins means extracting very high-level coefficients). For this reason, it is necessary to practically assess whether or not microphone mismatch actually affects our network. In this section, we propose different approaches to tackle the problem: first, we re-record a restricted set of test data points from the target microphone, and verify if the performance of the CNN drops after re-recording data; second, we propose a way to simulate the fact that an existing audio clip was acquired using the target microphone by means of impulse response convolution.

4.2.1 Re-recording part of the test set

Ideally, in order to verify whether a device mismatch impacts the classification capabilities of the network, we would need to assess the model's predictions on the same audio stimuli recorded from two different microphones:

- The microphone that recorded the data used for training (*source microphone*)
- The microphone that will be used in the final device (*target microphone*)

The above can be achieved by extracting part of the test dataset, re-recording it through the target microphone, and evaluating the obtained samples with the trained neural network. Bang & Olufsen's labs have a specific environment suitable for such a task: an anechoic chamber with a spherical array of 40 speakers that can emit sound without introducing reverberations from the room. More details about the loudspeaker array setup can be found in [4].

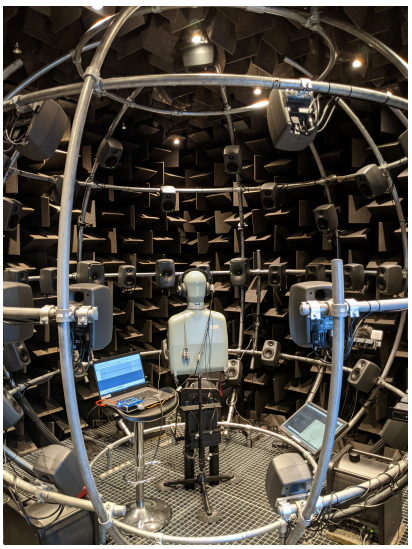


Figure 4.2: Recording setup with the headphones worn by the HATS in B&O's anechoic chamber.

The recording setup is the following: we use the same 30 minutes of audio from the original test set (uniformly sampled across the four classes) as in Section 4.1. We set the headphones at the center of the spherical speaker array, worn by a B&K 4128 head and torso simulator (HATS). We then play the samples from only one speaker placed in front of the HATS, directing the sound in the direction of the listener from a distance of about 1 meter. The recording setup is shown in Fig. 4.2. To minimize the spurious differences between the original and the replayed clips, we choose to play back the sound as mono even though the clips from TAU Urban Acoustic Scenes 2019 are originally stereo. To reduce the clips to mono from stereo, we extract only their first channel. The

files are then routed through the speaker array by means of a program produced with the software MAX.[1] The signal is processed by a RME MADIface XT audio interface, which does not support reproduction of sound at 16 kHz. Hence, we play the files at the original sampling rate of 48 kHz, then later down-sample them to 16 kHz to make them compatible with the model. The recording of sound from the *BeoMusic* is performed by using a special set of headphones whose internal microphone has been connected to a wire that runs externally to the device: this way, it is possible to directly stream the signal recorded by the microphone to a sound card and save it into a computer. In our setup, we use a Focusrite Scarlet Solo sound card to capture the output signal of the microphone at a sampling rate of 16 kHz, then save it into a laptop computer placed to the side of the headphones by means of the software Audacity.[2] It is relevant to point out that, while this setup is suitable to perform a comparative test between two microphones, it may not be completely faithful to all possible use-case scenarios where the headphones would be used. Indeed, here we are reproducing audio coming from a single source with no refraction coming from the surrounding environment: in settings like a normal room, sound reflections would be present. One might argue that this configuration is more representative of situations where the microphone is subject to highly directional audio stimuli in environments with low amounts of sound reflections, such as open spaces with few obstacles around the listener. While this limitation is reasonable for the purpose of this experiment, as a sanity-check, we also re-record the audio clips in a normal quiet room, played back from a B&O *A9* speaker. This serves as a verification of the obtained results: ideally, the classification quality obtained form a normal room with reverb should neither exceed that of the anechoic chamber nor be noticeably worse. This time, the headphones are placed on a stand facing directly the A9 and about 1.5 meters away from it. The sounds on the A9 are played back directly from a laptop computer. As regards the headphones, the recording setup is the same as before.

At the end of the procedure, we obtain a number of what we call *comparison sets*:

- The *source comparison set*, containing 30 minutes from the source microphone Soundman OKM II;
- The *target comparison set A*, containing the same 30 minutes of audio content recorded through the target microphone in the anechoic chamber;
- The *target comparison set B*, containing the same 30 minutes of audio content recorded again through the target microphone, but in a normal room.

We then test the same trained model used in Section 4.1 on all datasets. Because

[1] `https://cycling74.com/products/max`

[2] `https://www.audacityteam.org`

the audio content is the same, any possible difference in performance between the source and target sets should be caused purely by the microphone's mismatch. The process described thus far is illustrated in Fig. 4.3.

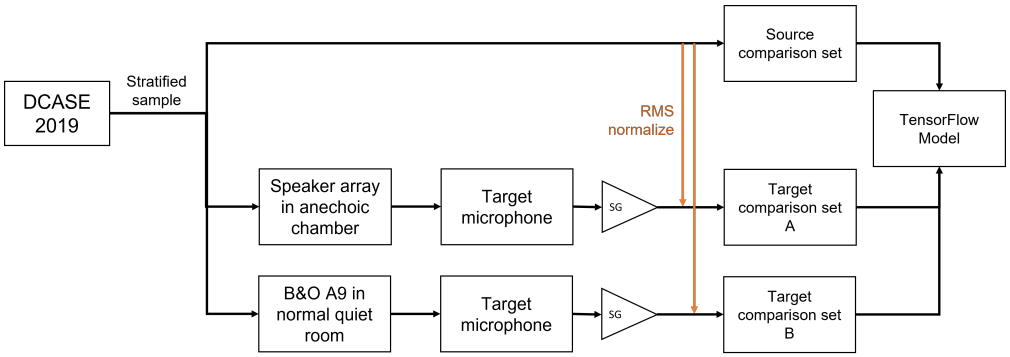


Figure 4.3: Diagram of the generation and evaluation of the *comparison sets*.

While the content of corresponding clips in the source and target comparison sets should be the same, we expect the re-recording process of the audio to introduce several spurious gains, denoted as SG in Fig. 4.3. Such gains originate from the routing of the input sound to a sound card, the reproduction of the sounds through the loudspeakers, etc. Therefore, with no specific post-processing, the final volumes of corresponding clips in the source and target sets are not guaranteed to be the same. While the model should indeed be robust to slight volume variations, our main goal here is to only measure the impact of the microphone mismatch with the influence of no other factor. Thus, we re-align the volume of the re-recorded clips by means of root mean square (RMS) normalization. We generally define the RMS of a signal $\mathbf{x}$ as:

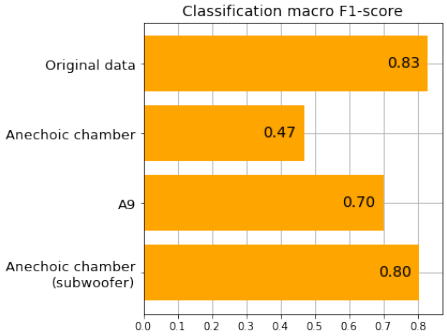
$$\mathrm{RMS}\left(\mathbf{x}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|x_i - \mu_{\mathbf{x}}\right|^2} \tag{4.1}$$

where N is the number of samples of $\mathbf{x}$ and and $\mu_{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} x_i$. The volume-normalized clip $\widetilde{\mathbf{x}}^{(t)}$ is then computed from the source clip $\mathbf{x}^{(s)}$ and the re-recorded clip $\mathbf{x}^{(t)}$ as:

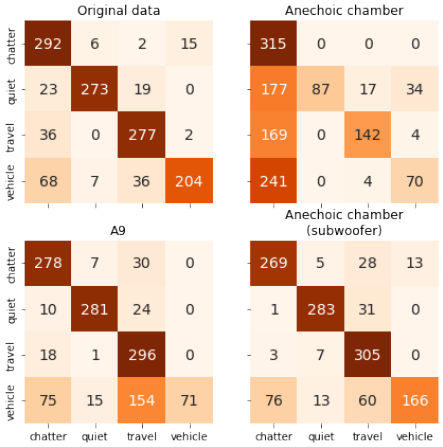
$$\widetilde{x}_i^{(t)} = \frac{x_i^{(t)}}{\mathrm{RMS}\left(\mathbf{x}^{(t)}\right)} \cdot \mathrm{RMS}\left(\mathbf{x}^{(s)}\right) \quad \forall i \in [1, N] \tag{4.2}$$

After normalization, the obtained comparison datasets are evaluated by the neural network with a regular inference procedure. Surprisingly enough, classification performance seems to drop dramatically with target set A, while it is only slightly

diminished on target set B; more specifically, it seems that the classes *travel* and *vehicle* are mostly ignored by the model while in the anechoic chamber, and the majority of samples are either classified as *chatter* or *quiet*. Conversely, the A9 introduces fewer disturbances, mostly in the *vehicle* class (Fig. 4.4b). This results in a performance drop of about 37% in terms of macro F1-score in the anechoic chamber, and of only 10% when using the A9 in a normal room (Fig. 4.4a). Such a result would seem to suggest that the reflections in a normal room actually improve classification performance; however, this does not seem plausible, as the original dataset was not recorded by playing back sounds inside a room, and it is unlikely that a model could exploit the phenomenon of reverberation without having experienced it during training.[3] Indeed, as previously remarked, one would have expected the performance on target set B to match that of dataset A at best, but not to surpass it.



(a) Macro F1-scores of different microphone conditions.



(b) Confusion matrices showing the network's classification performance in different microphone conditions.

Figure 4.4: Results of the microphone impact assessment. Re-recording the audio clips in an anechoic environment using the *target* microphone does not seem to critically worsen the classification performance of the model.

[3]Note that, when we say "reverberations", here we mean those that are introduced by re-recording an audio clip a second time with the *target* microphone after it was first captured by the *source* microphone in its original environment. As previously illustrated, this kind of "re-recording"-induced reverberations are present in the normal room, but not in the anechoic chamber. However, in the broader sense of the word, reverberations coming from objects in the original environment (such as cars and buildings) are always present in the recordings. We are not concerned about that kind of reverberations, as they are naturally present in all original audio clips.

In order to investigate the issue, we compare the Mel spectrograms produced by the preprocessing of the same clip in the source set, in target set A and target set B. We produce the spectrograms with the B&O library common to both pipelines, to make sure that they replicate exactly what data is given as input to the model at inference time. When compared, the spectrograms obtained from the source set and the target set B appear to be visually similar. Their corresponding audio clips also roughly sound the same, except for a slight (and expected) additional reverb in the target set B. However, in target set A, we notice a lack of intensity in the low frequencies of the re-recorded clips: specifically, the very first frequency bin of the spectrogram appears to be much weaker. For many samples, the phenomenon is visually noticeable by inspecting their time-frequency representation, but may also be confirmed by plotting the average value of the first energy bin across time (Fig. 4.5), which collects energy from 0 to roughly 120 Hz. We hypothesize that the performance drop in the anechoic chamber is due to the following:

1. The model is very sensitive to low frequencies;
2. The anechoic chamber's loudspeaker array is not playing that frequency range.

The first hypothesis can be verified with the saliency map technique [112]. In DL, a pixel-wise saliency map roughly indicates how much each individual pixel contributes to the classification output of a certain class. Let $\mathbf{g}_c(\mathbf{x})$ be the logit activation of the neural network associated to class c when the input is the spectrogram $\mathbf{x}$. The saliency map $S_c(\mathbf{x})$ of the sample $\mathbf{x}_0$ can be obtained by executing a normal forward pass through the network, then computing the gradient of the activation of class c with respect to the input spectrogram.

$$S_c(\mathbf{x}_0) = \nabla_{\mathbf{x}} \mathbf{g}_c(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} \tag{4.3}$$

This can be easily achieved by extending the normal back-propagation operation to the input tensor.

Because in the anechoic chamber a lot of *travel* and *vehicle* samples are classified as *chatter*, we inspect an instance of the *travel* class (extracted from the file `bus-milan-1180-45241-a.wav` of TAU Urban Acoustic Scenes 2019) and compute the saliency map associated to the class predicted by the model. In the case of the correctly predicted *travel* class, the model does not appear to overly pay attention to the first bin (Fig. 4.5, top row); however, when the clip is re-recorded in the anechoic chamber, the network seems to mostly focus on the lack of low frequencies and uses that hint to erroneously conclude that the sample belongs to the *chatter* class (Fig. 4.5, third row). This issue does not occur when the clip is played through the A9, which instead reproduces the low frequencies correctly (Fig. 4.5, second row).

The definitive confirmation of the fact that the model highly depends on the first Mel bin can be achieved by manually masking it and verifying that the network's predictions change drastically. As an example, let us take again the file

`bus-milan-1180-45241-a.wav` (from the source set), compute the Mel spectrogram of 3.45 seconds extracted from it, and feed the result into the CNN: the model's prediction is *travel* with 88% probability (top of Fig. 4.5). Subsequently, we substitute all coefficients of the last row of the input matrix (i.e. the lowest-frequency Mel bin) with a low constant value,[4] and repeat the forward pass: the most likely class becomes *chatter* with 91% probability. Such phenomenon is graphically illustrated in Fig. A.1 of Appendix A.1.

While this investigation uncovered an interesting weak spot of the model, one may argue that the problem here is not the network itself: the issue mostly lies in the speaker array of the anechoic chamber not reproducing the low frequencies that are sometimes necessary to correctly recognize certain acoustic scenes. Moreover, it is worth noticing that some of the inspected *travel*-labelled clips also contain some chatter information; however, thanks to the low-frequency information, the model is able to ignore them and focus on the fact that the user is on a bus or train, which may actually be a desirable behavior.

Further investigation revealed that the cutoff frequency of the loudspeakers that compose the spherical array is ~ 55 Hz, hence the first bin of the Mel spectrogram containing a relatively low energy. In order to reproduce low frequencies, a subfwoofer present in the room had to be activated by means of a specific setup. The subwoofer is an omnidirectional speaker that is meant to reproduce low frequencies only: indeed, in professional audio systems, subwoofers sometimes play independently from the rest of the speakers because of their omnidirectionality [32]. In contrast, the B&O A9 is a commercial product that is meant to operate in a standalone fashion and independently reproduces all frequencies in the audible range with no special setting required. This explains the improved results despite the non-optimal environment for audio reproduction.

We activate the subwoofer in the anechoic chamber, repeat the recording procedure and evaluate the model on the new clips. As shown in Fig. 4.4, the results in the anechoic chamber now mostly align with the results of the original audio, with a difference of about 3% F1-score. As expected, low frequencies are also back to normal energy levels and the model does not overly focus on the first Mel bin (Fig. 4.5, fourth row). These results suggest that, on a 30 minute test set, a device mismatch does not dramatically impact the model's performance; nevertheless, some slight degradation (3% F1-score) can be seen. The two following questions then naturally arise:

- Does this degradation remain the same when scaling up to a larger, more

[4]Because we take the logarithm of the Mel spectrogram, its numerical values are negative. We find -4 to be a reasonable "low constant", as frequency ranges of low energy seem to oscillate around that value.

statistically significant test set?
- Is there a way to verify the above without having to re-record hours of test data?
- In case of future model development, is it possible to generalize such procedure to any source microphone instead of just the Soundman OKM?

We attempt to tackle some of those issues in the next section, where we try to manually distort the input audio according to the characteristics of the target microphone to simulate the fact that the clip was captured with it.

4.2.2 Simulating new microphone data by means of impulse response convolution

Instead of having to re-record several hours of audio, one can aim to directly simulate the target microphone's behavior on existing audio clips. In general, microphones are designed to behave as linear time invariant (LTI) systems when operating in a desired frequency range [43]. A LTI system is a system whose output linearly depends on its input according to a time-invariant mapping. The relationship between input $x(t)$ and output $y(t)$ of the system can be fully described by means of its *impulse response* function $h(t)$:

$$y(t) = h(t) * x(t) \tag{4.4}$$

where $*$ denotes the convolution operation. In a discrete time setting, the impulse response describes how the system reacts to a unitary impulse $\delta(t) = \mathbb{1}_{t=0}$. Assuming the target microphone T can be modeled as an LTI system, it should be possible to convolve an input audio signal $x(t)$ with its impulse response $h_T(t)$ to introduce the same artifacts and distortions that the microphone would introduce when recording the audio track.

The problem then becomes how to capture $h_T(t)$. A popular approach is to design an ad-hoc "inverse filter" function $f(t)$ that performs the so-called "deconvolution" operation [30, 103]:

$$h(t) = y(t) * f(t) \tag{4.5}$$

The inverse filter function depends on the input signal $x(t)$. Indeed, by expressing (4.4) in frequency domain and using (4.5), it is possible to show the following:

$$\left.\begin{aligned} H(f) &= Y(f) \cdot \tfrac{1}{X(f)} \\ h(t) &= y(t) * f(t) \end{aligned}\right\} \longleftrightarrow f(t) = \mathcal{F}^{-1}\left\{\frac{1}{X(f)}\right\} \tag{4.6}$$

Where $\mathcal{F}^{-1}$ is the inverse Fourier transform. One possible approach to finding the inverse filter is then to choose an appropriate form of $x(t)$ so that $\mathcal{F}^{-1}\{1/X(f)\}$

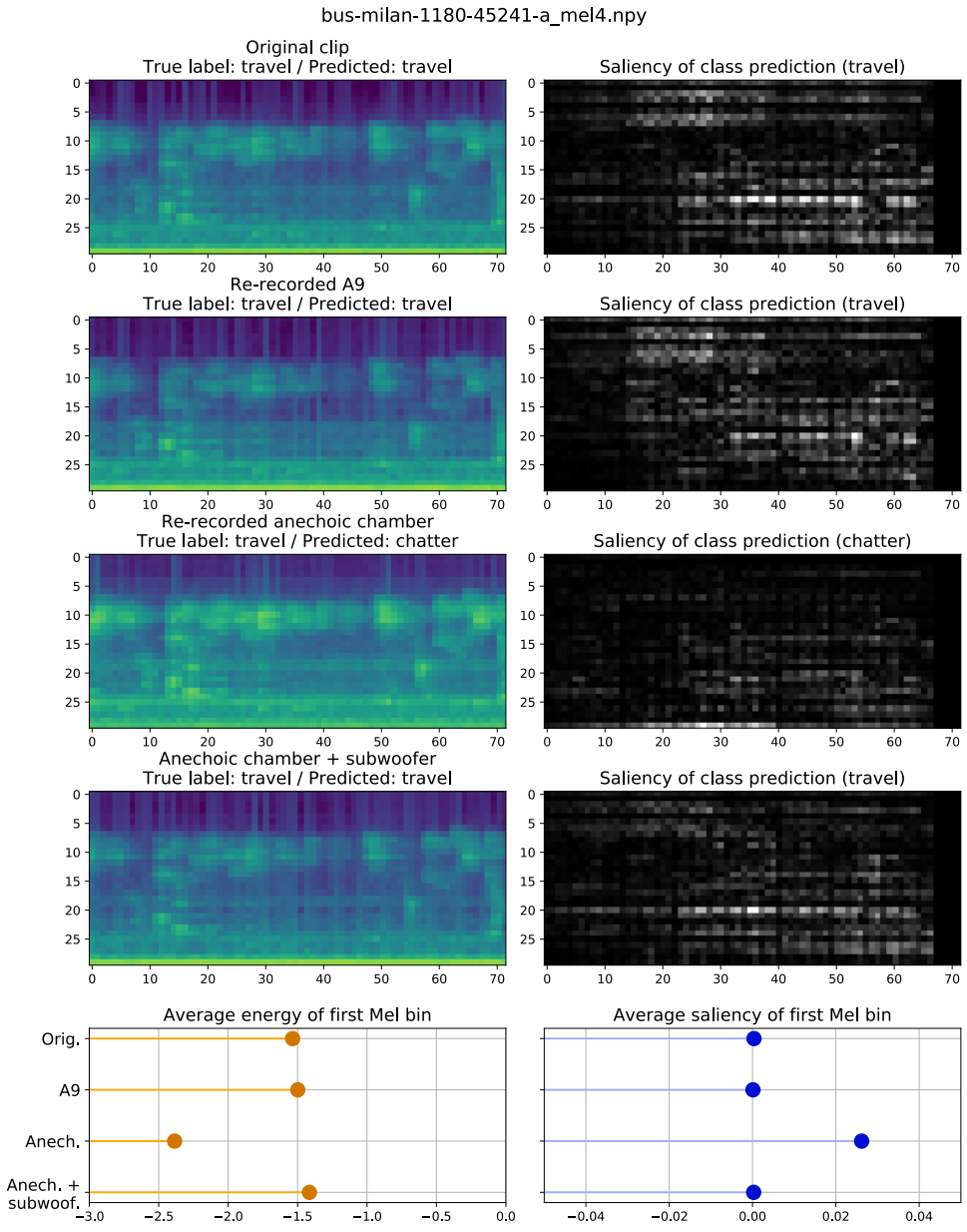


Figure 4.5: Left: Mel spectrograms of the same sample from the *travel* class under different conditions. Right: saliency maps for the class predicted by the neural network. Notice how the samples re-recorded in the anechoic chamber present lower energies at lower frequencies on average (bottom left). This seems to cause the model to overly concentrate on the lack of low frequencies and overlook the rest of the information: this can be observed by averaging the saliency value of the first Mel bin in different settings (bottom right).

is mathematically tractable. A popular choice of $x(t)$ is represented by a sinusoidal signal of duration T whose instantaneous frequency exponentially increases from f_1 to f_2, often referred to as "sine sweep" [117, 29, 88]:

$$x(t) = \sin\left(2\pi f_1 L\left[\exp\left(\frac{t}{L}\right) - 1\right]\right) \tag{4.7}$$

Where L is a parameter that depends on T and f_2 according to:

$$L = \frac{T}{\log\left(\frac{f_2}{f_1}\right)} \tag{4.8}$$

If $x(t)$ is chosen as previously described, then [88] shows how the inverse filter can be computed in closed form as an exponentially decaying, time-reversed version of the original input:

$$f(t) = \mathcal{F}^{-1}\left\{\frac{1}{X(f)}\right\} = \frac{f_1}{L}\exp\left(-\frac{t}{L}\right)x(-t) \tag{4.9}$$

In our experiments, we use $f_1 = 0$ Hz, $f_2 = 22050$ Hz and $T = 3$ seconds. In the anechoic chamber, we reproduce $x(t)$ at 44.1 kHz through a loudspeaker and record the microphone output $y(t)$ with the same setup described in Section 4.2.1. It is crucial to perform this operation within an anechoic environment: if not, the obtained impulse response $h_T(t)$ would not only include the effects introduced by the microphone, but also the sound reflections produced by the room itself.

A further issue to consider is that the microphone of the target device is not omnidirectional, i.e. it does not have the same sensitivity to air pressure from every angle: thus, in principle, the microphone has several impulse responses $h_{T,\theta}$ depending on which angle θ the impulse comes from. As mentioned in Section 4.2.1, the re-recording was performed by reproducing audio from the speaker facing the listener, i.e. from a 0° angle. Thus, this is the main direction we are interested in when assessing if the impulse response method and the re-recording method yield the same results.

Nevertheless, because the experimental setup is the same, we proceed to record impulse responses from all angles around the HATS: this will allow to simulate the fact that the input audio comes from very specific directions. As remarked in Section 4.2.1, this "high-directionality" setting may not be realistic in every use case: regardless, we choose to perform the experiment as it may be of interest to further understand the model's behavior under those specific conditions. Thus, we record 12 different impulse responses coming from equally spaced intervals of 30° on the horizontal plane where the microphone lies. In other words, we play $x(t)$ 12 times, each time from a different loudspeaker placed at a different angle with respect to the headphone placed on the HATS: this results in 12 different

recordings $y_\theta(t)$ for $\theta = 0°, 30°, 60° \ldots$. We then compute $h_\theta(t)$ by convolving each recording with the reverse filter as in (4.5). This way, with a single experimental setup, it is possible to verify two phenomena: first, if the performance of the model on the data captured with the target microphone is comparable to that simulated with the impulse response; second, if the previous hypothesis is confirmed, then it is also possible to verify how the model would react to highly directional stimuli. A directional stimulus can be simulated by convolving an audio input with the impulse response of the desired angle.

In practice, in our experiments, the same *source comparison dataset* from Section 4.2.1 is convolved with each response $h_\theta(t)$, forming a set of 12 new test sets, each corresponding to a different angle. After convolution, each obtained clip is RMS-normalized as in Section 4.2.1. The trained CNN is then evaluated on each new dataset by means of macro-averaged F1-score. For the purpose of tracking the various results, we establish that the loudspeaker directly in front of the wearer is at a 0° angle, and the rest of the speakers follow the standard polar coordinates convention. The recording microphone is installed in the left ear pad of the headphones: thus, its position corresponds to the 90° angle. As previously remarked, the purpose of this experiment is twofold:

1. The results on the 0° angle will reveal whether re-recording and using impulse response-based convolution yield approximately the same results on the model. If so, it should mean that impulse response convolution will be a viable way to assess the effect of microphone mismatch on a larger scale in the future.
2. The results on all other angles will reveal whether the model can distinguish highly directional stimuli with the same content and if its performance changes based on the direction of the input.

Evaluating the model on the data convolved with the impulse response at 0° results in a macro-averaged F1-score of 85%, which is in line with the results on the original, non-convolved test set and reasonably close to the results obtained when re-recording the data (see Fig. 4.4a). This would seem to suggest that the simulation via impulse response convolution can reproduce the same results obtained when re-recording the audio clips.

For further visual confirmation, we compare the spectrograms of the re-recorded signal and the one convolved with the impulse response of $\theta = 0°$ (Fig. 4.6): we render the complete spectrogram (not mapped onto the Mel scale) to have a clearer picture of the artifacts introduced by both methods. While of course not completely identical, the re-recording and convolution approaches seem to introduce similar distortions in the input signal. Some differences seem to be present from 18 kHz onwards; however, since the input signal is resampled to 16 kHz before preprocessing, this is not really relevant to our investigation. We can conclude the following: microphone mismatch is not a major concern for our system in its

current state, but if needed, data recorded from the target microphone can be simulated by means of convolution with the impulse response of the microphone itself.

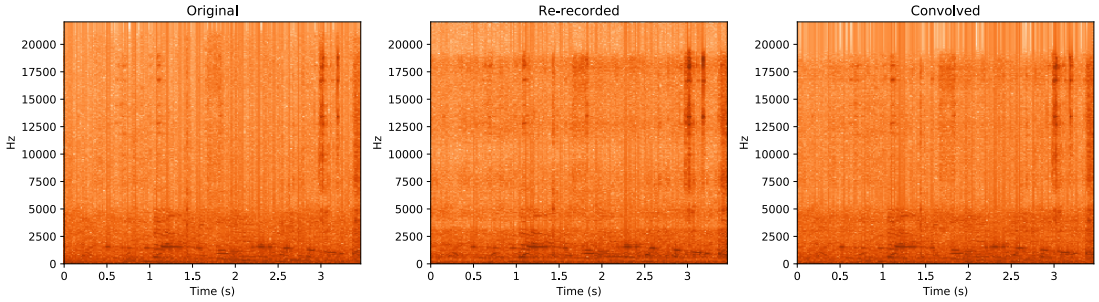


Figure 4.6: Comparison of the spectrograms of different versions of the same input test clip: the original from TAU Urban Acoustic Scenes 2019 (left), the one re-recorded in the anechoic chamber (middle), and the one convolved with the target microphone's impulse response (right). The spectrograms are computed with 1024 FFT samples and hop length of 768, but the Mel filters are not applied to them this time to ease visual comparison.

Nevertheless, and interestingly enough, reverberation effects produced by angles other than 0° seem to impact the model's performance to various degrees. The best results are achieved when the reverb comes from 0°, and the F1-score noticeably degrades at 90°, reaching a value as low as 0.46. In other words, reverberations coming directly from the direction of the microphone seem to greatly confuse the neural network. The performance at other different directions fluctuates, but is still acceptable. In order to verify that the degradation is not due to spurious reflections generated by the room's equipment during the recording, we repeat the acquisition of the impulse responses by rotating the entire HATS by angles of 90° and performing the whole experiment every time. This results in a set of 4 independent experiments, numbered from 1 to 4. In order to factor out possible reflections coming from the own torso of the HATS, in experiments 2 and 4 we only turn the head of the mannequin and leave the torso in its previous position. Moreover, to verify that the $x(t)$ input sweep is not so loud that it causes distortion in the microphone itself, we repeat each experiment at two different volume levels: a "loud" one and a "quiet" one, with a playback gain of -12 dB and -24 dB respectively. In other words, we produce 12 impulse angles $\times$ 4 head positions $\times$ 2 volume levels, for a total of 96 different impulse responses. We then convolve the source comparison set with each of the responses and test the CNN on it.

The new results are in line with the previous experiment. The model seems to consistently fail at 90°, regardless of the position of the HATS in the room. This happens when the sweep is played both at high and low volume. Fig. 4.7 illustrates

the results of the experiment in the loud setting (for the results in the quiet setting, see Fig. A.2 in Appendix A.1). We try to get an insight of why the network constantly fails at a certain direction by inspecting the frequency representation of the impulse responses coming from +90° (where the problem is most evident), −90° (the complete opposite side) and 0° (a middle point between the two). For completeness, we examine the spectra acquired from all four head positions. As expected, the behavior of a frequency response appears similar when coming from different head positions with the same angle θ. It is possible to see how the spectra of the impulse responses acquired at +90° present a number of "bumps" that are not found in the other two angles: as Fig. 4.8 illustrates, the bands around 500-600 Hz, 2000-3000 Hz and 5000-6000 Hz seem to have greater energy in the case of the $\theta = +90°$. In order to verify which of these is causing the misclassifications, we again take some of the original audio files and manually modify their frequency bands. More specifically, we inspect samples from the *quiet* and *travel* class, and check whether by tweaking the energy of one of the above-mentioned frequency ranges it is possible to force the network to misclassify the input clips as *chatter* (similarly to what was done in Section 4.2.1). While the boosted bands around 3000 and 5000 Hz do not seem to critically impact the model's decisions, it appears that the band around 500 Hz is quite relevant for the classification task: indeed, by lightly boosting the energy values of all Mel bins between 300 Hz and 700 Hz, it is possible to make some *quiet* and *vehicle* samples appear as *chatter* to the CNN.[5] Some examples of this phenomenon are graphically illustrated in Fig. A.5 of Appendix A.1.

Because of the multiple head positions and the asymmetric nature of the issue, it is unlikely that the frequency bumps found around +90° are caused by the reflections generated by the exposed metallic parts of the HATS. However, in order to have a tangible confirmation, we repeat a set of measurements after clothing the mannequin with a sweatshirt and a hat. Clothes on the torso simulator tend to absorb sound vibrations and can avoid reflections phenomena: in a sense, this makes the experiment more realistic, as it simulates the presence of clothes on a real user. Since no considerable differences were observed among different volume levels and head settings, the experiment was reproduced only for the *loud* setting at position 1 (top right of Fig. 4.7). Repeating the inference operation with the same model on all angles resulted in no significant differences compared to the unclothed HATS scenario. Moreover, the boost from 300 to 700 Hz frequencies is still present. For further insight about how the experiment with the clothed HATS was conducted and a graphical illustration of the results, see Appendix A.2.

[5] Again, because of the fact that the coefficients of the spectrograms are negative, we "boost" their values by multiplying them by a coefficient $c \in (0,1)$, thus making them "less negative". In our case, we choose $c = 0.7$.

At this point, it is reasonable to believe that the different behavior of the CNN for different values of θ is due to the microphone itself, either because of its intrinsic directivity or because of tolerances in the specific microphone unit we used. Whatever the root cause, this results in a difference in impulse responses coming from different angles, and the neural network at its current state does not seem to have sufficient generalization capabilities to tell that the two signals should actually be classified the same way. We acknowledge this problem and try to address it in Chapter 5, where we propose a number of techniques to increase the robustness of the model, and Chapter 6, where we analyze the impact of using a different CNN architecture in the pipeline.

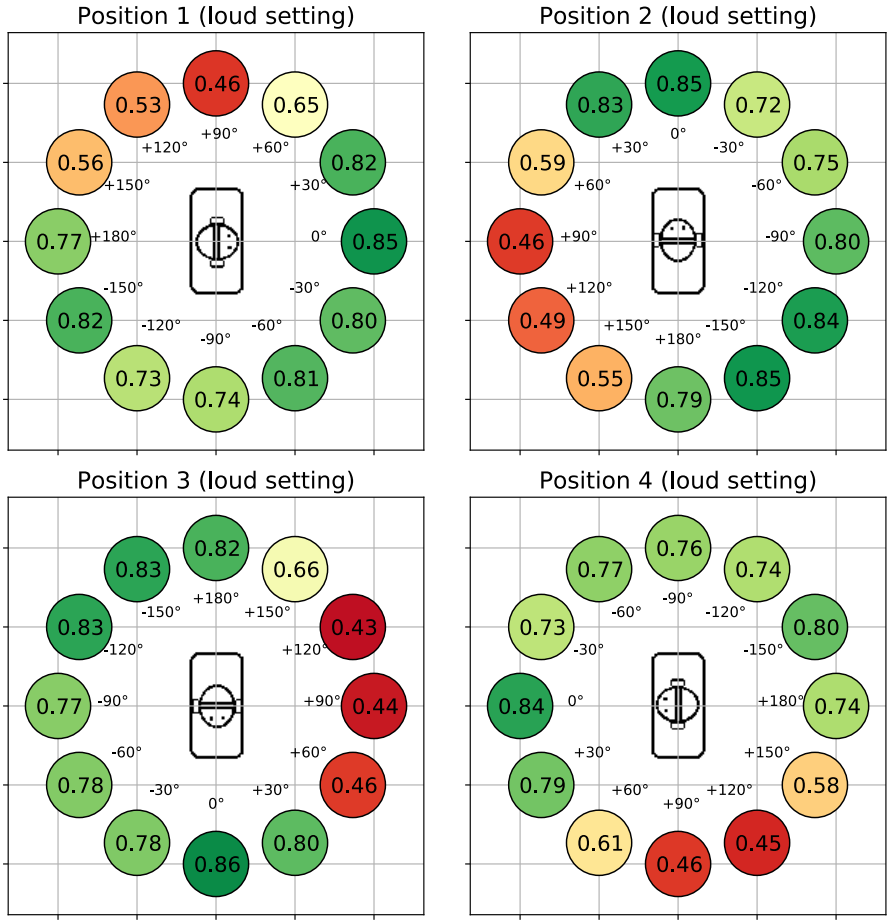


Figure 4.7: CNN performance when convolving the input with impulse responses acquired from different angles. Regardless of the position of the HATS in the anechoic chamber, the model seems to consistently fail at a 90° angle.

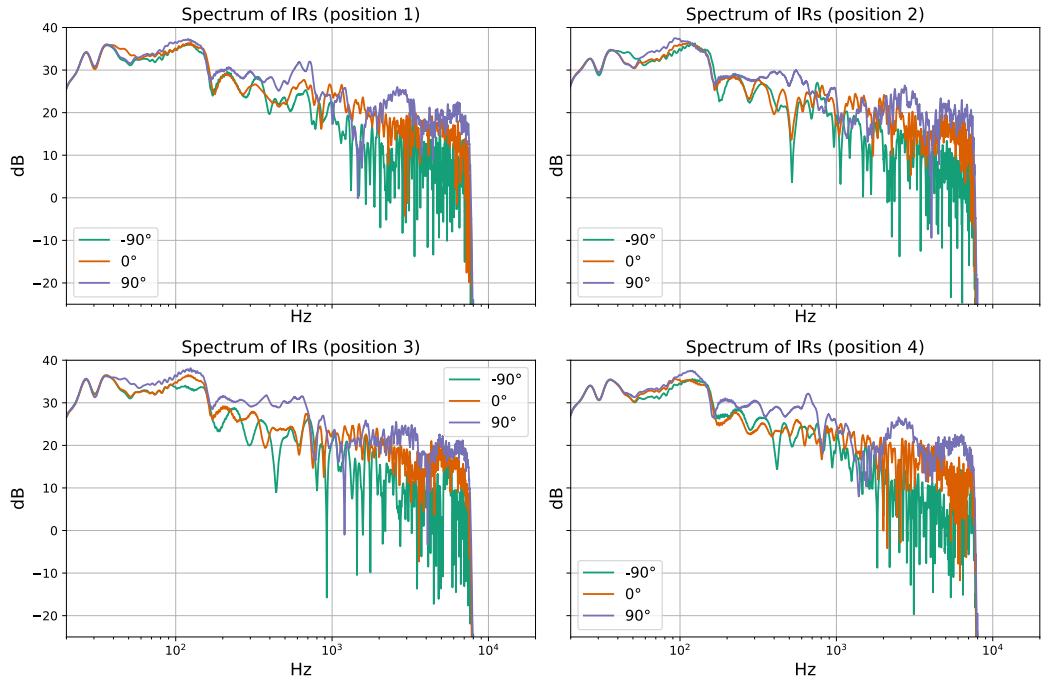


Figure 4.8: Spectra of impulse responses of the *BeoMusic* microphone coming from different angles. The data is reported for all four head positions illustrated in Fig. 4.7. The impulse responses were all re-sampled to 16 kHz prior to FFT computation, hence the cutoff at 8 kHz. No remarkable difference can be seen between different head positions: this suggests that the difference in frequency responses at different angles is not due to spurious reflections produced by the equipment, but rather to the functioning of the microphone itself.

For the moment, it is possible to conclude that the "gap" between the research pipeline and the production pipeline has been bridged to a reasonable extent. For the current modeling setup, we know that training and testing on the TAU Urban Acoustic Scenes 2019 dataset in the TensorFlow environment will be an acceptable approximation of the performance of the model in real use cases. We therefore move on to further investigations pertaining to the optimization of the neural network itself.

Chapter 5

Trade-offs in model robustness

In Chapter 4, we concentrated our efforts on closing the gap between the research prototyping environment and the production pipeline. We were able to show how it is reasonable to train and evaluate the TensorFlow model on data coming from the publicly available TAU Urban Acoustic Scenes 2019 dataset and expect that the same performance will be roughly maintained when deploying the model on the target embedded platform. In the process, we also uncovered some details about how the model takes decisions, making it more easily interpretable.

Still, even when having this knowledge, it is useful to evaluate the model in a real scenario while running it on a physical wearable device. Therefore, a *BeoMusic* was programmed to run the trained model on its hardware and perform the classification at the press of an external button while capturing input from its own microphone. A dedicated software patch also made it possible to record data from the target microphone to a smartphone via Bluetooth pairing, in order to be able to store audio input for subsequent evaluation and analysis.

In general, we follow the high-level strategy of establishing a protocol of continuous model development that includes the following steps:

1. **Test in the wild.** The model is tested on different situations to identify edge cases.
2. **Construct a test set.** Gather data that contains the discovered edge cases and keep it as an independent set of testing data.
3. **Adapt the model.** Find new training or inference strategies to solve the identified problem, and evaluate the effectiveness of such strategies on the gathered test data.
4. **Repeat.** Once the performance on the edge case is satisfactory, go back to step 1.

The goal of this process is to obtain a number of test sets representing different situations in which the model can fail. It will then be possible to use those as a benchmarking tool to develop a more and more reliable acoustic scene classifier.

Our approach is loosely based on the "data-centric AI" paradigm, which has recently gained momentum within the machine learning community, especially in the context of deployment of models in production systems [85]. According to the idea of data-centric AI, the data used in a ML pipeline should be iteratively refined after each deployment according to the issues found with the model. We adapt this approach to our scenario, where we want to perform an assessment of the robustness of the system to perturbations that could potentially make it fail.

One of the corner cases where the CNN's capabilities are known to decrease was already presented in Section 4.2.2: the susceptibility to reverberations coming from different angles. In this Chapter, we mainly focus on that and two other potential threats to the effectiveness of the model: the presence of wind in the input and the possible shifts in the data distribution underlying the four macro-classes established in Section 3.1.

5.1 New dataset split and baseline

When the training procedures carried out in Chapter 3 were performed, it was still unclear whether the phenomenon of device mismatch would make it necessary to collect a new dataset recorded from the target microphone. If that had been the case, we would have been in a "low-data scenario": recording new and diverse enough acoustic scenes from a specific microphone is an expensive and time-consuming operation that would not have allowed the construction of a large training set. Because of the uncertainty about the impact of microphone mismatch, the worst case scenario was assumed: that we would indeed be in a low-data scenario. Therefore, the training and validation sets used for the experiments in Chapter 3 were reduced to a very small size; as a natural consequence, the rest of the data from TAU Urban Acoustic Scenes 2019 was used to form a large, very comprehensive test set.

However, throughout Chapter 4, it was shown that device mismatch does not negatively impact the model to a considerable extent: this effectively disproved the low-data assumption. Hence, we are now free to use the TAU Urban Acoustic Scenes 2019 dataset for training purposes to a wider extent.

First, we decide to remove all samples from the original *metro_station* scene from the data. In Section 3.1, we established that this class belonged to the macro-label *chatter*; however, as Fig. 3.1 shows, clips from this scene are very often (and understandably) confused with *metro* scenes, which were instead assigned to the *travel* macro-class. Because the samples from *metro_station* reasonably include sounds that can also be found in *metro*, we conclude that the expected

macro-labeling of this class could equivalently be *chatter* or *travel*. In other words, this scene is essentially ambiguous: we find it could be of little benefit if used during training, and may even make proper evaluation of the network more difficult if used in testing. Therefore, we remove it from the available data; after this modification, we are left with 36 hours of stereo audio, which we split as follows:

- 26 hours (7800 stereo clips) are used for training;
- 4 hours (3000 stereo clips) are used for validation;
- 6 hours (3600 stereo clips) are used for the test set.

As previously done, we sample the data so that all 10 original scenes are equally represented in each split. The same setup as Chapter 3 is kept for training. The merging strategy of the original scenes into four macro-labels is also kept identical to that described in Section 3.1.

However, at test time, we now attempt to simulate a more realistic use case of the model. While the TAU Urban Acoustic Scenes 2019 dataset is composed of data points corresponding to 10 second clips, each clip originally belongs to a longer recording of a single scene, usually of the duration of approximately 3-5 minutes. For example, the clip `street_pedestrian-barcelona-141-4285-a.wav` belongs to the scene `barcelona-141` and is distinguished by its numerical ID `4285`. There are 21 other clips in the dataset that belong to the same scene: consequently, there are 210 seconds (3m 30s) total of the scene `barcelona-141`. It is possible to retrieve all the clips from the scene `barcelona-141` from the dataset, sort them by their numerical ID, then join them in a single WAV file to reconstruct the full scene recording. Our new test set is composed of 3-5 minute long WAV recordings of individual scenes reassembled as described above. We select a number of scenes that amount to approximately 6 hours of running time.

The goal of designing the new test set this way is twofold: first, we ensure that all test scenes are truly unseen at training time;[1] second, we try to retain contextual information during testing instead of evaluating the model on isolated 3.45 second clips. The latter aspect will especially be useful in Section 5.2.2, where we attempt to exploit the history of past predictions to make the model more robust.

Once the new dataset split has been established, it is possible to train the model again to obtain a new baseline evaluation. When doing so, we obtain a macro-averaged test F1 score of 78.1%. It is worth mentioning that such a score is not to be compared with the previous baseline, as the test set is different, smaller

[1]We point this out because simply performing a standard train-validation-test split as in Chapter 3 made it so that (different) data points belonging to a same scene would be seen both during training and testing. While this is not a mistake from a merely data-scientific perspective (no individual data point is seen both during training and testing), reserving a whole scene recording to either training or testing represents a more realistic, yet more challenging approach.

and less representative; plus, classification of its scenes has become more difficult for the previously explained reasons.

5.2 The issue of passive wind generated by the user's movement

When testing the network on the physical wearable device, a problem soon surfaced: while the model is somewhat robust in situations when the user is still, the classification performance seems to greatly deteriorate when the user is moving too fast while wearing the device. More specifically, if the classification is performed while walking at a sufficiently, yet not unnaturally quick pace, the model sometimes predicts *travel* even in environments that would normally be classified as either *chatter* or *quiet*, such as an office. A preliminary investigation on the matter consisted in recording some audio from the microphone while walking, then listening to it in order to look for possible artifacts or auditory clues that may be the cause of the misclassification. What appeared immediately noticeable from the recordings is that the microphone is very sensitive to the excitation caused by the air reaching it: when the user moves, this phenomenon is more evident, and an impulsive form of background noise can be heard. This was appointed as one of the possible reasons for the model's failures. Hereinafter, we refer to this event as *passive wind*. Passive wind can visually be observed in frequency domain from the model's input spectrogram, where bursts of energy at low frequencies are visible (Fig. 5.1).

We observe multiple spectrograms from a long 30 seconds clip and compare the the prediction probabilities produced by the network with the time-frequency representation of the input: we notice that, when more low-frequency energy bursts are present, the model is more likely to classify the scene as *travel* (Fig. 5.2). Moreover, in Section 4.2.1, it was established that the CNN is highly dependent on low frequencies for the classification of the *travel* scene. When put together, these two pieces of information lead us to believe that the reason behind passive wind being misleading for the model is those low-frequency bursts.

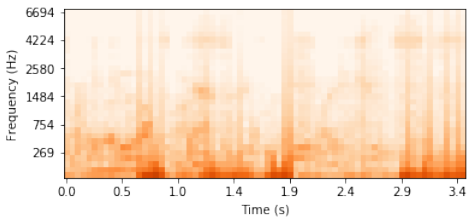


Figure 5.1: Spectrogram illustrating the issue of passive wind reaching the microphone. The original audio is a recording of office background noise while walking. Note the bursts of energy in the low-frequency regions corresponding to gusts of air.

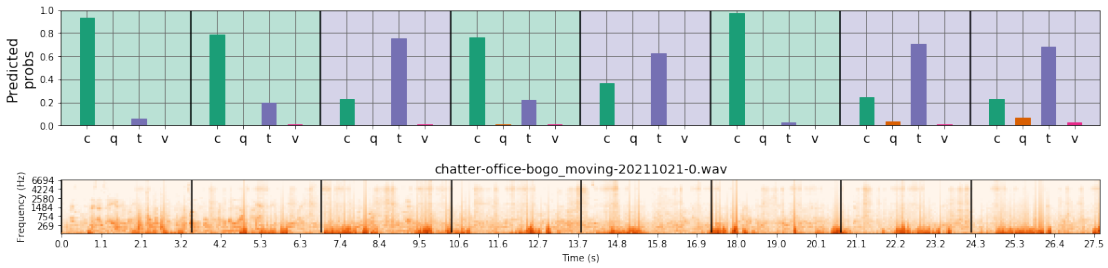


Figure 5.2: Model predictions for a 30-seconds clip recorded under the condition of passive wind with ground truth *chatter*. The top chart shows the probabilities produced by the network. The background of each frame shows the final prediction (green: *chatter*; purple: *travel*). In the presence of passive wind, the predicted label oscillates between the ground truth and *travel*.

5.2.1 Reproducing the passive wind issue by additive augmentation

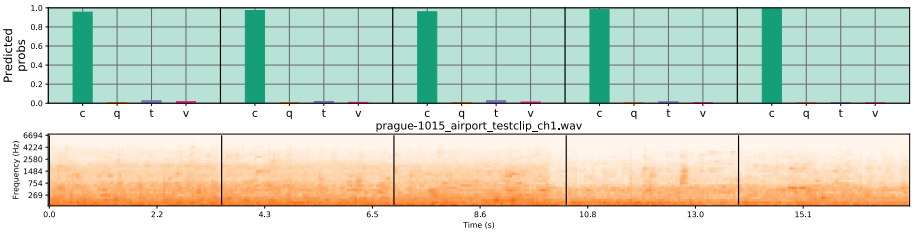
Whatever technique we propose to tackle the issue presented in Section 5.2, it is essential to first make sure we have a functioning method to reproduce it on a large scale, so that any attempted mitigation of the problem can be effectively evaluated. One simple, yet effective approach is to superimpose wind noise to the final recordings. We first go to a completely quiet room that is large enough to walk across at a certain pace: we do that to reproduce the passive wind phenomenon. We obtain a number of recordings from the microphone of the device where the only audible stimulus is the sound of passive wind. Notice that we do not have to worry about the device mismatch issue here, as the noise recording is coming from the target microphone itself.

Starting from a clean audio sample $\mathbf{x}$, it is then possible to obtain a version of it corrupted with a wind recording $\mathbf{n}$ of the the same duration by means of simple addition. The noise signal is rescaled first so that the resulting clip has a certain desired signal-to-noise ratio value (SNR), specified in dB:

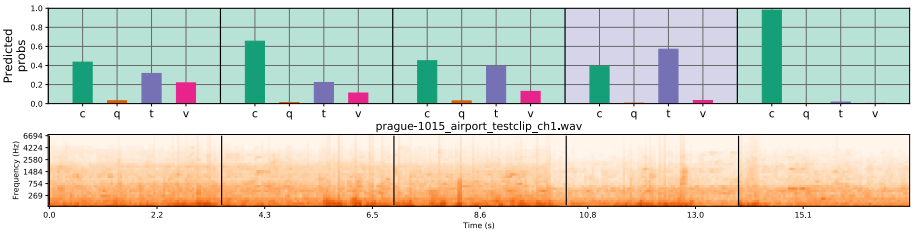
$$\begin{aligned} \mathrm{RMS}_{\text{target}} &= \mathrm{RMS}\left(\mathbf{x}\right) \cdot 10^{-\mathrm{SNR}_{dB}/20} \\ n'(t) &= \frac{n(t)}{\mathrm{RMS}\left(\mathbf{n}\right)} \cdot \mathrm{RMS}_{\text{target}} \\ x_{\text{noisy}}(t) &= x(t) + n'(t) \end{aligned} \tag{5.1}$$

If $\mathbf{x}$ and $\mathbf{n}$ are not of the same duration (as it is often the case) it is possible to either cut the noise clip to the desired length (when $\mathbf{x}$ is shorter than $\mathbf{n}$) or join multiple noise recordings in a random fashion (when $\mathbf{x}$ is longer than a single $\mathbf{n}$ clip). With this method, we are able to simulate the effect of wind into the microphone

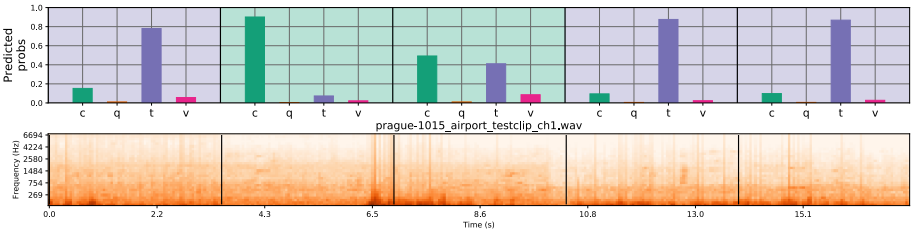
and bias the baseline model's prediction with varying degrees of severity: a visual demonstration is given in Fig. 5.3. This is proof that the approach described in (5.1) can effectively reproduce the issue of passive wind in a controlled manner.



(a) Predictions of the baseline model over a *chatter* clip.



(b) Predictions of the baseline model over the same clip as Fig. 5.3a with superimposed noise at −5 dB SNR.



(c) Predictions of the baseline model over the same clip as Fig. 5.3a with superimposed noise at −10 dB SNR.

Figure 5.3: The effect of additive wind noise at different SNR values over the same clip. When no noise is added (Fig. 5.3a), the model predicts the correct class (*chatter*) with high accuracy. Adding passive wind noise at −5 dB SNR (Fig. 5.3b) increases the outcome probability of the *travel* class to the point where some misclassifications occur. As the SNR decreases, the model becomes more and more biased towards the *travel* class (Fig. 5.3c).

Armed with this benchmarking tool, we now propose a number of techniques to try and tackle the issue of passive wind and evaluate them against the test set corrupted with different degrees of distortion. More specifically, we generate 5 new

versions of the test set that are corrupted with noise levels of 0 dB, −5 dB, −10 dB, −15 dB, and −20 dB SNR respectively. It is worth pointing out that −20 dB is indeed a very low quality signal, and we do not expect the model's performance to be closely comparable to the clean input scenario in such a situation: at that distortion level, even a human would find the acoustic scene classification task to be somewhat challenging. Nevertheless, we choose to test the model in such a scenario anyway in order to explore to which extent we can push the robustness to passive wind.

5.2.2 Post-processing approach: smoothing the predictions with a hidden Markov model

As previously observed, the effect that passive wind has on the system is to trigger spurious misclassifications due to impulsive low-frequency stimuli. Therefore, one straightforward and lightweight option to mitigate the problem would be to apply a smoothing filter to the class prediction to mask sporadic inaccuracies in the classification task: we implement such a filter as a hidden Markov model (HMM). HMMs have a long history of usage in speech-related tasks [97] and have occasionally been applied to ASC in the past [15, 27, 21]. In general, a HMM represents a time-dependent phenomenon with a fixed number of N possible states $S = \{s_1 \dots s_N\}$; the transition sequence $\{q_1 \dots q_T\}$ between such states is determined by a set of T time-ordered observations $O = \{x_1 \dots x_T\}$. A HMM is determined by three fundamental elements:

- a state transition matrix $\mathbf{A}$ where each cell $\mathbf{A}_{ij}$ represents the probability $\mathbb{P}(q_t = s_j \,|\, q_{t-1} = s_i)$ of transitioning from state s_i to state s_j;
- a conditional probability function $b_j(x_k)$ representing the probability $\mathbb{P}(x_k \,|\, q_t = s_j)$ of observing x_k while in state s_j;
- a set of initial state probabilities $\boldsymbol{\pi}_j = \mathbb{P}(q_1 = s_j)$.

The set $\lambda = \{\mathbf{A}, b, \boldsymbol{\pi}\}$ completely defines a HMM and can either be manually established or learned from the data. Then, by means of techniques such as the *forward propagation* algorithm [97], it is possible to compute the likelihood of a certain sequence of observations O given a model λ. In symbols, this would mean computing $\mathbb{P}(O|\lambda)$. A typical application of a HMM consists in learning one model λ_c for each class c of the task at hand, then for a given set of observations O' performing class inference as:

$$\hat{c} = \underset{c}{\operatorname{argmax}}\, \mathbb{P}(O'|\lambda_c) \tag{5.2}$$

In the case of ASC, the set of observations O is typically a set of feature vectors that compose a spectrogram [27]. In other words, for a task with C possible scenes, a set

of C HMMs provides a mapping from a single spectrogram to an estimated acoustic scene. In that case, the N states of the model represent abstract configurations of features vectors and N is a hyperparameter of the system.

As of today, this kind of usage of hidden Markov models in the context of ASC has been mostly replaced with neural networks [3]. Indeed, essentially all participants of recent editions of the DCASE challenge rely on CNNs applied to spectrograms without explicitly modeling the concept of a sequence over the input [19, 44, 17, 64, 130, 58].[2] We argue that this is because acoustic scene classification is less of sequential nature than other audio tasks (e.g. speech recognition), since an acoustic scene can be seen as a somewhat "stationary" input, where the exact order of occurrence of certain acoustic events is not exceedingly relevant: for example, hearing a car honk and a bicycle ring implies we are near a street regardless of their order.

Nevertheless, motivated by the phenomenon of the passive wind illustrated in Section 5.2, we argue that it is relevant to model sequentiality of the class prediction itself more than that of the feature vectors. More specifically, it is unlikely that the user will change the scenery they are immersed in more than once in a short amount of time: for example, when the wearer of the headphones is in a quiet park, they will likely remain there for a few minutes. Similarly, the value of the predicted class should not oscillate: if a car drives by and engine noises can be heard for just a few seconds, the predicted scene should not change.

In light of this assumption, we formulate the hidden Markov model in a different, more simplistic way than that described above: we employ only one HMM, where each of the N states corresponds to a possible predicted class (i.e. an acoustic scene). The $\mathbf{A}$ matrix then contains the probabilities of transitioning from scene i to scene j. The observation $\mathbf{x}$ is an entire spectrogram, and at each time unit t the backbone CNN performs a prediction on a different input $\mathbf{x}$. Using our HMM λ, we would like to estimate the quantity

$$\alpha_t(i) \triangleq \mathbb{P}(\mathbf{x}_1 \dots \mathbf{x}_t, q_t = s_i \mid \lambda) \tag{5.3}$$

to then perform the prediction of the next acoustic scene $\hat{s}_t$ as

$$\hat{q}_t = \underset{1 \leq i \leq N}{\operatorname{argmax}}\, \alpha_t(i) \tag{5.4}$$

To compute $\alpha_t(i)$ for every time instance t, we use the *forward algorithm* as

[2] One might advocate that a wide enough $(k_f \times k_t)$-sized convolutional filter applied on the input spectrogram will implicitly model some form of "sequentiality" since it can simultaneously process data coming from k_t consecutive time steps. However, as mentioned in Section 1.2.1, there exist specific ways to model time by means of pure convolutions [90, 40], none of which seem to be popular among the entries of the DCASE competitions.

described in [97]. The value of α for each time step is iteratively computed as:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\, \mathbf{A}_{ij}\right] b_j(\mathbf{x_t}) \tag{5.5}$$

We approximate the conditional $b_j(\mathbf{x_t})$ with the prediction $\mathbb{P}(q_t = s_j \mid \mathbf{x})$ produced by the network itself, assuming uniform sample density $\mathbb{P}(\mathbf{x})$ and class probability prior $\mathbb{P}(q_t = s_j)$:

$$\begin{aligned} b_j(\mathbf{x}) = \mathbb{P}(\mathbf{x} \mid q_t = s_j) &= \frac{\mathbb{P}(q_t = s_j \mid \mathbf{x})\, \mathbb{P}(\mathbf{x})}{\mathbb{P}(q_t = s_j)} \\ &\propto \mathbb{P}(q_t = s_j \mid \mathbf{x}) \end{aligned} \tag{5.6}$$

In our implementation, we compute (5.5) in logarithmic domain for numerical stability.

In order to apply a smoothing effect, we set the probability transition matrix $\mathbf{A}$ such that, for each scene, the probability of remaining in the same scene is p and the probability of transitioning to another scene is $\frac{1-p}{N-1}$. At the time step $t = 0$, we use the normal network prediction to initialize α, i.e. $\mathbb{P}(q_t = s_j \mid \mathbf{x})$. The HMM state is reset with every test clip.

In our experiments, we also investigate the usage of the *Viterbi algorithm* instead of the *forward algorithm* to smooth the predictions. The difference between the two is that *alpha forward* describes the probability of being in a specific state considering all previous possible state histories of the model (see (5.3)), while *Viterbi* estimates that probability by only taking into account the most likely sequence of states. The value modeled by *Viterbi* can be mathematically described as:

$$\delta_t(i) \triangleq \max_{q_1, q_2, \cdots q_{t-1}} \mathbb{P}(\mathbf{x}_1 \ldots \mathbf{x}_t,\, q_1, q_2 \ldots q_t = s_i \mid \lambda) \tag{5.7}$$

However, the computation of $\delta_t(i)$ requires processing a part or the whole sequence of inputs first, then applying a "backtracking" process to infer the most likely sequence of states. This would mean causing some latency in the prediction of the scene, which is not desirable. For this reason, we mainly focus on the analysis of the *alpha forward* algorithm. For a more in-depth discussion about *Viterbi*, see Appendix B.1.

5.2.3 Augmentation approaches: making the model more robust to noise through training

The approach presented in Section 5.2.2 only involves post-processing of the predictions produced by the neural network and is independent of the underlying model. In this section, we present possible methods to tackle the wind issue by

applying different augmentation techniques at training time. This way, we attempt to embed robustness directly into the CNN's learned weights instead of relying on a further post-processing step. Nevertheless, the augmentation and HMM approaches are not mutually exclusive and may be subsequently combined.

Popular general purpose augmentation techniques in ASC include mixup [79, 49, 129, 70] and channel swapping [79, 49]. However, they are not tailored to the specific issue of passive wind we are attempting to tackle. Moreover, in the image domain, techniques like CutMix [132] have been shown to produce models that are more robust to input corruption and to overconfidence on out-of-distribution samples with respect to mixup.[3]

The problem of passive wind can be roughly described as the network focusing too much on certain parts of the input spectrogram and neglecting the rest of the signal. Therefore, we choose to apply SpecMix [59], a technique that focuses on swapping frequency sub-bands and time frames of different input spectrograms, even those that belong to different classes. As the name implies, SpecMix can be considered as an adaptation of CutMix to the audio domain that is strongly inspired by SpecAugment. The original SpecMix technique described in [59] takes as input two spectrogram-label training couples $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$ and produces a new couple $(\tilde{\mathbf{x}}, \tilde{y})$ from them. The newly produced sample is used for training in place of the original input couples. The function that combines the two input samples is defined as

$$\tilde{\mathbf{x}} = \mathbf{M} \odot \mathbf{x}_A + (\mathbf{1} - \mathbf{M}) \odot \mathbf{x}_B \tag{5.8}$$

$$\tilde{y} = \lambda y_A + (1 - \lambda) y_B \tag{5.9}$$

where $\mathbf{M}$ is a binary mask produced using the same randomized rules of SpecAugment (see Section 3.3.2), and the coefficient λ is computed as

$$\lambda = \frac{\text{Number of pixels from } \mathbf{x}_A \text{ in } \tilde{\mathbf{x}}}{\text{Number of pixels in } \tilde{\mathbf{x}}} \tag{5.10}$$

In other words, two spectrograms $\mathbf{x}_A$ and $\mathbf{x}_B$ are merged into one by extracting random time frames and frequency bands from $\mathbf{x}_B$ and overlapping them over $\mathbf{x}_A$. The label for the newly generated sample is obtained by proportionally mixing the two original ground truths. This version of the algorithm halves the amount of training data during training, as two samples are used to generate a single new one: we modify the procedure so that the amount of training data is kept the same by generating an additional new training sample $(\tilde{\mathbf{x}}', \tilde{y}')$: to do so, we simply swap the pedices A and B in (5.8) and (5.9). A graphical representation of the technique

[3]For further insights about the mentioned augmentation techniques and why they were not suitable to tackle the passive wind issue, see Appendix B.2.

is shown in Fig. 5.4. When using SpecMix, we do not employ SpecAugment, as the two would clash with one another: while the former swaps regions of the input spectrograms, the latter simply zero-masks them. Zero-masking after applying the label smoothing described in (5.9) could make the ground-truth labels inconsistent, and therefore make it more difficult for the training procedure to converge. For example, SpecMix might produce a sample that is 20% *chatter* and 80% *travel*; however, if the zero-masking from SpecAugment is wide enough, the *chatter* part could undesirably be removed from the spectrogram, making the label essentially incorrect.

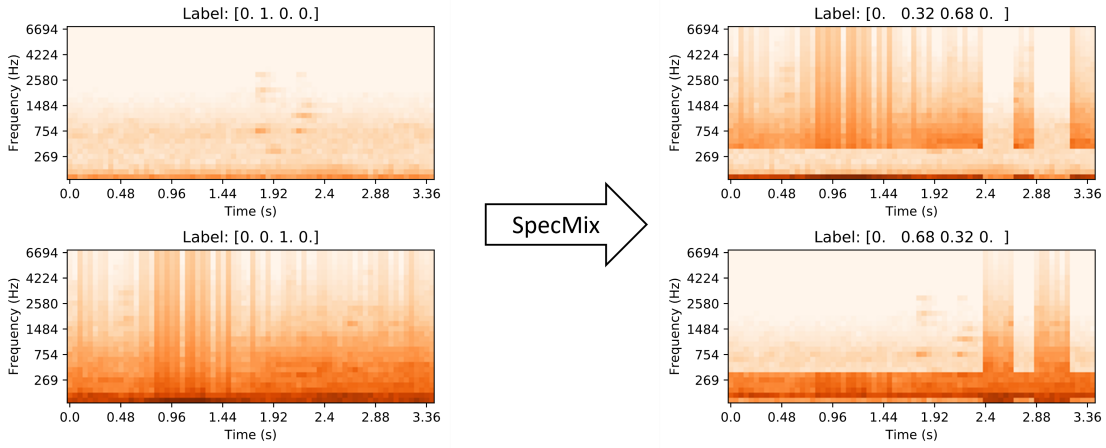


Figure 5.4: Example of SpecMix on two data points. On the right, time and frequency slices have been swapped between the two spectrograms, and their labels have been smoothed accordingly.

Again, the rationale behind the choice of SpecMix is that by swapping several different frequency bands of different inputs (potentially with different labels) the model will learn not to focus too much on the low-frequency region of the spectrogram. While this choice makes sense in this context, SpecMix is still just a general purpose technique that happens to fit our specific issue. Thus, we choose to also experiment with one further approach that is specifically tailored to the passive wind scenario: instead of using SpecMix, we directly superimpose passive wind noise over the input audio during training. This way, we encourage the model to learn representations that are independent of wind noise. The additive noise is produced in the way described in Section 5.2.1, with the exception that we randomly sample the SNR level to be between 0 and -20 dB. The noise clips used for augmentation are also recorded the same way as in Section 5.2.1.

Because our goal is to create a model that can correctly classify situations where passive wind can either be present or not, we refine this simple data corruption scheme by using an adversarial logit pairing technique. Adversarial logit pairing

(ALP) was introduced in [54] and originally consisted in achieving robustness against adversarial examples by encouraging the network to generate the same logits for a clean and an adversarial example during training. Adversarial examples [38] are neural network inputs to which an artificially crafted noise is added in order to maximize the probability of the network being wrong in its prediction. We modify the approach and adapt it to our needs by using wind-augmented clips instead of adversarial examples. This choice is loosely inspired by other techniques that propose to achieve domain shift robustness using adversarial training based on the natural variation of the data distribution rather than adversarial noise [102]. Let $\mathbf{x}$ be an input spectrogram, let $\mathbf{x}_n$ be the same sample with superimposed wind as in (5.1) and let $\mathbf{g}(\mathbf{x})$ be the logit vector produced by the neural network for a given input. ALP defines the following additional term for the loss:

$$\mathcal{L}_{alp}(\mathbf{x}, \mathbf{x}_n) = \|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}_n)\|^2 \tag{5.11}$$

Let $\mathcal{L}(\mathbf{x}, y)$ be a training loss of choice (cross-entropy loss plus some weight decay in our case). Then the new objective function to minimize becomes:

$$\mathcal{L}'(\mathbf{x}, y) = \mathcal{L}(\mathbf{x}, y) + \mathcal{L}(\mathbf{x}_n, y) + \lambda \mathcal{L}_{alp}(\mathbf{x}, \mathbf{x}_n) \tag{5.12}$$

Where λ is a manually tuned hyperparameter. In our experiments, we always use $\lambda = 0.5$ (in line with the value suggested by the authors of [54]). The rationale behind this formulation is that the network should learn to ignore disturbances coming from noise and classify noisy and non-noisy samples the same way.

We consider SpecMix and ALP to be mutually exclusive: because the additive noise augmentation takes place in time domain, it would have to be applied before SpecMix. Therefore, it would not make sense to apply additive noise to a sample, then swap some portions of its frequency representation with that of a different data point, and subsequently ask the network to produce the same logit for the augmented and the original signal. Therefore, we evaluate the two techniques separately.

5.2.4 Experimental results

We apply the techniques described in Section 5.2.2 and Section 5.2.3 keeping the same training hyperparameters. When using SpecMix during training, we apply it with a probability of 0.5.

Our experiments show that HMM-based methods improve the network's effectiveness in noisy scenarios up to -5 dB SNR. We find that the value of p that defines the $\mathbf{A}$ matrix does not make a significant difference as long as it is ≈ 0.7 or above; for lower values of p, the HMM does not noticeably change the baseline network's behavior. We report the results for $p = 0.9$. Because of its smoothing

effect, HMM post-processing seems to be beneficial for the model's accuracy in general. This is understandable: an acoustic scene generally does not change rapidly in a real context, and smoothing the output of the network is a reasonable way of making the model's predictions more reliable. More specifically, in the cases of the clean test set, 0 dB SNR and −5 dB SNR we get an F1-score increase of 7 to 10 percentage points in the case of *alpha forward* and 3 to 5 percentage points in the case of *Viterbi*. On the other hand, in high noise scenarios, the HMM seems to be detrimental to the overall performance. This is likely because if the network fails several times the hidden Markov model tends to encourage it to keep predicting the wrong scene. In other words, the smoothing is effective in correcting spurious misclassifications due to occasional bursts of passive wind, but will understandably start failing when the CNN's predictions are consistently wrong. Because of that, results with HMM are even worse than the baseline (sometimes dramatically) for wind noise levels of −10 dB SNR or lower.

Augmentation-based techniques have varying results. The use of SpecMix does not seem to critically impact model performance, at least in terms of raw F1-score. On the other hand, adversarial logit pairing has a sensible effect on how the neural network reacts to passive wind: a great boost in robustness is achieved in extremely noisy situations (+18 F1-score percentage points at −20 dB SNR with respect to the baseline), but the model is now less effective in clean or low-noise settings (-10 to -14 F1-score percentage points). In other words, ALP achieves good robustness at the expense of regular accuracy. See the left side of Fig. 5.5 for a graphical illustration of the results presented so far.

The phenomenon of trading regular model accuracy for increased robustness to certain kinds of perturbations has been studied in the literature especially in the context of adversarial examples [122, 134], which adversarial logit pairing is indeed related to. While passive wind perturbation cannot strictly be considered "adversarial" (being classified as such would require at least the involvement of some variation of a minimax objective [74, 122]), it is clear that a form of trade-off is still taking place: as the right side of Fig. 5.5 shows, using different training or post-processing techniques seems to allow the achievement of good performance on either clean scenarios or noisy scenarios, but not both. This phenomenon can be more easily visualized on the right side of Fig. 5.5, where we show that there is a negative correlation between the F1-score obtained on the clean test set and the test set at the maximum degree of noise.

We argue that the trade-off we are experiencing is not necessarily linked to the realm of adversarial examples: indeed, one of the most well-known problems in the machine learning field in general is the bias-complexity trade-off [109], sometimes referred to as fitting-generalization trade-off [25]. A simplistic way of illustrating the phenomenon is that small models are often easier to optimize but lack generalization capabilities (*underfitting*), while large models are capable of solving more complex

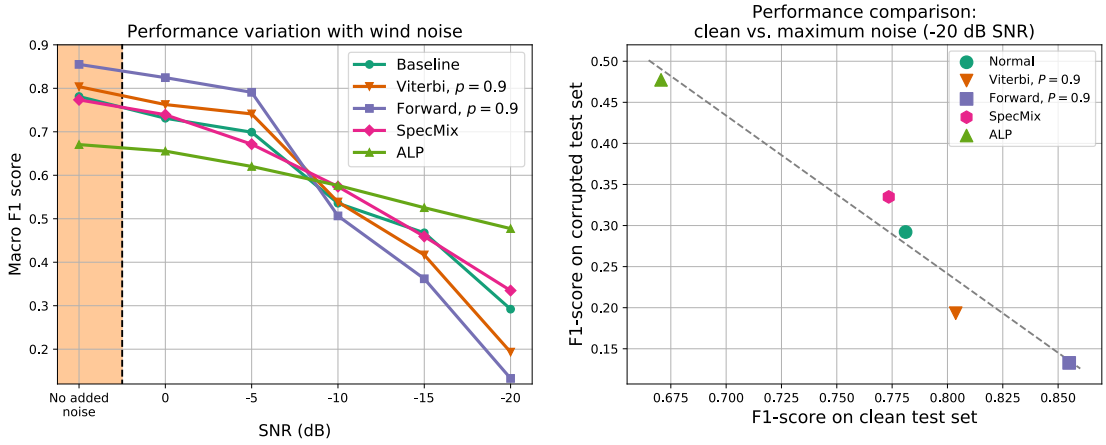


Figure 5.5: Evaluation of robustness techniques against passive noise. Left: performance comparison of different techniques at varying levels of noise. Right: comparison between performance on noiseless test set and performance at maximum noise level (−20 dB SNR).

problems but they are more likely to be poorly trained (*overfitting*).[4] More formally, it is possible to express the bias-complexity trade-off as the decomposition of the risk difference between an hypothesis h' chosen by means of some optimization algorithm and the Bayes risk [84]:

$$\mathcal{R}(h') - \mathcal{R}^* = \underbrace{\left(\mathcal{R}(h') - \inf_{h \in \mathcal{H}} \mathcal{R}(h)\right)}_{\epsilon_{est}} + \underbrace{\left(\inf_{h \in \mathcal{H}} \mathcal{R}(h) - \mathcal{R}^*\right)}_{\epsilon_{app}} \tag{5.13}$$

The first term ϵ_{est} is named *estimation error* and depends both on the optimized h' and the hypothesis class; the second term ϵ_{est} is called *approximation error* and only depends on the hypothesis class $\mathcal{H}$. When a model is simple, it is usually easier to optimize it to the best of its capability, i.e. to obtain a small difference between $\mathcal{R}(h')$ and $\inf_{h \in \mathcal{H}} \mathcal{R}(h)$, thus reducing the estimation error; however, the model may not be expressive enough to effectively describe the analyzed phenomenon, resulting in high approximation error (possibility of *underfitting*). On the other hand, more complex models have a "wider" hypothesis class and thus a potentially lower approximation error; however, reducing their ϵ_{est} is harder because of the more complex optimization search space $\mathcal{H}$ (possibility of *overfitting*).

[4] It is worth mentioning that some studies argue that the trade-off between robustness to adversarial examples and generalization capability is simply a manifestation of the bias-complexity trade-off: allegedly, neural networks can either be adversarially robust or generalize well, but not both [25, 95].

As previously remarked, the model we are working on is somewhat simple: one might then argue that it is easy to optimize the neural network to perform well on the clean test scenario (diminish ϵ_{est}) but hard to make it generalize to the noisy scenario (because of ϵ_{app}) or vice-versa. One obvious, yet reasonable approach to the problem might be to increase the dimension of the hypothesis class, i.e. to choose a larger CNN; however, as remarked in Section 2.2, the project was limited in that regard for most of its duration. Further insights about possible changes of the hypothesis class will be explored in Chapter 6.

Because the neural network cannot be changed, one naive option to try and further increase effectiveness on noisy scenarios without diminishing it too much on the clean test set is to manually adjust the transition probability matrix of the HMM to make it tailored to our specific passive wind problem. Because we know that the effect of passive wind is to make the network output the *travel* class, we can use that knowledge to our advantage to discourage the model from predicting it. More specifically, we bias the matrix $\mathbf{A}$ such that the value $\mathbb{P}(q_t = \text{travel} \,|\, q_{t-1} = s_i)$ is smaller than the probability of transitioning to any other scene for every s_i. This also holds for $s_i = \textit{travel}$; in other words, we empirically find it effective to make it less likely to remain in the *travel* scene once we are in it than to transition to another scene. We establish that the following values for a wind-specific $\tilde{\mathbf{A}}$ transition matrix give good results:

$$\tilde{\mathbf{A}} = \begin{bmatrix} 10 & 5 & 1 & 5 \\ 5 & 10 & 1 & 5 \\ 5 & 5 & 3 & 5 \\ 5 & 5 & 1 & 10 \end{bmatrix} \tag{5.14}$$

The matrix is subsequently normalized so that each row sums to 1 to make each probability distribution valid. From 1 to 4, the row/column indices represent the classes *chatter*, *quiet*, *travel* and *vehicle* respectively. In addition to this, we try to combine the hidden Markov model with each augmentation technique, since post-processing and training-time augmentations are not mutually exclusive. We find that the use of SpecMix along with the HMM with custom $\tilde{\mathbf{A}}$ matrix results in a good increase in robustness to noise without trading too much effectiveness on the clean data.

In general, the interesting aspect of biasing the values of $\mathbf{A}$ is that such an approach seems to partially bypass the bias-complexity trade-off by completely raising the F1-score curve presented on the left side of Fig. 5.5 without compromising on either extreme. In other words, using $\tilde{\mathbf{A}}$ appears to improve the results in the noisy setting without making things worse in the clean setting (see Fig. 5.6). However, we argue that this is not due to an "absolute" enhancement of the model or an increase in generalization capabilities: rather, we have found a way to bias the model even more precisely to elude the specific problem of passive wind by

injecting human-crafted knowledge into it. If we were to change the acoustic scenes to classify or to try to counteract a different class-specific issue, this method would probably not be effective: it would be necessary to re-tune the values of $\tilde{\mathbf{A}}$ according to the new configuration of the system. Nevertheless, the combination of SpecMix and the custom-tuned HMM currently seems to be the best trade-off between passive wind robustness and acceptable accuracy on clean input, and we therefore decide to keep this version of the model for subsequent experiments.

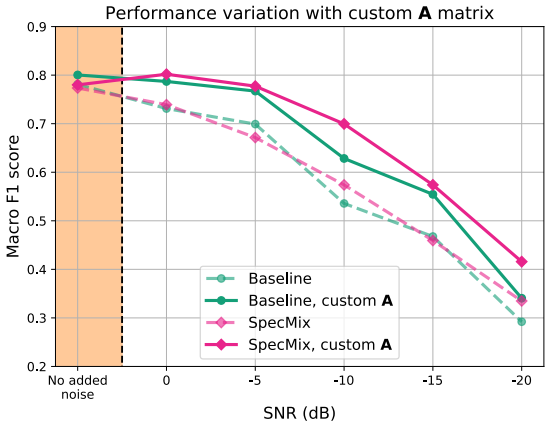


Figure 5.6: Evaluation of robustness against passive wind noise by combining SpecMix with the hidden Markov model and manually tuning the values of $\mathbf{A}$ for the specific problem of wind.

As a side-note, one might argue that the ability to choose which model to use according to the amount of noise in the input would considerably improve the overall performance, as we could then tune the values of $\mathbf{A}$ according to our needs: as our experiments have shown, using $p = 0.9$ works very well on clean settings, while the custom $\tilde{\mathbf{A}}$ is effective in noisy environments. A few techniques to estimate the amount of passive wind noise in a recording were considered as potential methods to put into practice the idea described above. We do not illustrate them in detail for patent-related reasons; nevertheless, we lacked the time to fully explore their potential, and they do not constitute a fundamental part of this work.

5.3 Evaluating out-of-distribution robustness

The empirical risk minimization objective expressed in (1.4) is designed to train a model $h(\theta, \mathbf{x}) = y$ that is optimal to explain the patterns found in a dataset $D \subset \mathcal{X} \times \mathcal{Y}$. This makes an implicit assumption that the training set D follows a certain probability distribution $\mathbb{P}(\mathcal{X}, \mathcal{Y})$. However, it is not guaranteed that the set of data D_{test} the model will be tested on after deployment will behave according to

the same probability distribution: test data may be corrupted in unforeseen ways, the training set may not have been entirely representative of the stimuli the model will be exposed to, or usage trends will simply change. This phenomenon is known as *concept drift* or *data drift* [124], and has recently received attention within the context of maintenance of ML models applied in production systems [136, 133, 69]. In general, is it possible to divide concept drift in two sub-categories according to what differentiates the training from the testing data distribution [72, 34]:

- A change in the relationship between input features and labels (i.e. $\mathbb{P}(\mathcal{X}, \mathcal{Y}) \neq \mathbb{P}_{test}(\mathcal{X}, \mathcal{Y})$);
- A change only in the input feature distribution (i.e. $\mathbb{P}(\mathcal{X}) \neq \mathbb{P}_{test}(\mathcal{X})$), while the mapping between input and label remains the same.

The latter event is sometimes referred to as *covariate shift* or *out-of-distribution* (OOD) *shift* [110], and it is the one of most interest in our investigation: as remarked in Section 2.2, the complexity of the world makes it impossible to think one can capture every possible acoustic scene that may be mapped to a certain set of labels. In other words, our training data will likely never include all possible acoustic descriptions of the high-level labels *chatter*, *quiet*, *travel*, *vehicle*. We therefore seek a way to evaluate how robust the trained model is to out-of-distribution shifts by evaluating it on a test set that contains different underlying acoustic scenes ($\mathbb{P}(\mathcal{X}) \neq \mathbb{P}_{test}(\mathcal{X})$) that are mapped to the same four macro-classes ($\mathbb{P}(\mathcal{X}, \mathcal{Y}) \approx \mathbb{P}_{test}(\mathcal{X}, \mathcal{Y})$).

We design the new test set by using data from a further past edition of the DCASE challenge. More specifically, we employ the TUT Acoustic Scenes 2017 development dataset [81], which was also originally used for an acoustic scene classification task. The difference between TUT Acoustic Scenes 2017 and the previously described TAU Urban Acoustic Scenes 2019 is that the former has a different set of acoustic scenes than the ones used thus far to train and evaluate models. The new scenes are listed in the first column of Table 5.1.

Of course, as remarked in Section 4.2, the device mismatch analysis was carried out with a Soundman OKM II as source microphone: thus, further benchmarks make sense only if the test data was recorded with the same device.[5] The data in TUT Acoustic Scenes 2017 was recorded with two devices: a Soundman OKM II and a Roland Edirol R-09: keeping only the audio files produced with the former, we are left with only a subset of the scenes listed in Table 5.1. We re-map the remaining scenes into the four macro labels according to their audio content and a guess of the expected prediction of the model based on the original acoustic scene's

[5]Or from the target microphone of the *BeoMusic*. But, as mentioned several times throughout this work, no sizeable amount of recording is available from it.

Original scene	**Re-mapped label**
Bus	Removed (overlap with TAU 2019)
Cafe / Restaurant	*chatter*
Car	*travel*
City center	*vehicle*
Forest path	*quiet*
Grocery store	*chatter*
Home	Removed (too ambiguous to label)
Lakeside beach	Removed (too ambiguous to label)
Library	Removed (wrong device)
Metro station	Removed (overlaps with TAU 2019)
Office	*quiet*
Residential area	Removed (too ambiguous to label)
Train	Removed (wrong device)
Tram	Removed (overlaps with TAU 2019)
Urban park	Removed (overlaps with TAU 2019)

Table 5.1: Classes from TUT Acoustic Scenes 2017 and their re-mapping into the new test dataset. Green indicates that the class was used in the new test set.

name. The new mapping is illustrated in the second column of Table 5.1. Because we wish to assess the model's ability to adapt to unseen acoustic scenes, we remove the scenes that TAU Urban Acoustic Scenes 2019 and TUT Acoustic Scenes 2017 have in common. As with the previous test set, we join clips coming from the same recording instance forming longer test audio clips of 2 to 4 minutes; we then split right and left channels in separate mono files and downsample them from their original 44.1 kHz sampling rate to 16 kHz. The result is a new test set with a total of 124 audio clips, among which:

- 40 are labeled as *chatter* ($\approx$ 2h 9m)
- 48 are labeled as *quiet* ($\approx$ 2h 30m)
- 22 are labeled as *travel* ($\approx$ 1h)
- 14 are labeled as *vehicle* ($\approx$ 53m)

We name the new dataset "out-of-distribution" (OOD) test set. Note that the OOD test set is smaller than the the previous test set, it is less balanced, and it also contains some arguably harder scenes to classify: for example, *city center* contains vehicles but also occasional chatter, and *grocery store* often has sounds of supermarket fridges that emit loud low-frequency noise and make it hard to recognize the scene even for a human (without the prior knowledge of what the scene is).

We evaluate the models trained thus far on the new test set. Fig. 5.7 shows the macro averaged F1-score achieved by each model plotted against the same metric obtained in the original test set. We find this visualization useful since we wish to find a model that can accurately describe both datasets and does not just happen to fit well only one or the other.

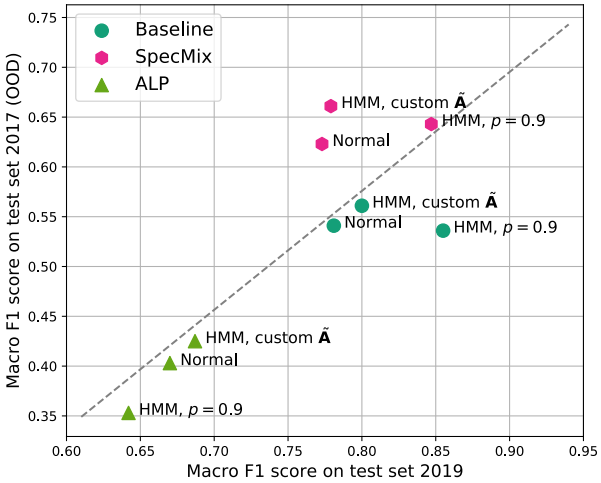


Figure 5.7: Macro F1 score of the different versions of the ASC model on the new OOD test set plotted against the scores on the normal test set. Where just "HMM" is indicated, we use a the $\mathbf{A}$ matrix with $p = 0.9$ as described in Section 5.2.2.

Some studies argue that the performance of a set of models on a OOD test set can often be predicted by a linear fit of their performance on the in-distribution test set [121]. The fitted line usually lies below the identity $y = x$, as most models perform worse on the OOD test set. The models that lie noticeably above the fitted line may be considered the most "robust". According to this criterion, the use of SpecMix combined with the HMM seems to yield the best results, as all the scores produced with it lie above the linear fit. A custom $\tilde{\mathbf{A}}$ matrix seems to be the most effective trade-off between performance and robustness, while $\mathbf{A}$ with $p = 0.9$ provides the greatest clean test set accuracy without losing too much performance on the OOD set.

In general, the effect of the HMM seems similar both in the case of the baseline model and the usage of SpecMix: $\mathbf{A}$ with $p = 0.9$ especially improves performance in the normal test set, while $\tilde{\mathbf{A}}$ increases robustness. This effect can be explained by the fact that the use of custom $\tilde{\mathbf{A}}$ is specifically targeted towards avoiding false positives of the *travel* class, which also happens to be the main kind of misclassification that takes place in the OOD test set. As previously mentioned,

this is due to the fact that a lot of samples from the *chatter* class include low-pitched background noise, such as refrigerators in the *grocery store* class or even air conditioning in the case of the *cafe/restaurant* class. However, this is not helpful in counteracting other kinds of frequent misclassifications, such as *quiet* samples being mistaken for *chatter* samples (especially frequent in the *forest path* class, likely because of bird sounds and occasional water courses). We conclude that the improvements introduced by the hidden Markov model are probably not critical in terms of robustness, and what little benefit is brought in that regard is somewhat situational to this particular OOD test set. Fig. 5.8 shows how the presence of the HMM impacts OOD robustness for each class both in the baseline scenario and with SpecMix.

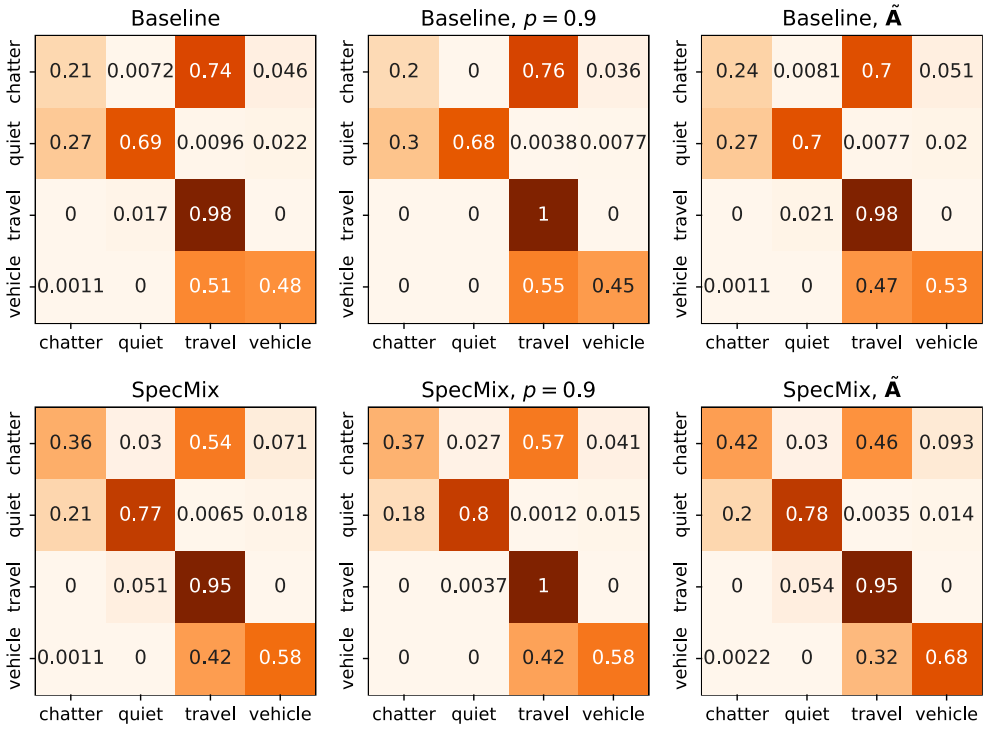


Figure 5.8: Confusion matrices of the baseline model and the model trained with SpecMix, evaluated on the out-of-distribution test set and postprocessed with two different HMM configurations.

On the other hand, as shown in Fig. 5.7, SpecMix seems to be generally beneficial for the overall OOD robustness of the model, resulting in about 10 percentage points increase in F1-score regardless of the kind of applied post-processing.

5.4 Evaluating robustness to impulse response angle

It was highlighted in Section 4.2.2 that the baseline CNN was sensitive to impulse responses of the target microphone coming from different angles; in this section, we verify how the newly trained neural network behaves in the same scenario. We focus on the SpecMix version of the model, as it seems to yield the best results overall so far. We convolve the audio signals of the new (in-distribution) test set with the 12 impulse responses presented in Section 4.2.2 and evaluate the baseline CNN and the SpecMix CNN on the newly generated datasets. Note that these are not produced from the 30-minute comparison sets from Section 4.2.2, but rather from the new test set described in Section 5.1. This allows comparison with the other results presented in this Section. Because our previous experiments showed that the position of the head during the recording of the sweep and the playback volume of the sweep itself have no significant impact, we only experiment with one impulse response configuration (position 1, loud setting as in the top left of Fig. 4.7).

Interestingly enough, SpecMix appears to have a beneficial effect in terms of robustness to impulse response angles: this is further confirmation of the general effectiveness of this augmentation technique. As observed in Section 5.2.4, a hidden Markov model on top of the neural network with $p = 0.9$ will improve the score where performance is already somewhat good (> 0.7, in this case) and will worsen it elsewhere. Conversely, using the previously defined $\tilde{\mathbf{A}}$ does not seem to have any noticeable effect on the classification performance (Fig. 5.9). This is again because $\tilde{\mathbf{A}}$ was designed to counteract the effect of passive wind and therefore to avoid the abuse of the *travel* class; however, the effect of the impulse responses is to bias the CNN mostly towards the *chatter* class, both in the case of the baseline network (Fig. A.3) and the network trained with SpecMix (Fig. A.4). One might then argue that it would be sufficient to design a further transition probability matrix that prevents the model from abusing both the *travel* and the *chatter* class; however, we feel this is not the right approach to the problem, as it would simply be an even more specific workaround that would likely be ineffective or even harmful with other kinds of test data. HMM-based post-processing might be acceptable to stabilize the network's outputs and avoid fluctuating predictions, or to tackle very general issues such as passive wind (which would take place in a practical scenario regardless of the underlying scenes to classify); conversely, we find the issue of angular impulse responses to be too specific and situational to be solved with post-processing. Attempting to embed robustness to the reverberations directly into the neural network's weights would be a sounder approach.

One straightforward option to try to achieve that would be to let the network

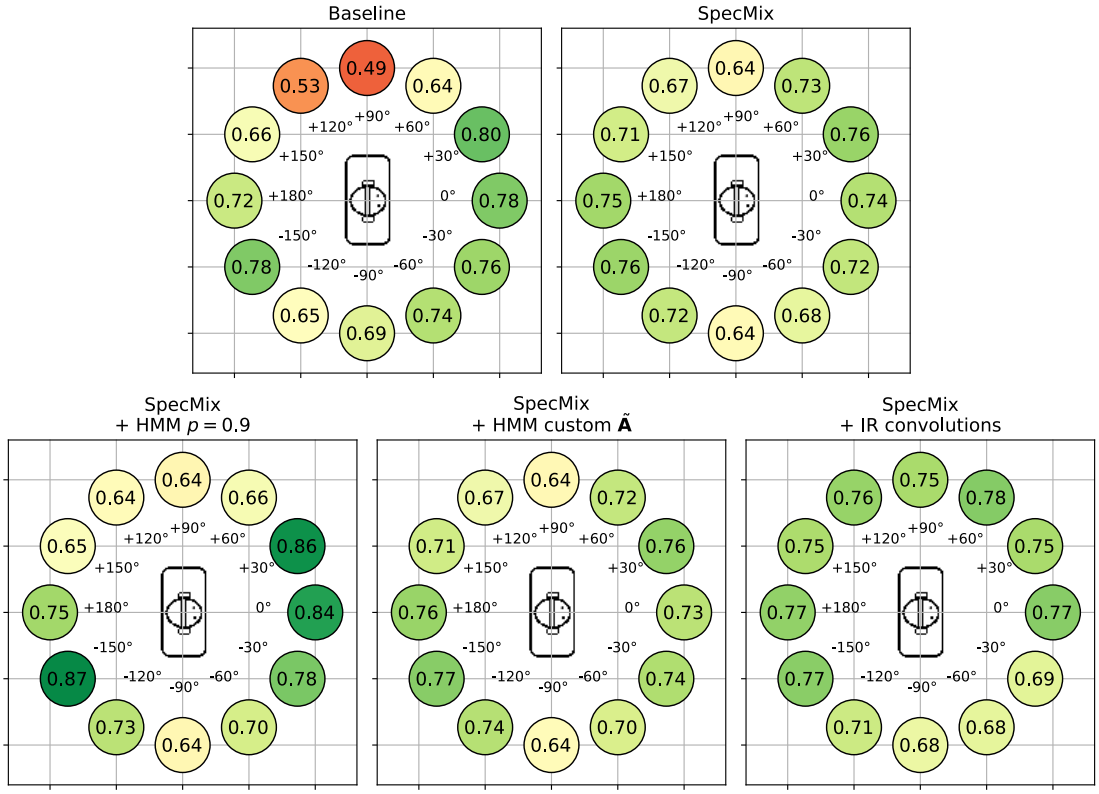


Figure 5.9: Comparison of the model's behavior under different impulse response directions.

know about the possible distortions caused by the impulse responses at training time. To do so, we retrain the CNN with the same SpecMix configuration, but this time we convolve each input clip with random impulse responses. More specifically:

- Each training audio clip has a probability of 0.7 to be selected for convolution with an impulse response;
- If that happens, one out of the 12 impulse responses (from position 1, loud setting) is randomly selected with uniform probability to be convolved with the input signal as in (4.4). Afterwards, the RMS of the result is normalized to that original audio clip as in (4.2).

This can be considered a form of augmentation in and of itself: indeed, the approach of augmenting the data with random reverberations by means of convolution has been used in the literature to increase the generalization capabilities of neural networks [100]. As the bottom right chart of Fig. 5.9 illustrates, this training approach can improve performance on the convolved test sets. This new model also scores around 77% macro-averaged F1-score on the normal test set, which is

in line with the previous SpecMix version of the model; however, performance on the OOD test set drops to around 53%, a lower value than the 62% achieved by the plain SpecMix-trained CNN.

Again, we seem to face the same dilemma illustrated in Section 5.2.4: when using a model with this level of complexity, it is possible to achieve decent robustness to one stimulus, but this often comes at the cost of decreased performance on a different kind of disturbance. The following was then concluded: with the current front-end preprocessing and back-end SB-CNN classifier, the SpecMix + $\tilde{\mathbf{A}}$ HMM setup provides a good tradeoff between regular ASC capability, robustness to passive wind and to out-of-distribution data shifts. Further efforts were put into understanding the limitations of the current network topology $\mathcal{H}$ and data representation: we would like to find out if modifying them can help in surpassing the trade-offs faced so far. We investigate the matter in the next Chapter.

Chapter 6

Impact of different back-end and front-end approaches

As the experiments described in Chapter 5 were being carried out, a technical change in the hardware configuration of the platform made it possible to experiment with new neural network architectures. This seemed to represent a valid research direction after having analyzed the various trade-offs encountered while training the SB-CNN. Naturally, to pursue this path, an inspection of the possibilities offered by the current state of the art in acoustic scene classification was necessary: the natural starting point was the 2021 edition of the DCASE challenge, whose *subtask A* specifically focuses on low-complexity models [77]. According to the rules of the challenge, the level of complexity of a neural network is purely based on the disk size of its non-zero parameters; neither the number of computations the network executes when processing the input features nor the complexity of the front-end preprocessing are taken into account. Thus, this concept of "low-complexity" does not necessarily align with the needs of an embedded application.

In spite of that, we inspect several top-performing systems of the DCASE competition and list the most popular complexity-reduction approaches employed by the participants of the challenge. We give a brief description of each technique along with some considerations on their potential usage within the scope of our project.

- **Pruning**: zeroing out unimportant parameters of the network [17, 58, 64, 130]. In the context of the DCASE 2021 challenge, this technique is used to make the network sparser and more easily compressible, with the ultimate goal of reducing its final disk usage. However, even with specific sparse matrix multiplication implementations, sparser models do not necessarily perform a lower amount of computations, unless the pruning follows specific hardware-dependent rules [131]. Because we are interested in a low amount of

computation other than storage size, we focus on different techniques.

- **Distillation**: transferring knowledge from a large, powerful neural network to a smaller, less complex one [44, 58, 130]. This can be achieved in several ways, e.g. by forcing the smaller model to imitate the output probabilities of its larger counterpart for a given training sample [46]. This technique does not involve compression of the final network architecture, but rather relies on the availability of a bigger pre-trained model specialized on the same final task of its slimmer version. Unfortunately, such a model was not readily available in our case, as our label set is different from that of the original DCASE challenge. Therefore, fine-tuning of some pre-existing large model would have been necessary: this would have required additional time. Moreover, at this point of the project, the interest had shifted to changing the actual network topology itself. For those two reasons, we looked for alternative approaches.
- **Quantization**: reducing the complexity of the numerical representation of the network's weights [17, 44, 58, 130]. This is a viable option, but it is independent from the neural network topology. It may be considered for future work.

Aside from those three popular techniques, one of the top-performing systems of the challenge employs a neural network architecture that includes depthwise-separable convolutions: the authors name it BC-ResNet-Mod [58]. As remarked in Section 2.3.2, our SB-CNN already makes use of depthwise-separable convolutions, which have a long history of being employed in low-complexity models [47]. We decide to keep exploring this direction and select BC-ResNet-Mod as new CNN topology $\mathcal{H}$. In the next section, we describe the architecture of the network.

6.1 BC-ResNet-Mod architecture

BC-ResNet-Mod is based on the BC-ResNet neural network, which was originally designed for efficient keyword spotting [56]. Its authors argue that the main feature of BC-ResNet is the *broadcasted residual* operation, which achieves low computational complexity by separately operating on the frequency and time dimensions.

In this section, we describe the building blocks of the BC-ResNet-Mod neural network. We start from a special kind of normalization layer used in the architecture called subspectral normalization; subsequently, we discuss the original BC-ResNet model; finally, we illustrate the differences between BC-ResNet and BC-ResNet-Mod.

6.1.1 Subspectral normalization

Before describing the broadcasted residual operation, it is necessary to illustrate the functioning of one of the inner components of BC-ResNet-Mod: subspectral normalization (SSN) [18]. SSN is a form of batch normalization (BN) [53] that divides the input feature mapping in S sub-bands along the frequency dimension, then individually normalizes each sub-band: the goal is to independently normalize the distribution of different portions of intermediate features in the frequency dimension, as different frequency bands usually convey different kinds of information.

More specifically, let $\mathbf{x} \in \mathbb{R}^{F \times T}$ be an input feature channel,[1] and let $\mathbf{x}_i \in \mathbb{R}^{F_i \times T}$ be the i^{th} subspectral band, i.e. a subset of the input extracted along the frequency dimension. Usually, all S subbands are chosen to be equally sized by setting $F_i = F/S$ for all i. SSN separately normalizes each subspectral band by independently applying batch normalization to it:

$$\tilde{\mathbf{x}}_i = \gamma_i \cdot \frac{\mathbf{x}_i - \mu_i}{\sigma_i} + \beta_i \tag{6.1}$$

where γ_i and β_i are learnable parameters, and μ_i and σ_i are established the same way as batch normalization (see [53]). All vector-scalar operations are broadcasted. If $S = 1$, then subspectral normalization becomes equivalent to regular batch normalization.

6.1.2 BC-ResNet

We now describe the fundamental building block of the BC-ResNet model: the *broadcasted residual* operation. Given an input feature mapping $\mathbf{x} \in \mathbb{R}^{F \times T \times C}$, a depthwise-separable convolution is applied only to the frequency dimension using $k_F \times 1$ shaped kernels, extracting features related to the spectrum in a single time-frame. We denote this operation as $f_{frq}(\mathbf{x}) : \mathbb{R}^{F \times T \times C} \mapsto \mathbb{R}^{F \times T \times C}$ (padding is applied to preserve the original tensor shape). Afterwards, a single temporal feature is obtained for each time-frame by average-pooling the result over the frequency dimension, obtaining a tensor $\mathbf{z} \in \mathbb{R}^{1 \times T \times C}$. This is intended to be a high-level representation of all frequency information in a single time-frame. Temporal variation of the features is then taken into account by applying a further depthwise-separable 1D convolution on the obtained coefficients, which we denote as $f_{tmp}(\mathbf{z}) : \mathbb{R}^{1 \times T \times C} \mapsto \mathbb{R}^{1 \times T \times C}$ (padding is used again to preserve input shape). This mechanism is paired with the popular residual learning approach [41], where

[1] We provide a channel-wise description since both BN and SSN operate independently over each channel.

skip connections are inserted between convolutional blocks. In [56], skip connections are used to sum both the input $\mathbf{x}$ and the intermediate mapping $f_{frq}(\mathbf{x})$ to the output of the temporal convolution $f_{tmp}(\mathbf{z})$. However, this would normally result in a shape mismatch, since $\mathbf{x}, f_{frq}(\mathbf{x}) \in \mathbb{R}^{F \times T \times C}$ but $f_{tmp}(\mathbf{z}) \in \mathbb{R}^{1 \times T \times C}$. Thus, the latter tensor is expanded back to $\mathbb{R}^{F \times T \times C}$ by applying the broadcasting operation to the frequency dimension (hence the name *broadcasted residual*). Overall, it is possible to formally express the broadcasted residual operation as:

$$y = \mathbf{x} + f_{frq}(\mathbf{x}) + \mathrm{BC}\left(f_{tmp}\left(\mathrm{AvgPool}\left(f_{frq}(\mathbf{x})\right)\right)\right) \tag{6.2}$$

Where BC denotes the broadcast expansion. A graphical illustration of the operation is given on the left side of Fig. 6.1.

Overall, the idea of this building block is to reduce the amount of network parameters and computations by splitting the convolution operation in two: one for the frequency dimension, and one for the time dimension. This allows to reduce the size of convolutional filters and therefore the number of multiplications performed per convolutional layer. This approach makes sense when used with time-frequency data representations, as the two axes of the input have different interpretations; the same consideration would not apply to other kinds of data type, e.g. images. One might argue that the average pooling layer between f_{frq} and f_{tmp} could result in a loss of information; however, the presence of the residual connection guarantees that more granular input stimuli are preserved, while also providing the well-known advantage of strengthening the gradient flow during backpropagation and thus counteracting the issue of vanishing gradients.

The BC-ResNet architecture is primarily composed of two building blocks based on the broadcasted residual operation: a *normal block* and a *transition block*. The former is a transformation that preserves the shape of the input tensor, while the latter changes the number of channels of the input and optionally downsamples it over the frequency dimension. In both the normal and transition blocks, f_{frq} and f_{tmp} are enriched with a number of supplementary operations in order to make them more suitable to be used in a neural network. In a normal block, the frequency convolution in f_{frq} is followed by a subspectral normalization, while the temporal convolution in f_{tmp} is followed by a batch normalization, a swish activation [98], a 1×1 convolution, and a dropout layer [116]. In a transition block, the frequency convolution in f_{frq} is preceded by a 1×1 convolution that performs the change in the number of channels, and followed by batch normalization and a ReLU activation; the frequency convolution itself is strided by a factor s to perform downsampling in the frequency domain; the temporal convolution in f_{tmp} is followed by a batch normalization, a swish activation [98], a 1×1 convolution, and a dropout layer [116]; the skip connection between input and output is removed. In both normal and transition blocks, a ReLU is applied as final operation. The two blocks are illustrated on the right side of Fig. 6.1. We refer to a set of n

consecutive normal or transition blocks as *BC-ResNet block*.

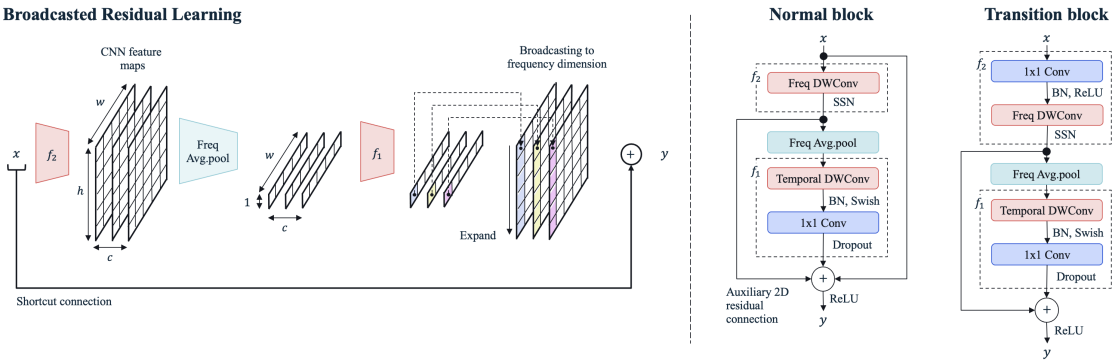


Figure 6.1: On the left: the broadcasted residual operation. On the right: the two main building blocks of the Bc-ResNet architecture. Diagram taken from [56] with permission from the authors. In our notation, f_{frq} is f_2 and f_{tmp} is f_1.

6.1.3 BC-ResNet-Mod

We now illustrate the difference between the BC-ResNet model described in Section 6.1.2 and BC-ResNet-Mod, which is the model we use in our experiments.

As previously mentioned, BC-ResNet was originally designed for the task of keyword spotting. In order to adapt the architecture to acoustic scene classification, the authors of [58] propose a modification to the original BC-ResNet: instead of using strided convolutions to perform downsampling, they use max pooling. This choice is motivated by a number of studies that have shown that the size of the receptive field of a neural network back-end seems to be a relevant regularizing parameter when tackling the ASC task [62, 63]. The authors of [62] argue that pooling is a straightforward non-parametric way of performing downsampling and therefore tuning the receptive field of a neural network. To get an initial 3D tensor feature mapping from the input spectrogram, a normal 2D convolution is performed as first operation. The overall model has two hyperparameters: the number S of frequency subbands used in SSN and the number of initial channels c produced by the first convolutional layer (i.e. the number of filters in the first layer). The final architecture of the model is reported in Table 6.1: that is what we refer to as BC-ResNet-Mod from now on.

Overall, the goal of both BC-ResNet variants is to construct a very deep network while using a reduced number of parameters and few computations. Aside from depthwise-separable convolutions and frequency-aware pooling, the authors also avoid using a final fully-connected layer, as those usually hold a high number of

Input shape	Operator	n	Output channels
$F \times T \times 1$	conv2d 5×5, stride 2	-	2c
$F/2 \times T/2 \times 2c$	stage1: BC-ResBlock	2	c
$F/2 \times T/2 \times c$	max-pool 2x2	-	-
$F/4 \times T/4 \times c$	stage2: BC-ResBlock	2	1.5c
$F/4 \times T/4 \times 1.5c$	max-pool 2x2	-	-
$F/8 \times T/8 \times 1.5c$	stage3: BC-ResBlock	2	2c
$F/8 \times T/8 \times 2c$	stage4: BC-ResBlock	3	2.5c
$F/8 \times T/8 \times 2.5c$	conv2d 1x1	-	N. classes
$F/8 \times T/8 \times$ N. classes	global avg pool	-	-
$1 \times 1 \times$ N. classes	-	-	-

Table 6.1: Basic BC-ResNet-Mod architecture for a generic input shape $F \times T \times 1$ and initial filters c (adapted from [58]). **n** indicates the number of blocks within each BC-ResNet block; when the number of input channels and output channels differ, the first sub-block is a transition block, all other sub-blocks are normal blocks.

parameters. Instead, the final class logits are obtained by properly tuning the number of channels in the final feature mapping and using global average pooling to shrink down the time and frequency dimensions to 1: the final $1 \times 1 \times N_{\text{classes}}$ tensor represents the logits.

6.2 Experimental setup

In order to isolate the effect of the new network topology over the experiments, we benchmark BC-ResNet-Mod against the baseline training setup presented in Section 3.3.3. After training, we test the CNN on the normal test set, its various noisy versions and the OOD test set introduced in Chapter 5. We set $c = 10$ to match the hyperparameters of the smallest architecture presented in [58]. Since our preprocessing includes only 30 Mel bins, the frequency resolution values across the layers of the network are 15, 7, and 3 (odd dimensions are rounded down). Because of the small amount of frequency coefficients, we initially decide not to divide the normalization into subbands and set $S = 1$. The dropout probability is always 0.1. This yields a final network of 7685 parameters, a size comparable to that of the baseline SB-CNN; however, BC-ResNet-Mod is much deeper, having more than double the layers. This allows us to keep the same memory requirements of the previous CNN, while also establishing if the depth of the network has a beneficial effect on the classification performance.

Because of the obtained results from the previous experiment and in order to

explore the effect of changing the front-end processing as well as the back-end, we subsequently investigate the use of an input spectrogram of higher time and frequency resolution. We choose to match the front-end parameters of the original paper [58]: we employ 256 Mel bins computed from an FFT window of 2048 coefficients (128 ms)[2] and a hop length of 480 samples (30 ms); we include more contextual information in a single sample by processing 10 seconds of audio per spectrogram. This results in a Mel spectrogram of size of 256×330. In order to separate the effect of the input resolution from that of the network topology, we also experiment with the new front-end parameters on the previous SB-CNN model; however, in order to do that, we need to make a modification to the old SB-CNN architecture. If we left it unchanged, the final feature mapping before the flatten layer would be in $\mathbb{R}^{6\times5\times48}$, which would in turn result in a 1440-dimensional flattened feature space. Because the size of the two final fully-connected layers would then be too large, we introduce a further 1×1 convolution before the flatten operation that compresses the feature mapping size from $\mathbb{R}^{6\times5\times48}$ to $\mathbb{R}^{6\times5\times3}$. The network then proceeds as originally described. The parameter count of the new SB-CNN version is 9845: we find this to be a good compromise between increase in memory usage (the original SB-CNN had 8354 parameters) and similarity to the original SB-CNN achitecture. We refer to this modified version as SBCNN-HR (as in "High Resolution"). Moreover, because of the higher number of frequency bands, it is now possible to experiment with more subbands in the subspectral normalization layers. We test two versions of the BC-ResNet-Mod: one that keeps $S = 1$ as in the previous case, and one that uses $S = 4$ as in [58].

6.3 Results

When using BC-ResNet-Mod with the original input resolution of 30×72, results did not improve in any noticeable way with respect to the baseline SB-CNN. On the clean test set, we obtain a macro-averaged F1-score of 79%: only a slight improvement over the 78% achieved by the baseline. Moreover, the model does not seem to exhibit a greater robustness to moving wind (Fig. 6.2a), and its OOD performance actually shows a noticeable degradation (Fig. 6.2b).

We try to hypothesize what may be the reason of this lack of improvement, considering that the BC-ResNet-Mod model has achieved state of the art in the ASC task in the DCASE challenge 2021. While it is true that the new network is deeper and can theoretically learn more complex acoustic representations, our input signal is still highly compressed, using only 30 Mel bands and 72 time frames.

[2]To be exact, the authors report a window of 130 ms, which would actually correspond to 2080 samples. For computational speed of the FFT, we reduce that to 2048.

Because the initial time-frequency representation is fairly simple, a deeper model might fail to properly generalize on it. We therefore decide to expand the input resolution as reported in Section 6.2. In doing so, we also introduce the new SBCNN-HR variation to the test setup.

Using 256 Mel bins noticeably improves performance over the clean test set. All models achieve between between 83% and 86% macro-averaged F1-score, including the SBCNN-HR: that is an increase of about 5-6 percentage points with respect to the previous setup. This result seems to suggest that a change of preprocessing is more impactful than a change of CNN topology: this is presumably due to the fact that we preserve more information from the original audio input, which no neural network would ever be able to exploit regardless of its size. Of course, this also implies a greater memory usage: with the new front-end configuration, the input tensor has a higher memory footprint than the neural network itself (a 256×330 spectrogram requires the storage of 84480 numerical values). The search for an optimal trade-off between time-frequency resolution and classification performance will be left for future work.

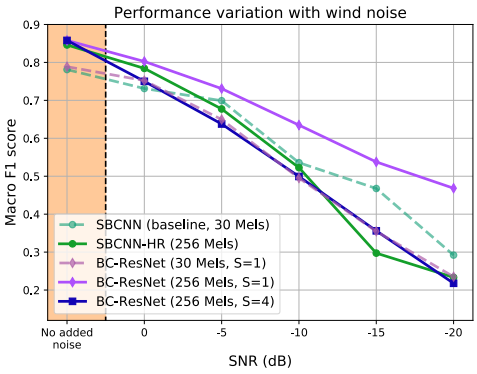
What a greater network depth does seem to have a beneficial effect on is out-of-distribution robustness: both BC-ResNet-Mod setups outperform SBCNN-HR in the OOD test set by almost 25 percentage points (Fig. 6.2b). This may indicate that the higher level of abstraction given by a deeper network provides greater generalization capabilities, despite BC-ResNet-Mod having a slightly lower parameter count than SBCNN-HR. The use of subspectral normalization over normal batch normalization seems to be vaguely useful in terms of performance on the clean test set (+2 percentage points with $S = 4$); however, SSN appears to make the CNN significantly less robust to wind noise (Fig. 6.2a).

Because of its superior passive wind resistance, we decide to experiment further with the version of BC-ResNet-Mod that uses 256 Mel bins and $S = 1$. We apply SpecMix and HMM post-processing to it in various combinations as previously done in Section 5.2.4. More specifically:

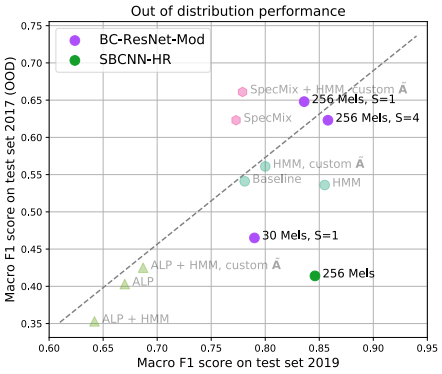
- we apply the hidden Markov model with both $\mathbf{A}$ ($p = 0.9$) and custom $\tilde{\mathbf{A}}$ on top of BC-ResNet-Mod using 256 Mel bins and $S = 1$;
- we apply the hidden Markov model with both $\mathbf{A}$ ($p = 0.9$) and custom $\tilde{\mathbf{A}}$ on top of BC-ResNet-Mod using 256 Mel bins, $S = 1$, and SpecMix during training.

It is possible to confirm the trend previously observed in Fig. 5.7: the use of the HMM with $p = 0.9$ is generally beneficial in terms of performance on the clean test set, while the custom $\tilde{\mathbf{A}}$ matrix from (5.14) provides a slight increase in robustness (Fig. 6.3b).

SpecMix appears to bring improvements in terms of overall performance in the noiseless scenario: it allows the CNN to maintain the same OOD robustness while



(a) Performance on noisy test set of BC-ResNet-Mod and SBCNN-HR. Dashed lines and solid lines indicate the usage of 30 Mel bands and 256 Mel bands in the preprocessing, respectively.



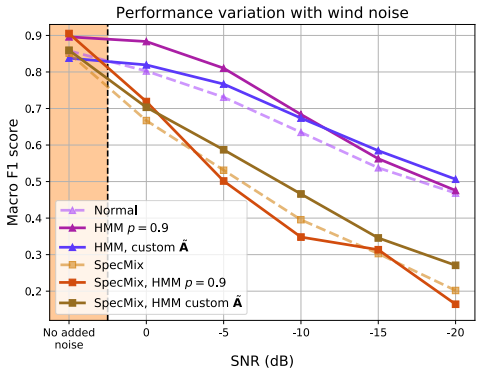
(b) Performance on out-of-distribution test set of BC-ResNet-Mod and SBCNN-HR. The performance from the old versions of the model is still visible in the background. The trend line was not re-fit to the new points to ease comparison with previous results.

Figure 6.2: Performance comparison of the new models (BC-ResNet-Mod and SBCNN-HR) over various test sets and with different parameter configurations.

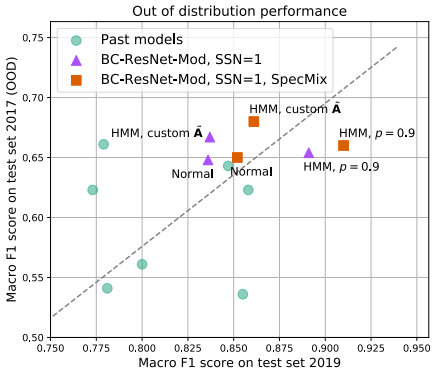
providing a performance increase in the normal test set (Fig. 6.3b). On the other hand, while generally being less effective on the out-of-distribution test set, the network version trained without SpecMix still seems more robust to wind noise (Fig. 6.3a).

Generally speaking, the modification in front-end preprocessing and network topology has increased the overall performance of the model; however, it seems that we are reaching the same trade-off situation described in Section 5.2.4. Different training setups and post-processing approaches will push the model to perform better in certain contexts while compromising on others: ultimately, the system in its current state is somewhat effective on several scenarios, and the best configuration depends on which aspect is more relevant to the final use case.

Lastly, we evaluate the SpecMix and non-SpecMix networks on the same impulse-response convolved test sets presented in Section 5.4. Interestingly enough, and conversely to the results obtained in Chapter 5, the neural network's performance seems to considerably drop when the acquisition from a different microphone is simulated: in both versions of the model, we detect an average F1-score drop of 30 percentage points across the different angles (first and second plots of Fig. 6.4). This is likely due to the fact that the artifacts introduced by the different impulse responses are more visible to the neural network in a higher frequency resolution



(a) Performance on noisy test sets of various versions of BC-ResNet-Mod. Dashed lines and solid lines indicate the usage of 30 Mel bands and 256 Mel bands in the preprocessing, respectively.



(b) Performance on out-of-distribution test set of BC-ResNet-Mod with variations of HMM and SpecMix. The performance from the old versions of the model is still visible in the background. The trend line was not re-fit to the new points to ease comparison with previous results.

Figure 6.3: Performance comparison of BC-ResNet-Mod with different combinations of HMM post-processing and training-time SpecMix augmentation.

representation; conversely, only using 30 Mels bins averages out the microphone-specific artifacts from the produced spectrogram. However, as our experiments have shown, it also has the side effect of concealing useful information for the classification task.

It now makes sense to try to fine-tune the network with audio convolved with the available impulse responses, to see if the CNN can quickly adapt to the new data distribution. This scenario can be compared to the transfer learning setup presented in Section 3.3.1, where we adapt an already trained model to a new device distribution. Here, we fine-tune using the full TAU Urban Acoustic Scenes 2019 training set convolved with impulse responses; however, because of the results obtained in Section 3.3.1, it is reasonable to believe that fine-tuning may be performed with a smaller amount of data (perhaps actually gathered from the target microphone) and still be effective.[3] We initialize BC-ResNet-Mod with the weights of the SpecMix-trained network and fine-tune it for 3 epochs keeping the same training hyperparameters as the previous experiments. We investigate two

[3]Note that this assumption would need more thorough experimental verification, as the front-end setup changed from that used in the experiments of Section 3.3.1. We leave this exploration to possible future work.

configurations: one in which only 60% of the training data is convolved with an impulse response, and one where all data is convolved. The reverberations are randomly applied in an online fashion as described in Section 5.4 by setting the probability p of performing the convolution with the input clip to 0.6 and 1.0 respectively. The results are shown in the third and fourth plots of Fig. 6.4. Using only a partial amount of convolved data will remarkably improve the results on all impulse response angles with no considerable performance loss on the original test set; however, we detect a slight degradation on the OOD test set. When using only convolved audio during fine-tuning, the network is able to fully recover its classification capabilities on all angles; however, it exhibits a noticeable performance drop in the original test set, and an even more evident one on the OOD test set.

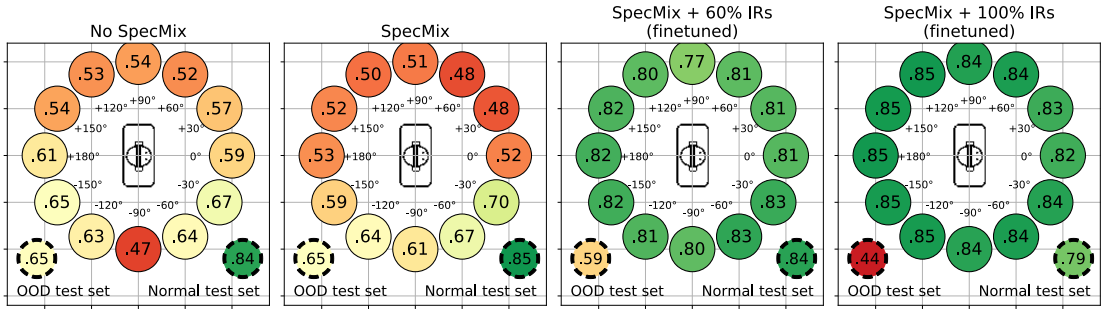


Figure 6.4: Evaluation of the new BC-ResNet-Mod setup over the test sets convolved with the impulse responses of the target microphone.

It seems that we again face a trade-off between accuracy on the original and the convolved test sets; however, this is surely not as problematic as having to balance between wind robustness and clean test set performance, as was the case in Section 5.2.4. Indeed, what we ultimately value is performance on the target microphone, which appears achievable with the proper training choices; moreover, by configuring the right proportion of convolved input audio at training time, it seems possible to achieve reasonable classification performance for both microphones.

Chapter 7

Conclusions and future work

7.1 Conclusions

In this work, we designed the grounding of a lightweight, production oriented, ML-based acoustic scene classification system. The development initially had to respect a number of constraints, notably the fixed front-end and back-end pipeline steps with a small neural network classifier and the lack of data coming from the target device. Moreover, the final classification task was not clearly outlined, and a set of suitable acoustic scenes had to be defined.

We therefore began by defining a set of scenes that was both well suited for the CNN at hand and plausible from a use-case perspective. We then assessed how much data would be needed to learn those scenes to the maximum extent of the baseline network's capabilities, and how to approach the eventuality that such an amount of data would not be available. We showed that the use of transfer learning and some straightforward augmentation strategy can be of great help in a low-data setting.

We then formalized the existence of two different pipelines: one for prototyping and training in TensorFlow, and one for the production environment running on the embedded platform. We initially established that all but one steps of the two pipelines behave the same way, and could therefore be unified. Once access to the target device's internals was possible, we investigated the issue of whether training the neural network on audio recorded from a source microphone would impact its classification performance when testing the model on data captured from a different target microphone. We concluded that, for the current front-end configuration, such a mismatch would not be problematic. This consistuted the final step in closing the gap between research and production, confirming that the prototyping environment can be representative of the performance of the model in the wild. In the process, we also uncovered some details about how the CNN

takes decisions, making a step in the direction of its interpretability. Moreover, we designed a technique to apply specific reverberations to an arbitrary audio clip in order to make it sound like it was acquired from the target microphone under certain directionality conditions.

We proceeded by testing the model on real scenarios from a physical pair of headphones; in doing so, we uncovered some of its weaknesses, such as the sensitivity to passive wind. This discovery was the starting point to the pursuit of making the model more robust to a number of external perturbations: passive wind, distribution shifts, input reverberations. We concluded that the generalization capability of the CNN can be enhanced by means of HMM-based post-processing and suitably chosen augmentation techniques; however, for a fixed front-end and back-end combination, there seems to be a fundamental limit to the generalization capability of the model, which eventually results in having to compromise between general robustness and performance on a specific task.

Thanks to a change in the hardware platform, we were subsequently able to investigate how modifications in front-end or back-end processing would impact the previously encountered trade-off situations. Specifically, we experimented with a more fine-grained input, a deeper and more complex CNN architecture, or both. According to our results, the greatest performance bottleneck of the pipeline is actually the front-end processing, and even a deeper, more refined neural network cannot improve the classification results if the feature representation of the input data is too coarse-grained. We showed that providing an input with higher frequency resolution and larger time coverage leads to a performance improvement regardless of the back-end classifier; moreover, in that situation, a deeper and more tailored neural topology can shine in terms of robustness with the proper post-processing and training time augmentations.

In conclusion: we started from a set of high-level requirements and a baseline neural network architecture. We formalized the task at hand, defining a set of acoustic scenes and obtaining proper testing and training data for them. We experimented with several pipeline modifications, including different pre-processing configurations, augmentation techniques, HMM-based post-processing, and different neural network architectures. We benchmarked those different setups against a variety of scenarios, and verified that the behavior of the pipeline was comparable in the prototyping environment and in the production platform. The outcome of this work is the foundation of a system that will hopefully be further developed and refined until it is suitable to be deployed in a real Bang & Olufsen product.

7.2 Future work

As previously mentioned, we believe the main contribution of this work was taking a set of high level requirements pertaining to the acoustic scene classification task and formalizing them within the context of a wearable audio product. We explored what problems may arise when trying to finalize a ML-based system for a consumer ready application and proposed a set of initial solutions. Ultimately, we hope that our work will serve as a framework to rigorously study and tackle each issue that emerged during our investigations. What follows is a list of suggestions, ordered by chapter, about what we believe would be meaningful research directions to concentrate future efforts on.

Chapter 2. We stated that ASC is an inherently open set problem: yet, during our analysis, we started off with a closed set of classes. This is because we chose to simplify our initial assumptions in order to get a first working version of the model. Nevertheless, a product-ready system should most likely include a mechanism that allows the recognition of unknown stimuli. This might be an alternative approach to solving the passive wind issue. Aside from the already mentioned [16], other research that deals with the open-set problem in ASC includes [67] and entries from the *Open set acoustic scene classification* sub-task from the DCASE challenge 2019, such as [127].

Chapter 3. We attempted to quantify how much data from a *target* microphone is needed when fine-tuning a model that was pre-trained on a dataset recorded by a different *source* microphone. However, this was carried out in a preliminary "proxy" setting, using the Support scenes dataset as *source*, TAU Urban Acoustic Scenes 2019 as *target*, and the SB-CNN as neural network. In order for the experiment to be more in line with the final configuration of the system, one wuold have to repeat the experiment with TAU Urban Acoustic Scenes 2019 as *source*, recordings from the *BeoMusic* as *target*, and BC-ResNet-Mod as neural network.

Chapter 4. We proposed the convolution with the impulse responses captured from the *BeoMusic*'s microphone as a means to "simulate" the availability of data recorded from the target device. While this was acceptable for a rough assessment of the impact of the device mismatch phenomenon, more realistic approaches to emulate the behavior of the model in real use cases should be sought: the problem with the impulse response stimuli is that they are highly directional. One straightforward, yet time-consuming option would be to replay the clips from EigenScape in the anechoic chamber in Ambisonics format and re-record them with the *BeoMusic*. Assuming that Ambisonics format is an acceptable approximation of real 3D audio experiences, a one-time recording of the full dataset would result in a valuable source of data to either train or test on.

Chapter 5. We mentioned that being able to tune the values of the $\mathbf{A}$ matrix in the HMM according to the amount of passive wind in the input signal would likely

be beneficial: methods of obtaining that estimate might be a valid exploration idea. There is also no guarantee that a HMM is the best post-processing option: one could experiment with completely end-to-end solutions such as using a recurrent neural network on top of the main CNN in order to perform post-processing that incorporates temporal information in a wider sense (an example of CNN + RNN setup can be found in [100]). A further, alternative approach to the problem of passive wind would be to separate the "wind detection" task from the ASC task altogether, and perform scene classification only when no wind noise is detected.[1] Concerning out-of-distribution robustness: we addressed the issue by simply training models with different generic approaches and evaluated them on a specific OOD test set to see which one would achieve the greatest robustness. However, out-of-distribution generalization is a widely studied problem in the literature, and several ad-hoc methods exist to approach it. The survey in [110] provides an overview of recent research and would be a good first step to investigate the topic.

Chapter 6. We listed a set of popular techniques in low-complexity ASC and argued why some of them were not suitable for our investigation. However, for the wider scope of the project, some of them would be reasonable techniques to adopt. Quantization would definitely be the most relevant, since it would represent an immediate reduction in memory usage. There is a possibility that teacher-student distillation methods might help in finding more general feature representations when using large general-purpose audio encoders as teacher networks: popular choices in the DCASE challenge seem to be VGG-ish [45] and L^3 [8] embeddings. However, we believe that a more relevant investigation would be which front-end parameters represent the best trade-off between performance and memory usage for the current state of the pipeline; once that has been established, a proper tuning of the c parameter of BC-ResNet-Mod would also be a useful. As previously mentioned, when changing the pre-processing parameters from those of Chapter 4, one would also need to repeat the microphone mismatch assessment to make sure that it is still valid with the new front-end configuration.

Further developments. So far, we have listed ways one could branch out from topics we laid some foundation of. However, a completely unexplored direction would be to tackle the open-set problem by learning an acoustic embedding space in an unsupervised or self-supervised way. Several experiments were carried out in that regard with promising results, likely enough to constitute a whole further chapter of this dissertation. However, for reasons related to intellectual property, we are not allowed to share the details.

[1] This would also simultaneously tackle the problem of *active* wind, i.e. the actual windy weather generating noise, and might have other uses outside the goal of acoustic scene classification.

Appendix A

Additional results on microphone mismatch assessment

A.1 Unclothed HATS

This section contains additional experiment results pertaining to the microphone mismatch assessment.

Fig. A.1 illustrates the sensitivity of the model to low frequencies: by artifically setting the lowest energy bin to a very low value, the prediction of the network can be changed from *travel* to *chatter*.

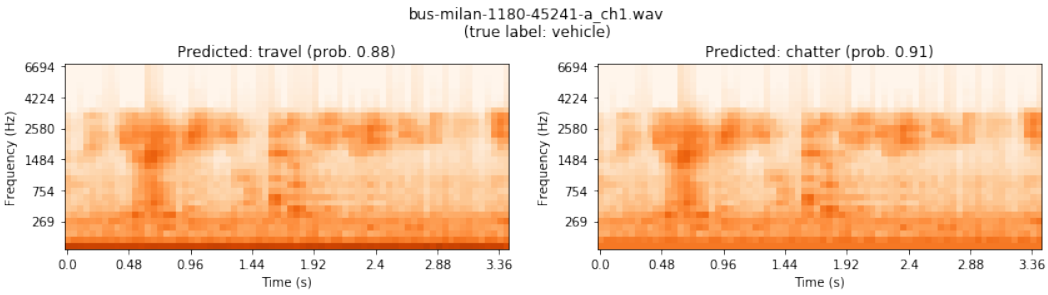


Figure A.1: Masking the first frequency bin of the Mel spectrogram to show the model's sensitivity to low frequencies.

Fig. A.2 shows the results obtained by performing the impulse response angle experiment described in Section 4.2.2 by playing the sweeps at -24 dB ("quiet" setting). The results are very similar to those obtained in the "loud" setting (-12

dB playback) and illustrated in Fig. 4.7.

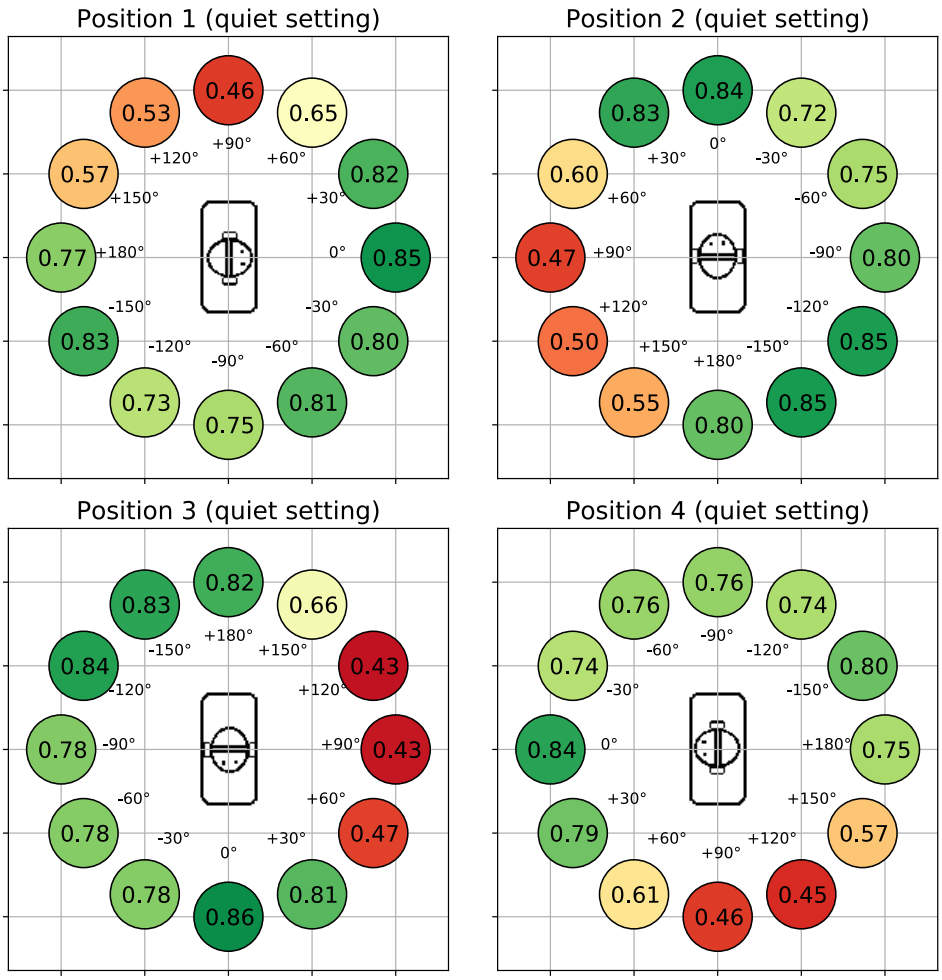


Figure A.2: Degradation in classification quality when the impulse reponse is captured at -24 dB (i.e. the "quiet" setting).

Fig. A.3 illustrates the confusion matrices of the classification obtained by evaluating the model on data convolved with impulse responses coming from several different angles. The figure highlights how +90° angles cause the CNN to fail because the classes *quiet* and *vehicle* are misclassified as *chatter*. Fig. A.5 reproduces and graphically illustrates such an issue: a boost in the frequencies between 300 and 700 Hz is sufficient to trick the model into a misclassification.

The problem can be mitigated with the use of augmentation techniques, but it is still quite noticeable at +90°: Fig. A.4 shows the confusion matrices obtained with the same test setup and a neural network trained with SpecMix as described in Section 5.2.3.

A.2 Clothed HATS

As mentioned in Section 4.2.2, a new set of impulse response recordings was taken after putting clothing on the HATS to make sure no spurious reflections would come from the mannequin's torso. Fig. A.6 shows how the mannequin was dressed. A hat was also put on to avoid reflections from the head itself. The rest of the recording setup remained unchanged from that described in Section 4.2.2.

Figure A.6: The recording setup with a (rather fashionable) clothed HATS.

Fig. A.7 shows that the issue occurs even when the HATS is clothed, so the energy boosts do not come from torso reflections. Indeed, this can also be verified by inspecting the spectra of impulse responses coming from the same angle when clothes are removed from the mannequin (Fig. A.8). Note that the gain difference between the two setups likely comes from a different gain setting somewhere in the recording pipeline, and its effect on the convolved signal is eliminated by the subsequent RMS-normalization. However, it is remarkable that there still seems to be some slight behavioral difference in high frequencies, which we know the model is not much affected by anyway. What is also evident is the energy boost in frequencies around 500 Hz for the +90° angle that causes the misclassifications.

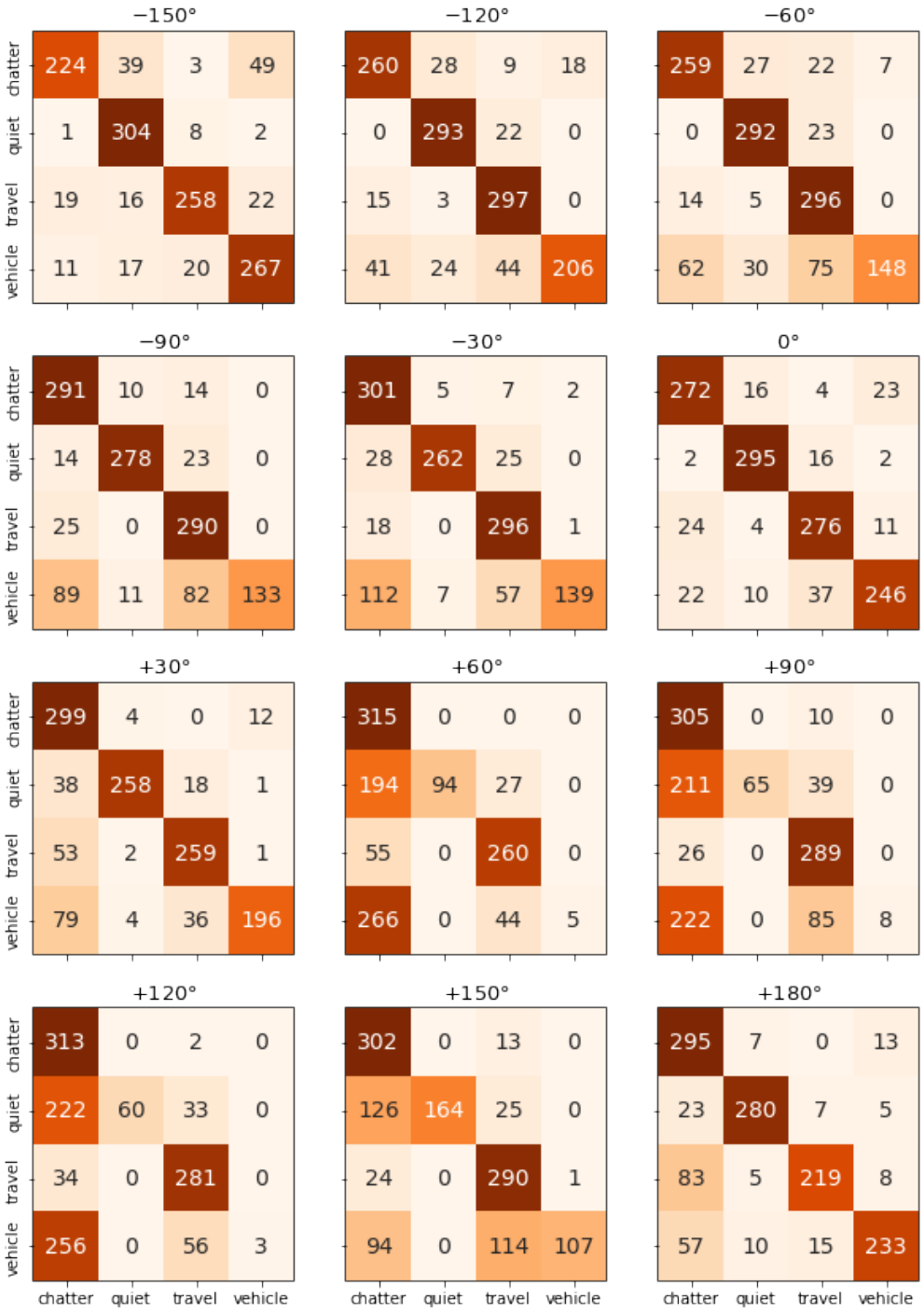


Figure A.3: Confusion matrices produced by the classification results of the SB-CNN evaluated on impulse responses from different angles.

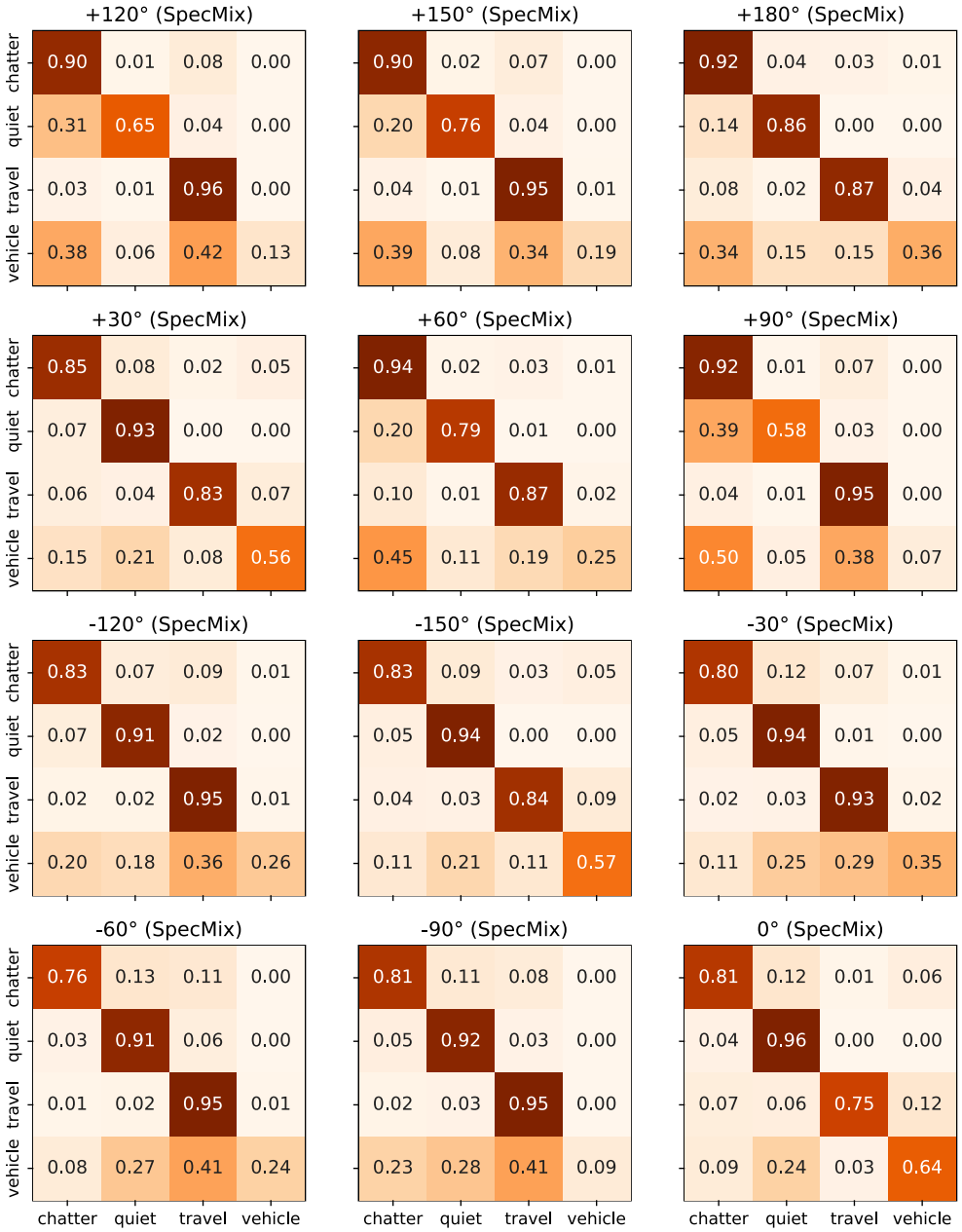
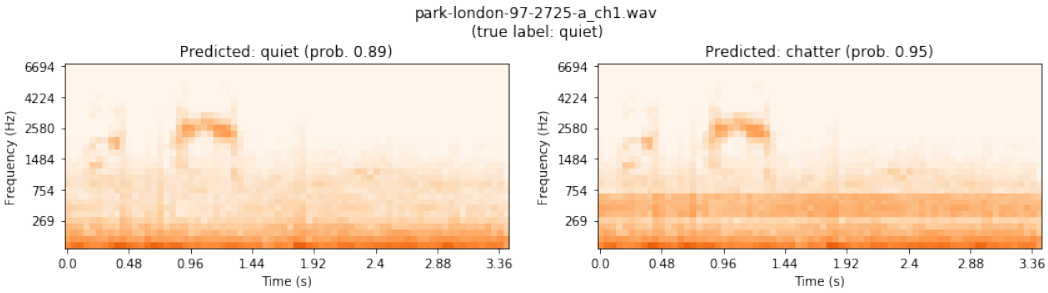
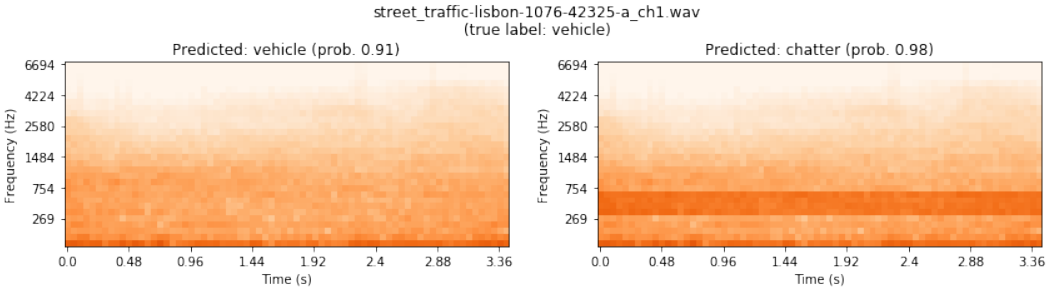


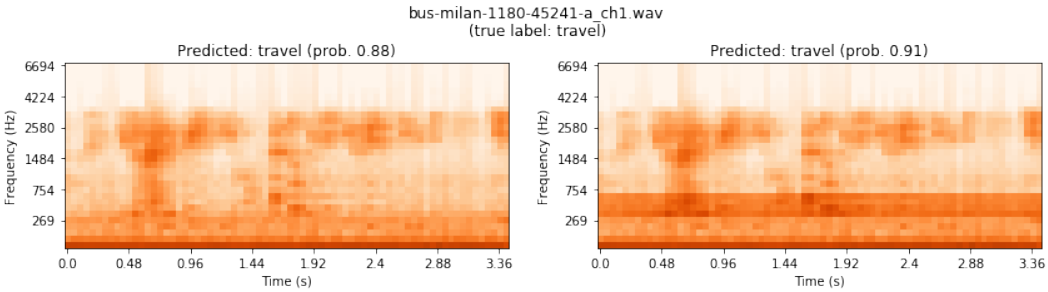
Figure A.4: Confusion matrices produced by the classification results of the SB-CNN evaluated on the same test set convolved with impulse responses from different angles as in Fig. A.3. This time, the network was trained with SpecMix augmentation.



(a) Disguising a *quiet* sample as a *chatter* sample.



(b) Disguising a *vehicle* sample as a *chatter* sample.



(c) The technique is not as effective in *travel* instances.

Figure A.5: Demonstration of how the SB-CNN can be tricked into mistaking the classes *quiet* and *travel* for *chatter* by boosting the values of the frequency bins between 300 and 700 Hz. As suggested by the confusion matrices in Fig. A.3, this problem is less evident in the *travel* class.

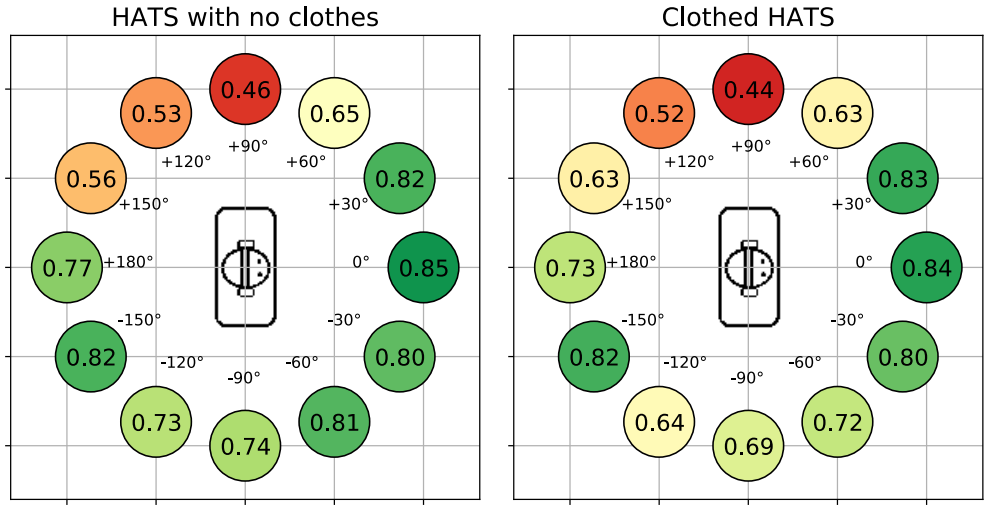


Figure A.7: Classification results comparison with unclothed and clothed HATS. No evident difference seems to emerge between the two scenarios.

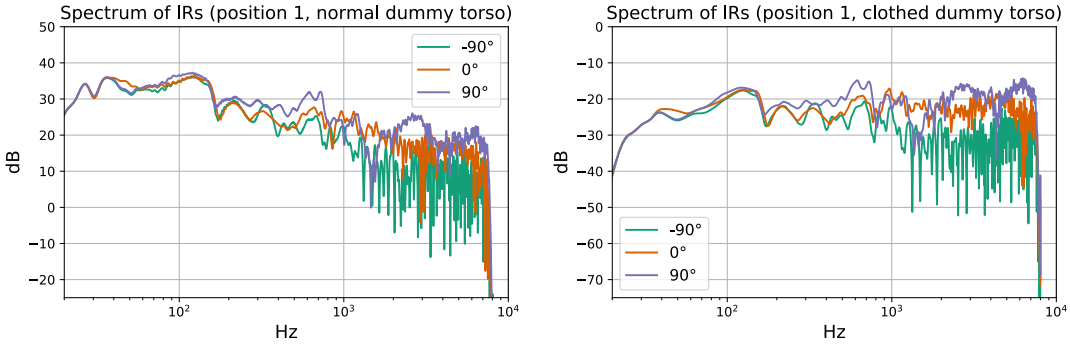


Figure A.8: Comparison of frequency responses of the *BeoMusic* microphone with unclothed and clothed HATS. The boost around 500 Hz for the +90° angle is present in both.

Appendix B

Further theoretical concepts

B.1 Viterbi algorithm

As mentioned in Section 5.2.2, the *Viterbi* algorithm represents an alternative to the *forward* approach in the usage of a hidden Markov model. Both algorithms allow to estimate the probability of the HMM being in state s_i at time t. However, while the *forward* algorithm estimates such a probability considering all possible past state transitions (see (5.3)), *Viterbi* only considers the most likely state sequence (as defined in (5.7)). The quantity $\delta_t(i)$ defined in (5.7) can be computed in a dynamic programming fashion by modifying (5.5):

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i) \mathbf{A}_{ij}\right] b_j(\mathbf{x_t}) \tag{B.1}$$

However, for each state j at time t, *Viterbi* also requires the storage of the most likely state at the previous time step:

$$\psi_{t+1}(j) = \underset{1 \leq i \leq N}{\operatorname{argmax}} \delta_t(i) \mathbf{A}_{ij} \tag{B.2}$$

At the end of the sequence (i.e. when $t = T$), it is possible to determine the most likely final state:

$$q_T^* = \underset{1 \leq i \leq N}{\operatorname{argmax}} \delta_T(i) \tag{B.3}$$

Thanks to the storage carried out in (B.2), one can infer the penultimate state of the sequence from the final state, and from that the previous one, and so on. This process is known as *backtracking*.

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right) \qquad t = T-1, T-2 \ldots 1 \tag{B.4}$$

This formulation implies that, in order to reconstruct the most likely state sequence, one has to first process the entire sequence of observations. In some settings, this

is not feasible: for example, in the ASC scenario, there is no concept of "end of sequence". To tackle this issue, one might delay the state prediction of a certain time step, process M further observations, then backtrack to find the optimal state sequence up to that point. For example, for $M = 2$, one would produce the state prediction of time t at time $t + 2$. The higher M, the more information the system can base its decision upon, the more accurate we expect the state prediction to be. However, higher values of M also correspond to higher delay. It is technically possible to set $M = 0$, in which case the state prediction rule collapses to (B.3) for all time steps.

In our problem setting, using Viterbi in the HMM postprocessing would mean introducing a delay of $M \cdot L$ seconds, where L is the duration of an audio clip used to produce a single spectrogram. For ease of implementation, the results presented in Section 5.2.4 are obtained by running Viterbi with $M = 0$.

B.2 Other augmentation techniques

In Section 5.2.3, we mentioned two augmentation techniques that, despite being popular in the ASC literature, were not deemed suitable to tackle the issue of passive wind: namely, channel swapping [79, 49] and mixup [135]. We now briefly describe both and illustrate why they were discarded.

Channel mix is a straightforward way of augmenting the data by swapping the channels of a stereo track at training time with a probability p. In our case, the technique is not applicable because we operate on mono files.

Mixup consists in creating a new training data point by linearly interpolating two input spectrograms $\mathbf{x}_i$ and $\mathbf{x}_j$, then computing a smoothed label accordingly from the one-hot encoded ground truths y_i and y_j:

$$\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1 - \lambda)\mathbf{x}_j \tag{B.5}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{B.6}$$

Where λ is randomly chosen from a uniform distribution in [0,1]. While this technique was widely employed in recent DCASE competitions, it does not really match our needs since it creates new samples by mixing the spectrograms in the entirety of their frequency range. Instead, as highlighted in Section 5.2, the disturbances generated by passive wind are mostly found in the low-frequency regions of the spectrum: therefore, a more localized augmentation would be more fitting.

According to this criterion, CutMix [132] would be a good candidate. CutMix was proposed as alternative to mixup and was originally employed with image data. It consists in swapping random patches of an image and tuning the one-hot encoded labels accordingly. It can be described by (5.8) and (5.9), the same equations as

SpecMix, the only difference being that $\mathbf{M}$ masks rectangles instead of stripes. This provides a localized perturbation of the input, which is desirable for the previously explained reasons.

SpecMix is nothing but an adaptation of CutMix from the image to the audio domain. Because it is more suitable to be applied to time-frequency representations than CutMix, we choose SpecMix as first augmentation technique.

Bibliography

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. «Convolutional Neural Networks for Speech Recognition». In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.10 (2014), pp. 1533–1545. DOI: 10.1109/TASLP.2014.2339736 (cit. on p. 5).

[2] J. Abeßer, Marco Götze, Tobias Clauss, Dominik Zapf, Christian Kühn, Hanna M. Lukashevich, Stephanie Kühnlenz, and S. Mimilakis. «Urban Noise Monitoring in the Stadtlärm Project - A Field Report». In: 2019 (cit. on p. 7).

[3] Jakob Abeßer. «A Review of Deep Learning Based Methods for Acoustic Scene Classification». In: *Applied Sciences* 10.6 (2020). ISSN: 2076-3417. DOI: 10.3390/app10062020. URL: https://www.mdpi.com/2076-3417/10/6/2020 (cit. on pp. 8, 50).

[4] Sauri Suarez Alejandro, N. Kaplanis, Stefania Serafin, and Søren Bech. «In-Virtualis: A study on the impact of concurrent Virtual Reality Environments in Perceptual Audio Evaluation of Loudspeakers». English. In: *2019 AES International Conference on Immersive and Interactive Audio*. Ed. by Tony Tew and Duncan Williams. 2019 AES International Conference on Immersive and Interactive Audio: Creating the Next Dimension of Sound Experience ; Conference date: 27-03-2019 Through 29-03-2019. Audio Engineering Society, 2019. ISBN: 978-1-942220-27-5. DOI: 10.17743/aesconf.2019.978-1-942220-27-5 (cit. on p. 29).

[5] Raziel Alvarez and Hyun-Jin Park. «End-to-end Streaming Keyword Spotting». In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6336–6340. DOI: 10.1109/ICASSP.2019.8683557 (cit. on p. 8).

[6] Audio Analytic. *The 2019 Hearables Report: AI attitudes and expectations*. 2019. URL: https://www.audioanalytic.com/hearables-report-2019/ (cit. on p. 18).

[7] M. Anusuya and S. Katti. «Front end analysis of speech recognition: A review». In: *International Journal of Speech Technology* 14 (June 2011), pp. 99–145. DOI: 10.1007/s10772-010-9088-7 (cit. on p. 12).

[8] Relja Arandjelovic and Andrew Zisserman. «Look, Listen and Learn». In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 609–617. DOI: 10.1109/ICCV.2017.73 (cit. on p. 80).

[9] Daniel Arteaga. «Introduction to Ambisonics». In: *Escola Superior Politècnica Universitat Pompeu Fabra, Barcelona, Spain* (2015) (cit. on p. 21).

[10] Haytham Assem and Declan O'Sullivan. «Towards Bridging the Gap between Machine Learning Researchers and Practitioners». In: *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*. 2015, pp. 702–708. DOI: 10.1109/SmartCity.2015.151 (cit. on p. 26).

[11] Jean-Julien Aucouturier, Boris Defreville, and François Pachet. «The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music». In: *The Journal of the Acoustical Society of America* 122.2 (2007), pp. 881–891. DOI: 10.1121/1.2750160. eprint: https://doi.org/10.1121/1.2750160. URL: https://doi.org/10.1121/1.2750160 (cit. on p. 8).

[12] Soo Hyun Bae, Inkyu Choi, and Nam Soo Kim. «Acoustic scene classification using parallel combination of LSTM and CNN». In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. 2016, pp. 11–15 (cit. on p. 8).

[13] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. «wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations». In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 12449–12460. URL: https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf (cit. on p. 8).

[14] Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. «On using specaugment for end-to-end speech translation». In: *arXiv preprint arXiv:1911.08876* (2019) (cit. on p. 23).

[15] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D. Plumbley. «Acoustic Scene Classification: Classifying environments from the sounds they produce». In: *IEEE Signal Processing Magazine* 32.3 (2015), pp. 16–34. DOI: 10.1109/MSP.2014.2326181 (cit. on pp. 7, 8, 49).

[16] Daniele Battaglino, Ludovick Lepauloux, and Nicholas Evans. «The open-set problem in acoustic scene classification». In: *2016 IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*. 2016, pp. 1–5. DOI: 10.1109/IWAENC.2016.7602939 (cit. on pp. 10, 79).

[17] Laurens Byttebier, Brecht Desplanques, Jenthe Thienpondt, Siyuan Song, Kris Demuynck, and Nilesh Madhu. *Small-Footprint Acoustic Scene Classification Through 8-Bit Quantization-Aware Training and Pruning of ResNet Models*. Tech. rep. DCASE2021 Challenge, June 2021 (cit. on pp. 13, 50, 66, 67).

[18] Simyung Chang, Hyoungwoo Park, Janghoon Cho, Hyunsin Park, Sungrack Yun, and Kyuwoong Hwang. «Subspectral Normalization for Neural Audio Data Processing». In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 850–854. DOI: 10.1109/ICASSP39728.2021.9413522 (cit. on p. 68).

[19] Hangting Chen, Zuozhen Liu, Zongming Liu, Pengyuan Zhang, and Yonghong Yan. *Integrating the Data Augmentation Scheme with Various Classifiers for Acoustic Scene Modeling*. Tech. rep. DCASE2019 Challenge, June 2019 (cit. on pp. 8, 18, 50).

[20] Selina Chu, Shrikanth Narayanan, C.-C. Jay Kuo, and Maja Mataric. «Where am I? Scene Recognition for Mobile Robots using Audio Features». In: July 2006, pp. 885–888. DOI: 10.1109/ICME.2006.262661 (cit. on pp. 7, 8).

[21] Brian Clarkson, Nitin Sawhney, and Alex Pentl. «Auditory Context Awareness via Wearable Computing». In: *Proc. 1998 Workshop Perceptual User Interfaces* (Dec. 1998) (cit. on pp. 8, 49).

[22] Corinna Cortes and Vladimir Vapnik. «Support-vector networks». In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018 (cit. on p. 8).

[23] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. «Data Augmentation for Deep Neural Network Acoustic Modeling». In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 23.9 (Sept. 2015), pp. 1469–1477. ISSN: 2329-9290. DOI: 10.1109/TASLP.2015.2438544. URL: https://doi.org/10.1109/TASLP.2015.2438544 (cit. on p. 23).

[24] Robert David et al. *TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems*. 2021. arXiv: 2010.08678 [cs.LG] (cit. on p. 26).

[25] Oscar Deniz, Noelia Vallez, and Gloria Bueno. «Adversarial Examples are a Manifestation of the Fitting-Generalization Trade-off». In: *Advances in Computational Intelligence*. Ed. by Ignacio Rojas, Gonzalo Joya, and Andreu Catala. Cham: Springer International Publishing, 2019, pp. 569–580. ISBN: 978-3-030-20521-8 (cit. on pp. 55, 56).

[26] Ha Do, Weihua Sheng, Meiqin Liu, and Senlin Zhang. «Context-aware sound event recognition for home service robots». In: Aug. 2016, pp. 739–744. DOI: 10.1109/COASE.2016.7743476 (cit. on p. 7).

[27] A.J. Eronen, V.T. Peltonen, J.T. Tuomi, A.P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi. «Audio-based context recognition». In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (2006), pp. 321–329. DOI: 10.1109/TSA.2005.854103 (cit. on pp. 8, 49).

[28] Muhammad Yusuf Faisal and Suyanto Suyanto. «Specaugment impact on automatic speaker verification system». In: *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE. 2019, pp. 305–308 (cit. on p. 23).

[29] Angelo Farina. «Advancements in Impulse Response Measurements by Sine Sweeps». In: *Journal of The Audio Engineering Society* (2007) (cit. on p. 37).

[30] Angelo Farina. «Simultaneous measurement of impulse response and distortion with a swept-sine technique». In: *108th Convention of the Audio Engineering Society*. Paris, France, 2000 (cit. on p. 35).

[31] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. «DeepGini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks». In: *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ISSTA 2020. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 177–188. ISBN: 9781450380089. DOI: 10.1145/3395363.3397357. URL: https://doi.org/10.1145/3395363.3397357 (cit. on p. 26).

[32] Darel Rex Finley. *Stereo Separation, Subwoofers, and Headphones, Demystified*. 2008. URL: http://alienryderflex.com/stereo_separation/ (visited on 11/01/2021) (cit. on p. 34).

[33] KUMAR G.SUVARNA, RAJU, K.A.PRASAD, Dr.Mohan CPVNJ, and P.Satheesh. «SPEAKER RECOGNITION USING GMM». In: *International Journal of Engineering Science and Technology* 2 (June 2010) (cit. on p. 8).

[34] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. «A survey on concept drift adaptation». English. In: *ACM Computing Surveys* 46.4 (2014). ISSN: 0360-0300. DOI: 10.1145/2523813 (cit. on p. 59).

[35] Wanying Ge, Jose Patino, Massimiliano Todisco, and Nicholas Evans. *Raw Differentiable Architecture Search for Speech Deepfake and Spoofing Detection*. 2021. arXiv: 2107.12212 [eess.AS] (cit. on p. 8).

[36] Jürgen T. Geiger, Björn Schuller, and Gerhard Rigoll. «Large-scale audio feature extraction and SVM for acoustic scene classification». In: *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2013, pp. 1–4. DOI: 10.1109/WASPAA.2013.6701857 (cit. on p. 8).

[37] Xavier Glorot and Yoshua Bengio. «Understanding the difficulty of training deep feedforward neural networks». In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html (cit. on p. 21).

[38] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. «Explaining and Harnessing Adversarial Examples». In: *International Conference on Learning Representations*. 2015. URL: http://arxiv.org/abs/1412.6572 (cit. on p. 54).

[39] Marc Ciufo Green and Damian Murphy. «EigenScape: A Database of Spatial Acoustic Scene Recordings». In: *Applied Sciences* 7.11 (2017). ISSN: 2076-3417. DOI: 10.3390/app7111204. URL: https://www.mdpi.com/2076-3417/7/11/1204 (cit. on p. 21).

[40] Karim Guirguis, Christoph Schorn, Andre Guntoro, Sherif Abdulatif, and Bin Yang. «SELD-TCN: Sound Event Localization amp; Detection via Temporal Convolutional Networks». In: *2020 28th European Signal Processing Conference (EUSIPCO)*. 2021, pp. 16–20. DOI: 10.23919/Eusipco47968.2020.9287716 (cit. on p. 50).

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep residual learning for image recognition». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on p. 68).

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification». In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123 (cit. on p. 21).

[43] Jens Hee. *Introduction to Linear Time Invariant Systems*. Oct. 2019. URL: http://jenshee.dk/signalprocessing/ltisystem.pdf (visited on 11/02/2021) (cit. on p. 35).

[44] Heo Hee-Soo, Jung Jee-weon, Shim Hye-jin, and Lee Bong-Jin. *Clova Submission for the DCASE 2021 Challenge: Acoustic Scene Classification Using Light Architectures and Device Augmentation*. Tech. rep. DCASE2021 Challenge, June 2021 (cit. on pp. 13, 50, 67).

[45] Shawn Hershey et al. «CNN architectures for large-scale audio classification». In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 131–135. DOI: 10.1109/ICASSP.2017.7952132 (cit. on p. 80).

[46] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. «Distilling the knowledge in a neural network». In: *arXiv preprint arXiv:1503.02531* 2.7 (2015) (cit. on p. 67).

[47] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV] (cit. on pp. 13, 67).

[48] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. «HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units». In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3451–3460. DOI: 10.1109/TASLP.2021.3122291 (cit. on p. 8).

[49] Hu Hu et al. *Device-Robust Acoustic Scene Classification Based on Two-Stage Categorization and Data Augmentation*. Tech. rep. DCASE2020 Challenge, June 2020 (cit. on pp. 28, 52, 89).

[50] Jonathan Huang, Paulo Lopez Meyer, Hong Lu, Hector Cordourier Maruri, and Juan Del Hoyo. *Acoustic Scene Classification Using Deep Learning-Based Ensemble Averaging*. Tech. rep. DCASE2019 Challenge, June 2019 (cit. on p. 8).

[51] M. Huzaifah. *Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks*. 2017. arXiv: 1706.07156 [cs.CV] (cit. on p. 12).

[52] Keisuke Imoto. «Introduction to Acoustic Event and Scene Analysis». In: *Acoustical Science and Technology* 39 (May 2018). DOI: 10.1250/ast.39.182 (cit. on pp. 7, 8).

[53] Sergey Ioffe and Christian Szegedy. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift». In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456 (cit. on pp. 12, 68).

[54] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. *Adversarial Logit Pairing*. 2018. arXiv: 1803.06373 [cs.LG] (cit. on p. 54).

[55] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel. «Speaker and Session Variability in GMM-Based Speaker Verification». In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.4 (2007), pp. 1448–1460. DOI: 10.1109/TASL.2007.894527 (cit. on p. 8).

[56] Byeonggeun Kim, Simyung Chang, Jinkyu Lee, and Dooyong Sung. «Broadcasted Residual Learning for Efficient Keyword Spotting». In: *Proc. Interspeech 2021*. 2021, pp. 4538–4542. DOI: 10.21437/Interspeech.2021-383 (cit. on pp. 67, 69, 70).

[57] Byeonggeun Kim, Seunghan Yang, Jangho Kim, and Simyung Chang. *QTI Submission to DCASE 2021: Residual Normalization for Device-Imbalanced Acoustic Scene Classification with Efficient Design*. Tech. rep. DCASE2021 Challenge, June 2021 (cit. on p. 28).

[58] Byeonggeun Kim, Seunghan Yang, Jangho Kim, and Simyung Chang. *QTI Submission to DCASE 2021: Residual Normalization for Device-Imbalanced Acoustic Scene Classification with Efficient Design*. Tech. rep. DCASE2021 Challenge, June 2021 (cit. on pp. 50, 66, 67, 70–72).

[59] Gwantae Kim, David K. Han, and Hanseok Ko. «SpecMix : A Mixed Sample Data Augmentation Method for Training with Time-Frequency Domain Features». In: *Proc. Interspeech 2021*. 2021, pp. 546–550. DOI: 10.21437/Interspeech.2021-103 (cit. on p. 52).

[60] Diederik P. Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980 (cit. on p. 13).

[61] Michał Kośmider. «Spectrum Correction: Acoustic Scene Classification with Mismatched Recording Devices». In: *Proc. Interspeech 2020*. Oct. 2020. DOI: 10.21437/Interspeech.2020-3088 (cit. on pp. 10, 28).

[62] Khaled Koutini, Hamid Eghbal-zadeh, Matthias Dorfer, and Gerhard Widmer. «The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification». In: *2019 27th European Signal Processing Conference (EUSIPCO)*. 2019, pp. 1–5. DOI: 10.23919/EUSIPCO.2019.8902732 (cit. on p. 70).

[63] Khaled Koutini, Hamid Eghbal-zadeh, and Gerhard Widmer. «Receptive-Field-Regularized CNN Variants for Acoustic Scene Classification». In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York University, NY, USA, Oct. 2019, pp. 124–128 (cit. on p. 70).

[64] Khaled Koutini, Schlüter Jan, and Gerhard Widmer. *Cpjku Submission to Dcase21: Cross-Device Audio Scene Classification with Wide Sparse Frequency-Damped CNNs*. Tech. rep. DCASE2021 Challenge, June 2021 (cit. on pp. 50, 66).

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on pp. 5, 8).

[66] Anders Krogh and John Hertz. «A Simple Weight Decay Can Improve Generalization». In: *Advances in Neural Information Processing Systems*. Ed. by J. Moody, S. Hanson, and R. P. Lippmann. Vol. 4. Morgan-Kaufmann, 1992. URL: https://proceedings.neurips.cc/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf (cit. on p. 6).

[67] Zuzanna Kwiatkowska, Beniamin Kalinowski, Michał Kośmider, and Krzysztof Rykaczewski. «Deep Learning Based Open Set Acoustic Scene Classification». In: *Proc. Interspeech 2020*. 2020, pp. 1216–1220. DOI: 10.21437/Interspeech.2020-3092 (cit. on p. 79).

[68] Marvin Lavechin, Marie-Philippe Gill, Ruben Bousbib, Hervé Bredin, and Leibny Paola Garcia-Perera. «End-to-End Domain-Adversarial Voice Activity Detection». In: *Proc. Interspeech 2020*. 2020, pp. 3685–3689. DOI: 10.21437/Interspeech.2020-2285 (cit. on p. 8).

[69] Triet Huynh Minh Le, Bushra Sabir, and Muhammad Ali Babar. «Automated Software Vulnerability Assessment with Concept Drift». In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 371–382. DOI: 10.1109/MSR.2019.00063 (cit. on p. 59).

[70] Yerin Lee, Soyoung Lim, and Il-Youp Kwak. *The CAU-ET Acoustic Scenery Classification System for DCASE 2020 Challenge*. Tech. rep. DCASE2020 Challenge, June 2020 (cit. on p. 52).

[71] Pei Liu, Xuemin Wang, Chao Xiang, and Weiye Meng. «A Survey of Text Data Augmentation». In: *2020 International Conference on Computer Communication and Network Security (CCNS)*. 2020, pp. 191–195. DOI: 10.1109/CCNS50731.2020.00049 (cit. on p. 23).

[72] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. «Learning under Concept Drift: A Review». In: *IEEE Transactions on Knowledge and Data Engineering* 31.12 (2019), pp. 2346–2363. DOI: 10.1109/TKDE.2018.2876857 (cit. on p. 59).

[73] Ling Ma, Dan Smith, and Ben Milner. «Environmental Noise Classification for Context-Aware Applications». In: vol. 2736. Sept. 2003, pp. 360–370. ISBN: 978-3-540-40806-2. DOI: 10.1007/978-3-540-45227-0_36 (cit. on p. 7).

[74] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. «Towards Deep Learning Models Resistant to Adversarial Attacks». In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=rJzIBfZAb (cit. on p. 55).

[75] Gustavo Mafra, Ngoc Duong, Alexey Ozerov, and Patrick Pérez. «Acoustic scene classification: An evaluation of an extremely compact feature representation». In: *Detection and Classification of Acoustic Scenes and Events 2016*. 2016 (cit. on p. 8).

[76] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (cit. on p. 26).

[77] Irene Martín-Morató, Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. *Low-complexity acoustic scene classification for multi-device audio: analysis of DCASE 2021 Challenge systems*. 2021. arXiv: 2105.13734 [eess.AS] (cit. on p. 66).

[78] Mark McDonnell. *Low-Complexity Acoustic Scene Classification Using One-Bit-Per-Weight Deep Convolutional Neural Networks*. Tech. rep. DCASE2020 Challenge, June 2020 (cit. on p. 28).

[79] Mark McDonnell. *Low-Complexity Acoustic Scene Classification Using One-Bit-Per-Weight Deep Convolutional Neural Networks*. Tech. rep. DCASE2020 Challenge, June 2020 (cit. on pp. 52, 89).

[80] Rick Merritt. *NVIDIA - What is MLOps?* 2020. URL: https://blogs.nvidia.com/blog/2020/09/03/what-is-mlops/ (visited on 10/07/2021) (cit. on p. 26).

[81] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen. «DCASE 2017 Challenge Setup: Tasks, Datasets and Baseline System». In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*. Nov. 2017, pp. 85–92 (cit. on p. 59).

[82] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. «A multi-device dataset for urban acoustic scene classification». In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*. Nov. 2018, pp. 9–13. URL: https://arxiv.org/abs/1807.09840 (cit. on pp. 10, 16, 28).

[83] Tom M. Mitchell. «Machine Learning». In: New York: McGraw-Hill, 1997, p. XV. ISBN: 978-0-07-042807-2 (cit. on p. 2).

[84] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. 2nd edition. The MIT Press, 2018. ISBN: 0262039400 (cit. on p. 56).

[85] Andrew Ng. *MLOps: From Model-centric to Data-centric AI*. https://www.deeplearning.ai/wp-content/uploads/2021/06/MLOps-From-Model-centric-to-Data-centric-AI.pdf. Accessed: 2022–03-21. June 2021 (cit. on p. 44).

[86] Jon Opedal Nordby. «Environmental sound classification on microcontrollers using Convolutional Neural Networks». MA thesis. Norwegian University of Life Sciences, Ås, 2019 (cit. on p. 12).

[87] Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, and Vincent Gripon. «Quantization and Deployment of Deep Neural Networks on Microcontrollers». In: *Sensors* 21.9 (Apr. 2021), p. 2984. ISSN: 1424-8220. DOI: 10.3390/s21092984. URL: http://dx.doi.org/10.3390/s21092984 (cit. on p. 26).

[88] Antonín Novak, Laurent Simon, František Kadlec, and Pierrick Lotton. «Nonlinear System Identification Using Exponential Swept-Sine Signal». In: *IEEE Transactions on Instrumentation and Measurement* 59.8 (2010), pp. 2220–2229. DOI: 10.1109/TIM.2009.2031836 (cit. on p. 37).

[89] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. «Activation functions: Comparison of trends in practice and research for deep learning». In: *arXiv preprint arXiv:1811.03378* (2018) (cit. on p. 4).

[90] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: 1609.03499 [cs.SD] (cit. on pp. 8, 50).

[91] Daniel Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Cubuk, and Quoc Le. «SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition». In: Sept. 2019, pp. 2613–2617. DOI: 10.21437/Interspeech.2019-2680 (cit. on p. 23).

[92] Daniel S Park, Yu Zhang, Chung-Cheng Chiu, Youzheng Chen, Bo Li, William Chan, Quoc V Le, and Yonghui Wu. «Specaugment on large scale datasets». In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6879–6883 (cit. on p. 23).

[93] Mikko Parviainen, Antti Eronen, Anssi Klapuri, and Vesa Peltonen. «Recognition of Everyday Auditory Scenes: Potentials, Latencies and Cues». In: *Audio Engineering Society Convention 110*. Audio Engineering Society. 2001 (cit. on p. 7).

[94] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 26).

[95] Anibal Pedraza, Oscar Deniz, and Gloria Bueno. «On the Relationship between Generalization and Robustness to Adversarial Examples». In: *Symmetry* 13 (May 2021), p. 817. DOI: 10.3390/sym13050817 (cit. on p. 56).

[96] Vesa Peltonen, Juha Tuomi, Anssi Klapuri, Jyri Huopaniemi, and Timo Sorsa. «Computational auditory scene recognition». In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2. 2002, pp. II-1941-II–1944. DOI: 10.1109/ICASSP.2002.5745009 (cit. on p. 8).

[97] L.R. Rabiner. «A tutorial on hidden Markov models and selected applications in speech recognition». In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286. DOI: 10.1109/5.18626 (cit. on pp. 49, 51).

[98] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions*. 2017. arXiv: 1710.05941 [cs.NE] (cit. on p. 69).

[99] Mirco Ravanelli and Yoshua Bengio. «Speaker Recognition from Raw Waveform with SincNet». In: *2018 IEEE Spoken Language Technology Workshop (SLT)*. 2018, pp. 1021–1028. DOI: 10.1109/SLT.2018.8639585 (cit. on p. 8).

[100] Mirco Ravanelli, Jianyuan Zhong, Santiago Pascual, Pawel Swietojanski, Joao Monteiro, Jan Trmal, and Yoshua Bengio. «Multi-Task Self-Supervised Learning for Robust Speech Recognition». In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2020, pp. 6989–6993. DOI: 10.1109/ICASSP40776.2020.9053569 (cit. on pp. 8, 64, 80).

[101] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. «Testing Machine Learning based Systems: A Systematic Mapping». In: *Empirical Software Engineering* (Nov. 2020). DOI: 10.1007/s10664-020-09881-0 (cit. on p. 26).

[102] Alexander Robey, Hamed Hassani, and George J. Pappas. *Model-Based Robust Deep Learning: Generalizing to Natural, Out-of-Distribution Data*. 2020. arXiv: 2005.10247 [cs.LG] (cit. on p. 54).

[103] Antoni Torras Rosell. «Methods of measuring impulse responses in architectural acoustics». MA thesis. Technical University of Denmark, Oct. 2009 (cit. on p. 35).

[104] Olga Russakovsky et al. «ImageNet Large Scale Visual Recognition Challenge». In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (cit. on p. 7).

[105] Stuart Russell and Peter Norvig. «Artificial Intelligence: A Modern Approach». In: 3rd ed. Prentice Hall, 2010. Chap. 18 (cit. on p. 3).

[106] Yuma Sakashita and Masaki Aono. *Acoustic Scene Classification by Ensemble of Spectrograms Based on Adaptive Temporal Divisions*. Tech. rep. DCASE2018 Challenge, Sept. 2018 (cit. on p. 8).

[107] Justin Salamon and Juan Pablo Bello. «Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification». In: *IEEE Signal Processing Letters* 24.3 (2017), pp. 279–283. DOI: 10.1109/LSP.2017.2657381 (cit. on pp. 12, 13).

[108] Nitin Sawhney and Pattie Maes. *Situational Awareness from Environmental Sounds*. Tech. rep. Massachusetts Institute of Technology, Dec. 1998 (cit. on pp. 7, 8).

[109] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. ISBN: 978-1-10-705713-5 (cit. on p. 55).

[110] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. «Towards out-of-distribution generalization: A survey». In: *arXiv preprint arXiv:2108.13624* (2021) (cit. on pp. 59, 80).

[111] Connor Shorten and Taghi M. Khoshgoftaar. «A survey on Image Data Augmentation for Deep Learning». In: *Journal of Big Data* 6.1 (June 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0 (cit. on p. 23).

[112] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. «Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps». In: *Workshop at International Conference on Learning Representations*. 2014 (cit. on p. 33).

[113] Karen Simonyan and Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1409.1556 (cit. on pp. 8, 21).

[114] Arshdeep Singh, Padmanabhan Rajan, and Arnav Bhavsar. «Deep Multi-view Features from Raw Audio for Acoustic Scene Classification». In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York University, NY, USA, Oct. 2019, pp. 229–233 (cit. on p. 8).

[115] Peter Sørensen, Bastian Epp, and Tobias May. «A depthwise separable convolutional neural network for keyword spotting on an embedded system». In: *EURASIP Journal on Audio, Speech, and Music Processing* 2020 (June 2020). DOI: 10.1186/s13636-020-00176-2 (cit. on p. 13).

[116] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on pp. 12, 69).

[117] Guy-Bart Stan, Jean Jacques Embrechts, and Dominique Archambeau. «Comparison of different impulse response measurement techniques». In: *Journal of the Audio Engineering Society* 50 (Apr. 2002), pp. 249–262 (cit. on p. 37).

[118] Sangwon Suh, Sooyoung Park, Youngho Jeong, and Taejin Lee. *Designing Acoustic Scene Classification Models with CNN Variants*. Tech. rep. DCASE2020 Challenge, Sept. 2020 (cit. on p. 8).

[119] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. «On the importance of initialization and momentum in deep learning». In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, Dec. 2013, pp. 1139–1147. URL: https://proceedings.mlr.press/v28/sutskever13.html (cit. on p. 21).

[120] Gen Takahashi, Takeshi Yamada, Nobutaka Ono, and Shoji Makino. «Performance evaluation of acoustic scene classification using DNN-GMM and frame-concatenated acoustic features». In: *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE. 2017, pp. 1739–1743 (cit. on p. 8).

[121] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. «Measuring Robustness to Natural Distribution Shifts in Image Classification». In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18583–18599. URL: https://proceedings.neurips.cc/paper/2020/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf (cit. on p. 61).

[122] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. «Robustness May Be at Odds with Accuracy». In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=SyxAb30cY7 (cit. on p. 55).

[123] Shuai Wang, Johan Rohdin, Oldřich Plchot, Lukáš Burget, Kai Yu, and Jan Černocky. «Investigation of specaugment for deep speaker embedding learning». In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7139–7143 (cit. on p. 23).

[124] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. «Characterizing concept drift». In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 964–994 (cit. on p. 59).

[125] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. «A survey of transfer learning». In: *Journal of Big Data* 3.1 (May 2016), p. 9. ISSN: 2196-1115. DOI: 10.1186/s40537-016-0043-6. URL: https://doi.org/10.1186/s40537-016-0043-6 (cit. on p. 21).

[126] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. «Time Series Data Augmentation for Deep Learning: A Survey». In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Survey Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4653–4660. DOI: 10.24963/ijcai.2021/631. URL: https://doi.org/10.24963/ijcai.2021/631 (cit. on p. 23).

[127] Kevin Wilkinghoff and Frank Kurth. *Open-Set Acoustic Scene Classification with Deep Convolutional Autoencoders*. Tech. rep. DCASE2019 Challenge, June 2019 (cit. on p. 79).

[128] Tingting Wu, Yunwei Dong, Zhiwei Dong, Aziz Singa, Xiong Chen, and Yu Zhang. «Testing Artificial Intelligence System Towards Safety and Robustness: State of the Art». In: IAENG International Journal of Computer Science. 2020 (cit. on p. 26).

[129] Yuzhong Wu and Tan Lee. *Searching for Efficient Network Architectures for Acoustic Scene Classification*. Tech. rep. DCASE2020 Challenge, June 2020 (cit. on p. 52).

[130] Chao-Han Huck Yang et al. *A Lottery Ticket Hypothesis Framework for Low-Complexity Device-Robust Neural Acoustic Scene Classification*. Tech. rep. DCASE2021 Challenge, June 2021 (cit. on pp. 50, 66, 67).

[131] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. «Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism». In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ISCA '17. Toronto, ON, Canada: Association for Computing Machinery, 2017, pp. 548–560. ISBN: 9781450348928. DOI: 10.1145/3079856.3080215. URL: https://doi.org/10.1145/3079856.3080215 (cit. on p. 66).

[132] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. «CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019 (cit. on pp. 52, 89).

[133] Jan Zenisek, Florian Holzinger, and Michael Affenzeller. «Machine learning based concept drift detection for predictive maintenance». In: *Computers & Industrial Engineering* 137 (2019), p. 106031. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106031. URL: https://www.sciencedirect.com/science/article/pii/S0360835219304905 (cit. on p. 59).

[134] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. «Theoretically Principled Trade-off between Robustness and Accuracy». In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 7472–7482. URL: https://proceedings.mlr.press/v97/zhang19p.html (cit. on p. 55).

[135] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. *mixup: Beyond Empirical Risk Minimization*. 2018. arXiv: 1710.09412 [cs.LG] (cit. on p. 89).

[136] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. «An overview of concept drift applications». In: *Big data analysis: new algorithms for a new society* (2016), pp. 91–114 (cit. on p. 59).