Queen Mary University of London
School of Electronic Engineering and Computer Science

Automatic detection and classification of bird sounds in low-resource wildlife audio datasets

Gnostothea-Veroniki (Veronica) Morfi

Submitted in partial fulfilment of the requirements
of the Degree of Doctor of Philosophy

2019

Statement of Originality

I, Gnostothea-Veroniki Morfi, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third partys copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature:

Date: 12/03/2019

Details of collaboration and publications:

Deductive refinement of species labelling in weakly labelled birdsong recordings. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Morfi and Stowell (2017).

Data-efficient weakly supervised learning for low-resource audio event detection using deep learning. In Proceedings of the Detection and Classification of Acoustic Scenes and Events (DCASE) 2018 Workshop Morfi and Stowell (2018a).

Deep learning for audio event detection and tagging on low-resource datasets. In Applied Sciences. Morfi and Stowell (2018b).

NIPS4Bplus: a richly annotated birdsong audio dataset. Under review in PeerJ CS. Morfi et al. (2018).

Abstract

There are many potential applications of automatic species detection and classification of birds from their sounds (e.g. ecological research, biodiversity monitoring, archival). However, acquiring adequately labelled large-scale and longitudinal data remains a major challenge, especially for species-rich remote areas as well as taxa that require expert input for identification. So far, monitoring of avian populations has been performed via manual surveying, sometimes even including the help of volunteers due to the challenging scales of the data. In recent decades, there is an increasing amount of ecological audio datasets that have tags assigned to them to indicate the presence or not of a specific bird species. However, automated species vocalisation detection and identification is a challenging task. There is a high diversity of animal vocalisations, both in the types of the basic syllables and in the way they are combined. Also, there is noise present in most habitats, and many bird communities contain multiple bird species that can potentially have overlapping vocalisations.

In recent years, machine learning has experienced a strong growth, due to increased dataset sizes and computational power, and to advances in deep learning methods that can learn to make predictions in extremely nonlinear problem settings. However, in training a deep learning system to perform automatic detection and audio tagging of wildlife bird sound scenes, two problems often arise. Firstly, even with the increased amount of audio datasets, most publicly available datasets are weakly labelled, having only a list of events present in each recording without any temporal information for training. Secondly, in practice it is difficult to collect enough samples for most classes of interest. These problems are partic-

ularly pressing for wildlife audio but also occur in many other scenarios.

In this thesis, we investigate and propose methods to perform audio event detection and classification on wildlife bird sound scenes and other low-resource audio datasets, such as methods based on image processing and deep learning. We extend deep learning methods for weakly labelled data in a multi-instance learning and multi task learning setting. We evaluate these methods for simultaneously detecting and classifying large numbers of sound types in audio recorded in the wild and other low-resource audio datasets.

Acknowledgements

As Kavafis wrote in his poem 'Ithaka':

As you set out for Ithaka
hope the voyage is a long one,
full of adventure, full of discovery.

The journey is always more important than the destination. And it's most enjoyable when you have people around you to share it with. And what a crowd of amazing people I had.

First, the **C4DM** crew, who are brilliant, supporting people, there was never a boring day spent in their presence. And most importantly out of them, I'd like to say a huge thank you to my supervisor **Dan Stowell**, that also doubled as my therapist. I know sometimes my extreme optimism may make me seem like I got everything under control but whenever I was in doubt he was always there to put me back on track. Another special shout-out to my lunch group: ladies thank you for talking about anything and everything else except for work.

Next up, I'd like to say a special thank you to my dance crew and family away from home: **DGC**. I was never one for extra-curriculum hobbies but during my PhD I realised how important they are to keep you sane. Also, having to dance K-pop in the middle of a crowded London square or street is one thing I never thought I'd be doing, yet it's the one thing that makes me feel more alive than ever.

I'd also like to thank my other extra-curriculum hobby group, my **D&D** family. It's truly the best thing anyone can do when staying at

home. Thank you for allowing me to save the world a couple of times and letting my imagination rage.

Finally, it wouldn't be a proper PhD acknowledgements if I didn't thank **my family**. My dad once told me that studying was my job and I shouldn't worry about anything else. I was 8 years old and I think I took it more literally than he intended to. So... I made studying my job. And what a great job it is; full of adventure, full of discovery. Here's to many more research years to come.

Licence

'Life is problems. Living is solving problems.'

Raymond E. Feist

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The increasing human impact on the Earth's ecosystems has led to the massive eradication and fragmentation of natural habitats (Vitousek et al., 1997). This change, along with the evolution of the climate system, has accelerated the extinction of several species (Chapin et al., 2000) and caused the endangerment of many ecological processes (Fearn et al., 2008, 2010).

The complexity and increasing fragility of the interactions between human and nature require new types of investigation if we are to be able to face the challenge of environmental surprises and take into account legacy effects (Liu et al., 2007). One recent approach to deal with these challenges is acoustic ecology (Pijanowski et al., 2011). This area of research focuses on studying of the soundscape, which is the acoustic

footprint of a landscape including its plants and animals, and may well be a source of a vast amount of information that could be used efficiently in monitoring schemes.

The application of soundscape analysis could enable us to efficiently investigate the dynamics of animal behavior, particularly when habitats are modified, fragmented, or destroyed. Birds are good bioindicators of such changes and bird populations have long been regarded as a good indicator of the broad state of wildlife, due to the fact that they occupy a wide range of habitats and respond to environmental pressures that also operate on other groups of wildlife. Many studies have indeed focused on the monitoring of bird species' richness and distribution in an attempt to highlight differences in environmental health (Andrén, 1994; MacArthur et al., 1962). Because they are a well-studied taxonomic group, drivers of change for birds are better understood than for other species groups, which enables better interpretation of any observed changes (Bardeli et al., 2010).

However, acquiring adequately labelled large-scale and long-term bird soundscape data remains a major challenge, especially for species-rich remote areas as well as taxa that require expert input for identification (Ferraz et al., 2008). So far, monitoring of avian populations has been performed via manual surveying, often even including the help of expert volunteers due to the challenging scales of the data (Johnston et al., 2014; Kamp et al., 2016). In recent decades, some ecological audio datasets have been published that have tags assigned to them to indicate the presence or absence of specific bird species. Despite these

datasets, that could be used to train an automatic system, automated detection and classification of vocalisations remains a challenging task. There is a high diversity of animal vocalisations, both in the types of the basic syllables and in the way they are combined (Scott Brandes, 2008; Kroodsma, 2005). Also, there is noise present in most habitats, and many bird communities contain multiple bird species that can potentially have overlapping vocalisations (Luther, 2008; Luther and Wiley, 2009; Pacifici et al., 2008). All these limitations (i.e. lack of data, complex and overlapping vocalisations, habitat noise) make automatic monitoring of avian populations challenging. In the present work we address these challenges, and in particular we develop approaches that can be used whenever there is a lack of soundscape data by applying machine learning techniques to automatically detect bird vocalisations in wildlife recordings and classify them to an originating species.

1.2 Aim

The aim of this work is to develop machine learning methods for achieving automatic wildlife monitoring of songbirds by using soundscape audio recordings. Due to the nature of songbird soundscape data these methods should to be able to adjust to the low quantity of recordings, that may contain complex or overlapping vocalisations from multiple species, as well as the low quality of metadata such as annotations. These methods should also be able to generalise to other types of audio with similar characteristics.

1.3 Thesis Structure

Chapter 2 introduces the main bodies of existing research which we will build upon. It begins by considering the previously proposed machine learning approaches for audio event detection and audio tagging for audio, speech and music, and then surveys relevant research on songbirds vocalisations. The chapter concludes by reflecting on this existing work to consider a strategy for achieving the research aim.

Chapter 3 focuses on machine learning approaches used in image processing for image denoising and template matching in order to detect and classify, respectively, songbird vocalisations from their spectrogram. Finally, the chapter highlights the limitations of these methods.

Chapter 4 introduces NIPS4Bplus, the first ecological audio dataset that contains bird species tags and temporal annotations, and can be used for training supervised automated methods that perform bird vocalisation detection and classification and can also be used for evaluating methods that use only audio tags or no annotations for training. This chapter describes the process of collecting and selecting the recordings comprising the dataset, and then presents our approach of acquiring the tags and temporal annotations. Finally, it provides statistical information about the labels and recordings comprising the dataset.

Chapter 5 investigates deep learning approaches that can be used for audio event detection and classification and how factorising the task into multiple less complex tasks can achieve a better performance. Different settings of deep learning, such as multi instance learning (MIL) and multi-task learning (MTL) are combined and evaluated.

Chapter 6 concludes the thesis, drawing comparisons and contrasts between the proposed methods and their predecessors, and considering the prospects for further research.

1.4 Contributions

The principal contributions of this thesis are:

- Chapter 3: a two step process, of detection/segmentation and classification, that can be applied to recordings with only audio tags present in them in order to refine a list of possible labels for each audio event. This process consists of a novel detection of areas of interest, acquiring the segments in question and a novel classification approach via deductive label refinement using a template matching algorithm.

- Chapter 4: the first ecological audio dataset that contains bird species tags and temporal annotations.

- Chapter 5: a novel loss function for deep learning in a multi-instance learning (MIL) setting for audio event detection that takes

into consideration every instance prediction in order to be computed and can pick up harder-to-detect audio events, outperforming the state-of-the-art MIL loss functions.

- Chapter 5: an approach to decompose a complex sound scene transcription task into tractable sub-tasks which are then feasible to train from limited data.

- Chapter 5: a novel multi-task learning (MTL) deep neural network architecture that maintains the advantages of joint training and also incorporates any advantages of independently training for each task.

1.5 Publications

Portions of the work detailed in this thesis have been presented in national and international scholarly publications, as follows (journal publications highlighted in bold):

- Chapter 3: most of the work described in this chapter was presented in the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Morfi and Stowell, 2017).

- Chapter 4: the detailed description of the NIPS4Bplus dataset is currently under review in **PeerJ Computer Science** (Morfi et al., 2018).

- Chapter 5: Section 5.3 on the new multi instance learning loss function was presented in the 2018 Detection and Classification of Acoustic Scenes and Events (DCASE) workshop (Morfi and Stowell, 2018a).

- Chapter 5: The overall task factorisation work described in this chapter was published in the **Applied Sciences Special Issue on Computational Acoustic Scene Analysis** (Morfi and Stowell, 2018b).

Furthermore, an application of an adaptive Fourier transform for individual bird indentification was accepted and presented in the International Speech Communication Association Conference (Interspeech) (Stowell et al., 2016a). As this in not related with our main research focus, a more detailed description is not included in this thesis.

Chapter 2

Background

The potential applications of automatic species detection and classification of birds from their sounds are many: ecological research, soundscape analysis, biodiversity monitoring, archival (Dawson and Efford, 2009; Lambert and McDonald, 2014; Drake et al., 2016; Sovern et al., 2014; Marques et al., 2013). Automatic bird soundscape analysis could enable us to efficiently investigate the dynamics of behaviour in bird populations. By understanding their response to environmental pressures we could generalise those observations to other groups of wildlife. With birds being one of the most well-studied taxonomic groups, being able to detect and classify bird vocalisations automatically will allow for more progress in the domain of automatic soundscape monitoring.

In Section 1.1, we described some of the difficulties of acquiring adequately replicated large-scale and longitudinal data, especially for remote areas and species-rich taxa. Manual surveying is still one of the most

common ways of avian population monitoring, which requires extensive manual labour. For many vocal taxa such as birds (Aide et al., 2013; Furnas and Callas, 2015; Campos Cerqueira and Aide, 2016; Frommolt, 2017), insects (Fischer et al., 1997) and bats (Mac Swiney Gonzlez et al., 2008; Armitage and Ober, 2010), automated audio recording offers a powerful tool for acoustic monitoring schemes. Currently, the bottleneck is not so much automated data collection for monitoring, as there is a rapid build-up of audio databases (Ribeiro Jr et al., 2017; Wrege et al., 2017; Stowell et al., 2016b), but more the process of detecting species vocalisations over extensive recordings covering tens of thousands of hours.

In recent decades, there has been an increasing amount of ecological audio datasets that have *tags* assigned to them to indicate the presence or not of a specific bird species. Utilising these datasets and the provided tags, some research has investigated automatically determining if a bird is active in a recording (i.e. bird audio detection) (Grill and Schlüter, 2017; Pellegrini, 2017; Adavanne et al., 2017; Cakir et al., 2017; Thakur et al., 2017; Kong et al., 2017a) or which species of birds are vocalising in a recording (i.e. bird species classification) (Goëau et al., 2017; Salamon and Bello, 2017; Knight et al., 2017). However, the methods proposed for either task typically do not predict any information about the temporal location of each audio event (bird vocalisation) or the number of its occurrences in a recording.

In this thesis, we will refer to labels that provide information only about the absence/presence of an audio event in a recording as *weak* labels or *tags* of a recording. By contrast, temporal annotations that

provide information about the temporal location of an audio event will be referred to as *strong* labels. This terminology is adopted from work on multi instance learning to which we turn in Section 2.5. In the rest of this thesis we will refer to datasets that only have this type of weak labels, may contain rare events and have limited amounts of training data as *low-resource* datasets (see Section 2.6 for a detailed description).

The focus of this thesis is on birdsong detection and classification. As noted, there are many difficulties that are posed due to the nature of these tasks, with recordings from nature having vocalisations of different bird species overlapping in time, and even recordings deriving from a single forest containing up to hundreds of different bird species, while the dataset size can be very small in comparison. By nature, the classes will be quite unbalanced with some birds being active only in a couple of recordings. Furthermore, annotating ecological data with temporal annotations to train sound event detectors and classifiers is a time consuming task involving a lot of manual labour and expert annotators, due to the high diversity of animal vocalisations and the noise present in most habitats. These factors make detailed annotations laborious to gather, while on the other hand acquiring audio tags takes much less time and effort, since the annotator has to only mark the active sound event classes in a recording and not their exact boundaries. This means that many ecological datasets lack temporal annotations of bird vocalisations, which is unfortunate since they are vital to the training of automated methods that predict the temporal annotations which could potentially then solve the issue of needing a human annotator. Also, vocalisations in reference

databases (e.g. Xeno-Canto: `https://www.xeno-canto.org/`) are typically based on targeted or manually curated recordings, hence lack both biological and technical variation present in field data to be classified. Our aim is to implement a polyphonic birdsong detector and classifier -more than one species may be active at a specific time instance- that given a birdsong dataset with only weak labels for the training set, can automatically predict the temporal information of a bird vocalisation (start and end time) and classify the species vocalising.

To establish the basis upon which this thesis is developed, in the rest of this chapter we introduce the main research areas which are related to our aim. We start with a brief introduction in the field of machine learning and deep learning that provides different methods and approaches for audio tasks relevant to our research. We then introduce the task of audio tagging and describe techniques that can predict the weak labels of recordings. We continue with methods performing weak-to-strong prediction. We present these both in a general audio setting and for bird vocalisation applications. We also describe the multi instance learning setting for machine learning that is widely used for weak-to-strong prediction. We then introduce the type of data that is the focus and motivation for our research. We describe a way of training for multiple tasks in a multi task learning setting. We conclude the chapter by reflecting upon how the state of the art in these fields bears upon our choice of strategy.

2.1 Machine Learning

Machine learning (Bishop, 2006) is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning was born from pattern recognition and the idea that computers can learn from data. Machine learning algorithms build a mathematical model of sample data, known as *training data*, in order to make predictions or decisions. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to adapt. In this section, we describe the main aspects of machine learning, focusing on the methods and models we will be using for our experiments, which are also the most commonly used ones in audio event detection and classification.

2.1.1 Machine Learning Algorithms

Machine learning algorithms are classified into several broad categories according to the way the make use of their input data. Some of the main ones are: supervised learning, semi-supervised learning, weakly supervised learning, unsupervised learning.

In *supervised learning*, an algorithm builds a mathematical model of a set of data that contains paired examples of the inputs and the desired outputs. For example, if the task were determining whether a recording contained a certain sound event, the training data for a supervised learning algorithm would include recordings with and without that event

(the input), and each recording would have a label (the output) designating whether it contained the event. Most classification and regression algorithms are built in a supervised learning setting. Classification algorithms are used when the outputs are restricted to a limited set of values (classes). Regression algorithms can have any value within a range as output.

In the case of *semi-supervised learning* algorithms, some of the training examples are missing their desired output. For example, if the task were, once more, determining whether a recording contained a certain sound event, the training data could be comprised of a small annotated dataset and a larger dataset with no annotations.

Weakly supervised learning is supervision with uncertain or weakly annotated data. The first type of labels refers to cases where the given labels are not always the ground truth, while the second one refers to cases where the training data are given with only coarse-grained labels (weak labels). In this thesis we focus on the latter case, for weakly supervised learning settings with weakly annotated data.

In *unsupervised learning*, the set of available training data contains only inputs with no desired outputs. Unsupervised learning algorithms are used to find structure in the data, e.g. grouping or clustering of data points. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities.

For all the above categories, a machine learning algorithm f will have

an input x, and try to learn parameters Θ in order to produce an output o:

$$o = f_{\Theta}(x) \tag{2.1}$$

2.1.2 Feature Transformations of Input Data

In order to solve any machine learning task meaningful characteristics are extracted from the input data. These characteristics are called features and in this section we will describe two different ways of extracting features for machine learning tasks.

The first way of extracting features is called feature engineering and involves using domain knowledge of the data to create features that make machine learning algorithms work. This requires handcrafting appropriate features. For audio tasks, such as event classification or detection, some of the most commonly used manually engineered features are spectrograms (time-frequency representations of an audio file) and mel-frequency cepstral coefficients (MFCCs) (Zheng et al., 2001). Manual feature engineering can, at times, be a difficult and computationally expensive process.

On the other hand, feature learning is a set of techniques that allows a system to automatically learn the representations needed for temporal detection or classification tasks from the input data. Feature learning is often used as a pre-processing step before performing these tasks. This replaces manual feature engineering, and allows a machine to both learn

the features and use them to perform a specific task. Feature learning can be either supervised or unsupervised. Examples of supervised and unsupervised feature learning models include artificial neural networks, multilayer perceptrons, dictionary learning, independent component analysis, autoencoders, matrix factorization and various forms of clustering.

2.2 Neural Networks

Artificial neural networks are one of the main models used in machine learning. They are brain-inspired systems that try to replicate the way the brain learns. Such systems do not have any predefined task-specific rules but learn from input examples.

Neural networks consist of input and output layers, as well as hidden layers consisting of connected units (neurons) that transform the input into something that the output layer can use. Each connection between neurons is called an edge and can transmit information from one neuron to another. In common implementations, the signal at a connection between neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons at different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the hidden layers multiple times. An example of a neural network with a single hidden

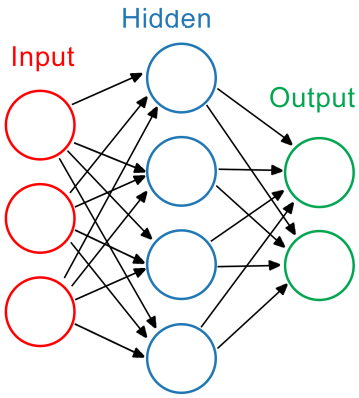


Figure 2.1: Artificial neural network that consists of an input layer, a single hidden layer and an output layer, each consisting of a different number of neurons.

layer is depicted in Figure 2.1. In this example network the outputs o are produced by applying the transformation of hidden layer f_H to the input x and finally applying the output layer f_O transformation to that:

$$o = f_O(f_H(x))) \tag{2.2}$$

Even though they have been around since the 1940s (McCulloch and Pitts, 1943; Farley and Clark, 1954; Kleene, 1956; Rochester et al., 1956), it is only in the last several decades that neural networks have become a major part of artificial intelligence due to the arrival of a technique called *backpropagation* (Werbos, 1975), which allows networks to adjust their hidden layers of neurons in situations where the outcome does not match the ground truth. Also, available computing power increased through the use of GPUs and distributed computing for training neural networks. Another important recent advance is deep learning that consists of multiple hidden layers in a single network that are used to extract different

features each to produce the final prediction.

2.2.1 Deep Learning

A deep neural network is defined as an artificial neural network with multiple layers between the input and output layers (Schmidhuber, 2015). Modern state-of-the-art deep learning is focused on training deep neural network models. Training refers to the weight optimisation process in which the error of predictions is minimized such that the network reaches a specified level of accuracy. The method mostly used to determine the adjustment of each neuron is called backpropagation (Werbos, 1975), that calculates the partial derivatives of the loss function with respect to each weight.

One of the main issues that can arise with naively trained deep neural networks is *overfitting*. Overfitting occurs when a model is too closely fit to a limited set of data points (training data) and cannot generalise to unseen data. Deep neural networks are prone to overfitting due to the added layers of abstraction, which allow them to model rare dependencies in the training data. The most commonly methods used to alleviate overfitting are dropout regularisation (Dahl et al., 2013), which randomly omits units from the hidden layers during training, Ivakhnenko's unit pruning (Ivakhnenko, 1971), weight decay (l_2-regularization), and sparsity (l_1-regularization). Finally, data can be augmented via methods such as cropping, rotating and stretching such that smaller training sets can be increased in size to reduce the chances of overfitting. Furthermore,

validation-based early stopping (Prechelt, 2012) can be used while training to avoid overfitting. This method uses a validation set, separate from the training data and the error on the validation set is used as a proxy for the generalization error in determining when overfitting has begun.

In the rest of this section, we present some of the most widely used classes of deep neural networks.

2.2.2 Multilayer Perceptron

An important general class of deep neural networks is the multilayer perceptron (MLP). An MLP model consists of one or more hidden layers and it can produce predictions by utilising nonlinear activation functions. An MLP model can be used for binary or multi-class classification and regression tasks. MLP is a type of feed-forward network where the information moves in one direction (forward) from the input to the output nodes. In feed-forward networks there are no cycles or loops. Figure 2.2 depicts such a multilayer perceptron having multiple hidden layers that follows equation 2.3 in order to predict the output predictions o. In equation 2.3 f_{H_i} denotes the hidden layer transformations with $i = 1, 2, ..., N$, f_O is the output transformation and x is the input to the MLP model.

$$o = f_O(f_{H_N}(...(f_{H_2}(f_{H_1}(x))))) \tag{2.3}$$

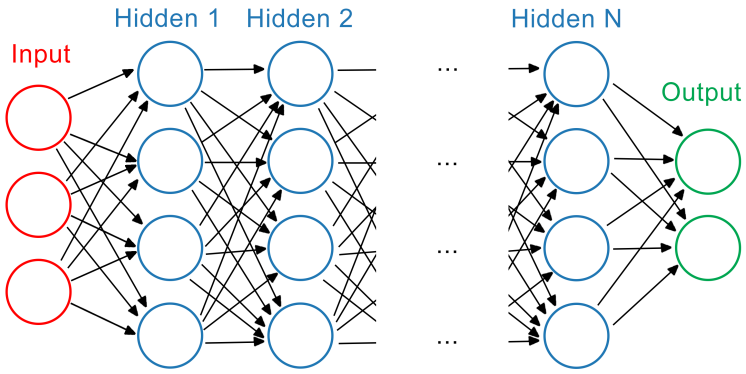


Figure 2.2: A multilayer perceptron (MLP) with multiple hidden layers.

2.2.3 Convolutional Neural Network

A convolutional neural network (CNN) (LeCun et al., 1999) is another type of feed-forward network that is related to an MLP and can be used when the input data are ordered e.g. in space (image pixels) or time (audio frames). It usually contains one or more convolutional layers and pooling followed by fully connected layers. The convolutional layer is the building block of a CNN and uses a convolution operation applied to its input passing the result to the next layer. The convolutional layer's parameters can be described as a set of learnable filters (or kernels), which have a small receptive field -the region visible to each kernel- but extend through the full depth of the input. Each filter is convolved across the width and height of the input, computing the dot product between the entries of the filter and the input in order to produce a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. The final output of the convolutional layer derives from stacking the activation maps for all filters along the depth dimen-

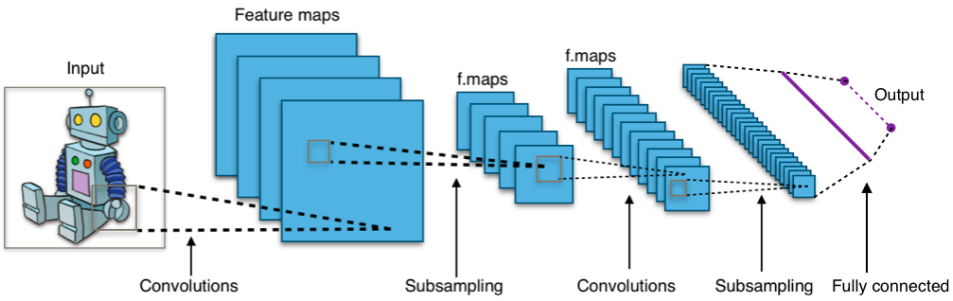


Figure 2.3: A convolutional neural network (CNN). *Image credit to Wikipedia.*

sion. Convolutional networks may also include pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. Figure 2.3 depicts such a CNN that consists of two convolutional layers each followed by a pooling layer, a fully connected hidden layer and a final fully connected layer to predict the output. The prediction of the output follows equation 2.4:

$$o = f_O(f_H(p_2(c_2(p_1(c_1(x))))) \tag{2.4}$$

where o denotes the output prediction for input x, c_i represent the convolution layers and p_i the pooling layers with $i = 1, 2$, f_H indicates the transformation of the hidden layer and f_O denotes the transformation at the final output layer.

One can notice that a convolutional neural network is quite similar to an MLP with the only difference being that the weights of a convolutional layers are constrained to be shift invariant rather than an arbitrary function, as is the case with MLP.

2.2.4 Recurrent Neural Network

A recurrent neural network (RNN) (Kolen and Kremer, 2001), unlike a feed-forward neural network, is a neural network in which some connections between neurons form a directed cycle enabling it to exhibit dynamic temporal behaviour. It can use their internal memory to process arbitrary sequences of inputs. This means that the output depends not only on the present inputs but also on the previous steps' neuron state. In basic RNN architectures, each node in a recurrent layer is connected with a directed (one-way or bi-directional) connection to every other node in the next successive time step, as depicted in Figure 2.4. In a supervised learning setting, sequences of input vectors arrive at the input nodes, one vector at a time. At any given time step, each non-input unit computes its current activation (result) as a nonlinear function of the weighted sum of the activations of all units that connect to it. The error for each individual sequence is the sum of the deviations of all targets from the corresponding activations computed by the network. The overall error for multiple sequences is the sum of the errors of all individual ones.

The output o_t of a RNN for a specific time instance t can also be described as:

$$o_t = f_O(h_t) \tag{2.5}$$

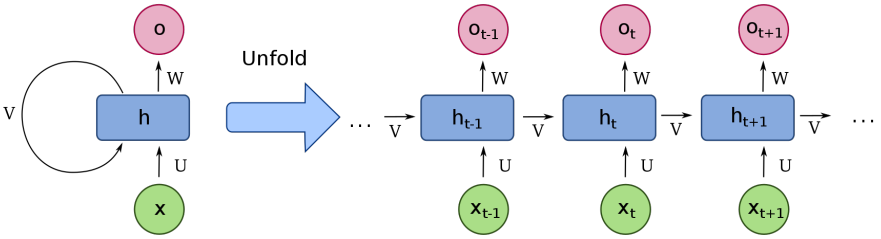


Figure 2.4: A recurrent neural network (RNN) with one-way connections and its unfolded form. (x: input, h: hidden layers, o: ouput, V: connection of previous' hidden layer's output to next one, U and W: weight matrices). *Image credit to François Deloche.*[1]

where f_O is the transformation applied at the output layer and:

$$h_t = f_H(h_{t-1}, x_t) \tag{2.6}$$

where f_H denotes the transformation of the hidden layer applied to input x_t and the output of the previous time instance h_{t-1}. The hidden layers h in a RNN describe the memory of the network and are usually referred to as states. In equation 2.6, h_t denotes the new state and h_{t-1} the old state of the network for time instance t.

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task. However due to the long-term dependencies between input and output, RNNs can suffer from the problem of the *vanishing/exploding gradient* (Bengio et al., 1994; Pascanu et al., 2013). The vanishing/exploding gradient problem manifests when training very deep neural networks. As errors propagate from layer to layer, they shrink or grow exponentially with the number of layers, im-

[1] https://commons.wikimedia.org/wiki/User:Ixnay

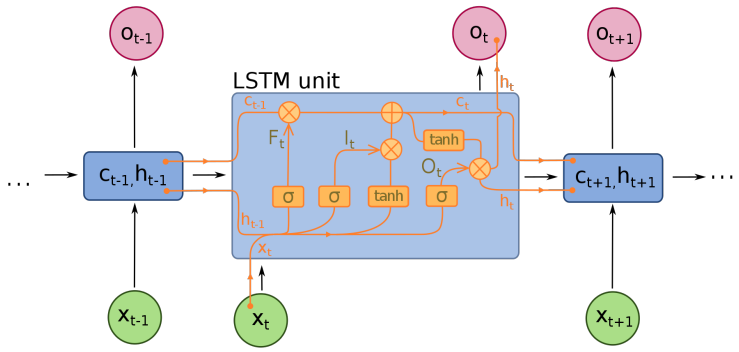


Figure 2.5: Long short-term memory (LSTM) unit. *Image credit to François Deloche.*

peding the tuning of neuron weights that is based on those errors. RNNs are trained by unfolding their directed cycles into very deep feed-forward networks, where a new layer is created for each time step of an input sequence processed by the network. Hence, they can greatly suffer from vanishing/exploding gradient.

Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a specific recurrent neural network architecture composed of LSTM units that can model temporal sequences and their long-range dependencies more accurately than conventional RNNs, avoiding the vanishing gradient problem. A common architecture of LSTM units is composed of a cell (the memory part of the LSTM unit) and three regulators, referred to as gates, of the flow of information inside the LSTM unit: an input gate, an output gate and a forget gate (Gers et al., 2000). Figure 2.5 shows the structure of an LSTM unit. I_t, O_t and F_t inside the LSTM unit denote the input, output and forget gates, respectively, of the unit.

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in Cho et al. (2014). In general GRUs have

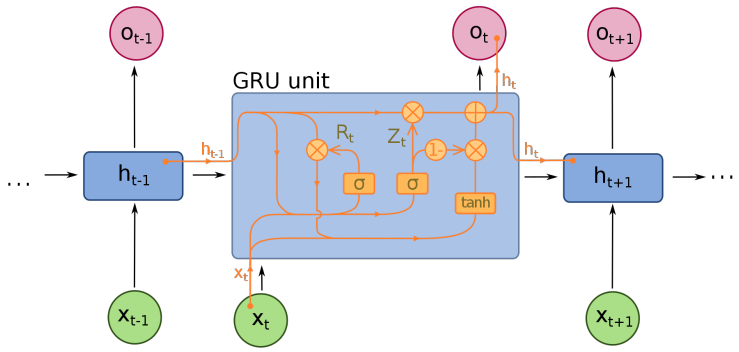


Figure 2.6: Gated recurrent unit (GRU). *Image credit to François Deloche.*

the same advantages as LSTM and perform even better when used on smaller datasets, due to the fact that they have fewer parameters since they lack an output gate as depicted in Figure 2.6, with R_t denoting a reset gate and Z_t an update gate.

All the machine learning and deep learning methods introduced have been applied for multiple tasks for audio (e.g. audio tagging, audio event detection) that relate to the research done in the scope of this thesis. We will describe the different applications in detail in the following sections to establish the basis upon which this thesis is developed.

2.3 Audio Tagging

In recent decades, there has been an increasing amount of audio datasets that have labels manually assigned to them to indicate the presence or not of a specific event type. We refer to these type of labels as *weak* labels and they lack any temporal information such as the temporal location of each event or the number of occurrences in a recording. Different

methods can utilise these labels for different tasks such as data mining. Most common methods make use of machine learning (Sections 2.1) and deep learning (Section 2.2.1).

A lot of research has been done in tagging of audio recordings with their weak labels using machine learning and more specifically deep learning methods. In Choi et al. (2016); Dieleman and Schrauwen (2014), the authors propose a music tagging algorithm using deep convolutional neural networks. In Xu et al. (2017a), the authors proposed to a deep neural network to handle the multi-label audio tagging. In Xu et al. (2017b); Adavanne et al. (2017), the authors use a stacked convolutional recurrent network, that consists of convolutional layers followed by recurrent layers, to perform environmental audio tagging and tag the presence of birdsong, respectively. While in Pons et al. (2018), the authors explore models for end-to-end music audio tagging when there is a large amount of training data.

In this thesis, we mainly focus on bird vocalisations. Early studies on audio tagging for bird vocalisations focused on small datasets in order to properly classify the species active in a recording. These datasets were usually noise-free and/or manually segmented and only contained a small number of species. Methods with fewer limitations followed (Lakshminarayanan et al., 2009; Damoulas et al., 2010; Briggs et al., 2012; Lee et al., 2008). In more recent years, even though reliable automated identification algorithms that would perform comparably to an expert observer are still non-existent (de Camargo et al., 2017), many authors have proposed methods for bird audio detection (Adavanne et al., 2017;

Pellegrini, 2017) and bird species classification, e.g. in the context of LifeCLEF classification challenges (Goëau et al., 2015, 2016, 2017) and more (Salamon and Bello, 2017; Knight et al., 2017).

However, more work is needed to address the problem of identifying all species and the exact times of their vocalisations in noisy recordings containing multiple birds. Moreover, these tasks need to be achieved with minimal manual intervention, in particular without manual segmentation of recordings into birdsong syllables, only using the weak labels of the recordings. Furthermore, a large amount of data is needed in order to train a neural network that can achieve a good quality performance. However, acquiring that kind of annotated data for bird monitoring is a nearly impossible task.

2.4 Weak-to-strong prediction

Recently, there has been an increase in demand for transcription predictions for a variety of audio recordings instead of just the tags of a recording. Some potential applications where audio event transcription is needed are context awareness for cars, mobiles, etc., surveillance for dangerous events and crimes, analysis and monitoring of biodiversity, recognition of noise sources and machine faults (Nandwana and Hasan, 2016; Stowell and Clayton, 2015; Eronen et al., 2006; Goetze et al., 2012). Depending on the audio event to be detected and classified in each task it may become difficult to collect enough samples for them. Furthermore, different tasks use task specific datasets, hence the amount of recordings

available may be limited.

In comparison to supervised techniques that are trained on strong labels, there has been relatively little work on learning to perform audio event transcription using weakly labelled data. This type of setting is sometimes referred to as *weak-to-strong prediction*. In Briggs et al. (2012); Ruiz-Muñoz et al. (2015) the authors try to exploit weak labels in birdsong detection and bird species classification. In Fanioudakis and Potamitis (2017) the authors use deep networks to tag the location of bird vocalisations with two different approaches: (a) they train multiple neural networks, one to predict an approximation of the ground truth and another one to refine the detected events and (b) extract spectral blobs -regions of interest- from a spectrogram of a recording to use for ground truth. The first approach needs a lot of training time and resources and its evaluation is ambiguous, since it uses predicted segments as ground truth, while for the second approach the blob detector cannot differentiate between bird vocalisations and other sounds. In Roger et al. (2018), the authors propose a bioacoustic segmentation based on the hierarchical Dirichlet process hidden Markov model (HDP-HMM) to infer song units in birdsong recordings, but is limited to a single species vocalising in a recording. In Schlüter (2016) singing voice is pinpointed from weakly labelled examples. In Kong et al. (2017b), the authors use a joint detection-classification network that slices the audio into blocks and an audio detector and classification on each block then uses the overall audio tag to train using the weak labels of a recording. In Adavanne and Virtanen (2017) the authors train a network that can perform automatic

scene transcription from weak labels and in Hershey et al. (2017) audio from YouTube videos is used in order to train and compare different previously proposed convolutional neural network architectures for audio event detection and classification. Finally, in Kumar and Raj (2016, 2017) the authors use weakly labelled data for audio event detection in order to move from the weak labels space to strong labels. The majority of the datasets used for all the above methods either come from transcription/detection challenges (e.g. challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)) or online sources that contain a large number of training data, such as Youtube or Xeno-Canto. [2]

Most of the above methods formulate the provided weak labels of the recordings into a *multi instance learning* problem, which is the most common formulation of weak-to-strong prediction. In the following section we describe multi instance learning as a setting for training neural networks to perform this task and its limitations.

2.5 Multi Instance Learning

The concept of multi instance learning (MIL) was first properly developed in Dietterich et al. (1997) for drug activity detection. MIL is described in terms of *bags*, with a bag being a collection of instances. The existing weak labels are attached to the bags, rather than the individual instances within them. Positive bags have at least one positive instance, an instance for which the target class is active. On the other hand, neg-

[2] `https://www.xeno-canto.org/`

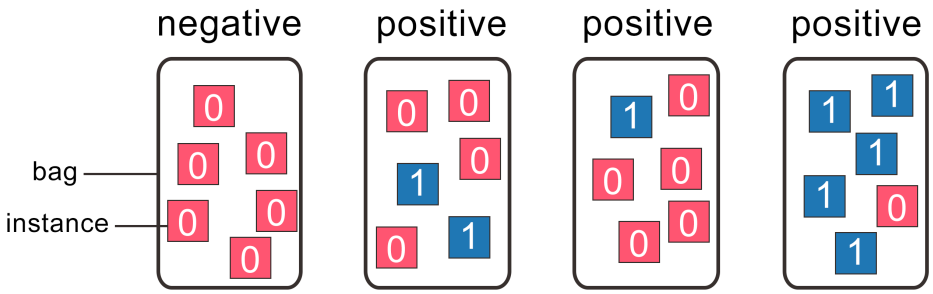


Figure 2.7: Examples of positive and negative bags of instances in a multi instance learning setting.

ative bags contain negative instances only, the target class is not active in them. A negative bag is thus pure while a positive bag is presumably impure, since the latter most likely contains both positive and negative instances. Hence, all instances in a negative bag can be uniquely assigned a negative label but for a positive bag this cannot be done. There is no direct knowledge of whether an instance in a positive bag is positive or negative. Thus, it is the bag-label pairs and not the instance-label pairs which form the training data, and from which a classifier which classifies individual instances must be learned.

Let the training data be composed of N bags, i.e. $\{B_1, B_2, ..., B_N\}$, the i-th bag is composed of M_i instances, i.e. $\{B_{i1}, B_{i2}, ..., B_{iM_i}\}$, where each instance is a p-dimensional feature vector, e.g. the j-th instance of the i-th bag is $[B_{ij1}, B_{ij2}, ..., B_{ijp}]^T$. We represent the bag-label pairs as (B_i, Y_i), where $Y_i \in \{0, 1\}$ is the bag label for bag B_i. $Y_i = 0$ denotes a negative bag and $Y_i = 1$ denotes a positive bag. Figure 2.7 presents a few example cases of positive and negative bags and their corresponding instances.

One naïve but commonly used way of inferring the individual instances' labels from the bag labels is assigning the bag label to each instance in that bag: we refer to this method as *false strong labelling*, because this makes use of a false inference. During training, a neural network in the MIL setting with false strong labels tries to minimise the average divergence between the network output for each instance and the false strong labels assigned to them, identically to an ordinary supervised learning scenario. However, it is evident that the false strong labelling approach uses a biased substitute for the loss for a strong label prediction task, hence it has some disadvantages. When using false strong labels some kind of early stopping is necessary since when perfect accuracy is achieved on the training data that would mean all positive instance predictions for a positive bag. However, there is no clear way of defining a specific point for early stopping for the MIL scenario. This is the same issue that all deep learning approaches in the MIL setting face since the error we try to minimise is all of them is only an approximation of the actual error. As mentioned before a positive bag might include both positive and negative instances, yet false strong labels will force the network towards positive predictions for both. Additionally, when using false strong labels there is an imbalance of positive and negative instance labels compared to the unknown ground truth, since a substantial amount of negative instances are labelled as positive during training leading the classifier to produce more false positives in general. Finally, a negative instance may appear in both a negative and positive recording, however due to the incorrect labelling of negative instances as positive in positive bags, the network may not learn the proper prediction for this

kind of instance.

As an alternative to false strong labels, one can attempt to infer labels of individual instances in bag B_i by defining a set of rules for each case. If $Y_i = 0$, all instances of bag B_i are negative instances, hence instance label $y_{ij} = 0, \forall j$, while on the other hand, if $Y_i = 1$, at least one instance of bag B_i is equal to one. For all instances of bag B_i, this relation between the bag label and instance labels can be simply written as:

$$Y_i = \max_j y_{ij} \tag{2.7}$$

The conventional way of training a neural network for strong labelling is providing instance specific (strong) labels for a collection of training instances. Training is performed by updating the network weights to minimize the average divergence between the network output in response to these instances and the desired output, the ground truth of the training instances. In the MIL setting using equation (2.7) to define a characteristic of the strong labels, we must modify the manner in which the divergence to be minimized is computed, as proposed in Zhou and Zhang (2002). Let o_{ij} represent the output of the network for input B_{ij}, the j-th instance in B_i, the i-th bag of training instances. We define the predicted output label for bag B_i as:

$$O_i = \max_{1 \leq j \leq M_j} (o_{ij}) \tag{2.8}$$

and the bag-level divergence as:

$$E_i = L\left(O_i, Y_i\right) \tag{2.9}$$

where L refers to any suitable loss function (e.g. Euclidean distance, binary crossentropy) and Y_i is the ground truth label assigned to bag B_i. The overall divergence on the training set is obtained by summing the divergences of all the bags in the set:

$$E = \sum_{i=1}^{N} E_i \tag{2.10}$$

Equation (2.9) indicates that if at least one instance of a positive bag is perfectly predicted as positive, or all the instances of a negative bag are perfectly predicted as negative, then the error on the concerned bag is zero. Otherwise, the error is based on the instance whose corresponding actual output value is the maximal among all the instances in the bag.

MIL scenarios are common for weak-to-strong prediction tasks due to the many publicly available datasets that lack strong annotations. For this thesis, we define an MIL scenario where the dataset not only contains weak labels but is comprised by low-resource data, as we describe in the following section.

2.6 Low-Resource Data Scenarios

The term *low-resource* was first encountered in speech studies as it refers to spoken languages that lack a great number of speakers and/or proper

documentation or dictionary. According to LORELEI, low-resource languages are languages for which no automated human language technology exists.[3] More specifically in natural language processing (computational linguistics) the term low-resource language refers to languages that for a given task, there is no algorithm using currently available data to automatically do the task with adequate performance.

In this thesis, we refer to *low-resource* datasets as datasets that, similar to low-resource languages, could have any of the following attributes:

- limited amount of data
- limited amount of reference recordings for each class
- rare classes
- weak labels (no strong labels)

In recent decades, there has been an increase in demand for transcription predictions for a variety of audio recordings instead of just the tags of a recording (see Section 2.4). Depending on the audio event to be detected and classified in each task it may become difficult to collect enough samples for them. Furthermore, different tasks use task specific datasets (e.g. with specialised lists of events types), hence the amount of recordings available may be limited. Additionally, it is easier to acquire weak labels for the recordings instead of strong labels. Hence, a lot of low-resource datasets are available for different research tasks. When used

[3]Low Resource Language for Emergent Incidents (LORELEI) is a US government funded project aiming at developing human language technology for low-resource languages. More information can be found at `https://www.darpa.mil/program/low-resource-languages-for-emergent-incidents`.

for training audio event detectors, low-resource datasets often present the issue of weak-to-strong prediction which will be one important focus of this thesis.

2.7 Multi-Task Learning

In machine learning, optimisation usually occurs over an explicit performance metric. In supervised learning the metric captures the accuracy in performing a specific labelling task. In order to do this, a single model or an ensemble of models is trained to perform the desired task and then they are fine-tuned and tweaked until their optimal performance is reached. So far acceptable performance has been achieved in ML tasks by this type of training. However, when focusing on a single task, we ignore information that might help improve on the metric in question; specifically, information from the training signals of related tasks. By sharing representations between related tasks, we can enable our model to generalize better on our original task. This approach is called Multi-Task Learning (MTL) (Caruana, 1997).

MTL aims to improve the performance of multiple learning tasks by sharing useful information among them. So far, MTL has been used successfully across a number of machine learning applications, such as natural language processing (Collobert and Weston, 2008), speech recognition (Deng et al., 2013) and computer vision (Girshick, 2015). MTL can be very useful when using low-resource datasets since it can exploit useful information from other related learning tasks to help alleviate the

issue of limited data. Based on the assumption that the multiple tasks are related and thus there is some information in the data relevant to more than one task, MTL is empirically and theoretically found to lead to better performance than independent learning. MTL is similar to transfer learning (Pan and Yang, 2010) which also transfers knowledge from one task to another by storing knowledge gained while solving one problem and applying it to a different but related problem. However, the focus of transfer learning is to help a single target task by initially training on one or multiple other tasks while MTL uses multiple tasks to help each other. Furthermore, MTL can be viewed as a generalization of multi-label learning (Zhang and Zhou, 2014) when different tasks in multi-task learning share the same training data.

In a sense, MTL performs implicit data augmentation to the sample size that we are using for training our model. When training a model on one task, we aim to learn a good representation for this task that ideally ignores the data-dependent noise and generalizes well. Different tasks should have different noise patterns and a model that learns two tasks simultaneously is able to learn a more general representation. While learning just one task bears the risk of overfitting to it, learning two tasks jointly with MTL enables the model to obtain a better representation through averaging the noise patterns. Furthermore, if there is a lack of training data, such as the low-resource data scenario, it can be difficult for a model to differentiate between relevant and irrelevant features. MTL can help the model focus its attention on those features that actually matter as other tasks will provide additional evidence for the relevance

or irrelevance of those features. Finally, MTL acts as a regularizer by introducing an inductive bias, a set of assumptions made by the model to learn multiple target functions, from training for multiple tasks. Hence, it reduces the risk of overfitting. An overview of MTL can be found in Zhang and Yang (2018).

The two most common ways to perform MTL in deep neural networks are referred to as *hard and soft parameter sharing*. Hard parameter sharing has been the most commonly used approach so far and it goes back to Caruana (1993). In hard parameter sharing some, usually the initial, layers are shared between all tasks while each task also has task-specific output layers, as depicted in Figure 2.8. Hard parameter sharing greatly reduces the risk of overfitting as shown in Baxter (1997). This is due to the fact that the more tasks we are learning simultaneously, the more our model has to find a representation that captures all of them and the less the chance of overfitting on our original task.

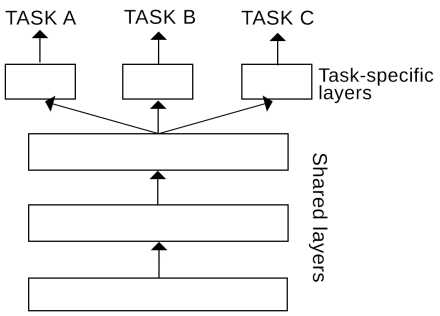


Figure 2.8: Hard parameter sharing for multi-task learning in deep neural networks.

On the other hand, in soft parameter sharing each task has a separate model and parameters, as depicted in Figure 2.9. The distance between the parameters of the constrained layers of the model is then regularised

(i.e. encouraged to take small values) in order to encourage similarity between the parameters.

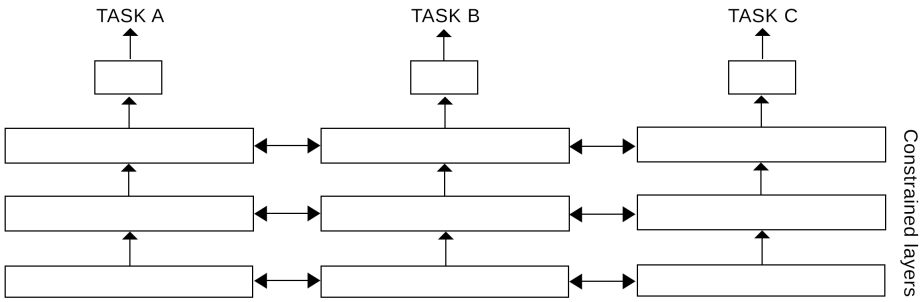


Figure 2.9: Soft parameter sharing for multi-task learning in deep neural networks.

Based on the attributes of MTL it is a good fit for both the settings of MIL and low-resource data and we will further explore it in following chapters.

2.8 Strategy

In this chapter we have set the context for our research topic, introducing the topics of audio tagging, weak-to-strong prediction, multi instance learning, low-resource data and multi-task learning. We are now in a position to reflect upon how to achieve our aim (Section 1.2) in light of this context, and devise an appropriate strategy.

In machine learning, applications that try to achieve audio transcription usually focus on the final task at hand. However training a model to predict an audio transcription using a low-resource dataset can sometimes prove to be impossible, especially when using deep neural networks

since they need a large amount of data to train properly. A network needs to have enough parameters to be able to predict all the different classes without ignoring any rare events, but also be small enough or have just the right amount of regularisation as to not overfit the limited amount of training data available. This becomes even harder when the task is a weak-to-strong prediction where the network needs to predict full transcriptions from weak labels.

In this thesis, due to the low-resource setting, we first investigate machine learning methods that perform segmentation of areas of interest in a spectrogram and then predict the class of those segments via template matching. These methods are inspired by image processing methods and tasks. However, such methods lack many of the generalisation properties that are found in deep learning.

In order to achieve greater generalisation, we then investigate deep learning approaches for our task. In a low-resource setting, there is no specific way of defining a network and type of training that ensures that a transcription task will be successful. However, a full transcription task can be defined as multiple intermediate tasks of detection and classification that might be easier to train even when using a low-resource dataset. We propose and investigate how training for intermediate tasks can be used to boost the performance of a full transcription task. For each intermediate task we propose a training setup to optimise their performance. We also compare the results of training the intermediate tasks independently and in multi-task learning setting.

Chapter 3

Automatic Segmentation-Classification of Bird Vocalisations

Our aim is to extract bird vocalisations in a fully automatic way out of soundscapes and create an algorithm that classifies them in a low-resource setting, where we only have the weak labels of the recordings. To achieve our goal, we first explore a two step process: first a segmentation algorithm that detects all vocalisations, and then a classification method that labels the retrieved segments.

We implemented and tested two different systems. Both systems performed segmentation by an event detection algorithm based on image processing of spectrograms (Section 3.1). For the first system, the classification occurs through a deductive label refinement procedure, utilizing

the weak labels and using a template matching method throughout the dataset (see Section 3.2.1). For the second system described in Section 3.2.2, a deep convolutional classifier is trained using the segments produced at the first step.

3.1 Segmentation

To segment bird vocalisations from spectrogram data, we employ the event detection paradigm used in Fodor (2013), Lasseck (2015) and Potamitis (2015). This process is used in order to detect the specific coordinates of the bird vocalisations taking place in a spectrogram, disregarding any noise. In our case, since we have the labels per recording, only recordings labelled as having at least one bird species present are used in the extraction of segments.

We combine and refine the three aforementioned paradigms, in order to create a segmentation process that will best fit our automatic transcription task. All three methods are very closely related but have some differences. For all of them there are recordings for which one will produce better results than the other two, in the sense of including segments that the other two fail to find or excluding segments produced by noise that the other two detect as vocalisations. The method proposed by Fodor was the first to appear in the context of bird recognition and opened the way for others. The method proposed by Lasseck includes a spectral enhancement stage that reduced the number of segments produced by noise and discards acoustic events that fill the whole recording

(e.g. Cicada songs, rain). Hence, in general, it produces fewer segments than the other two methods. The different stages that compose each method are showcased below.

Fodor segmentation:

1. Gaussian filter for smoothing the spectrogram
2. Local gradient applied to the spectrogram
3. Binarization that sets any value of the spectrogram that is in the highest 10% as 1 and the rest as 0
4. Binary opening & binary closing for morphological noise removal
5. Filling of holes and removal of small objects that are too small to be considered a bird call (size $\leq$ 100 pixels)

Potamitis segmentation:

1. Gaussian filter for smoothing the spectrogram
2. Enhancement of edges by adding the difference of smoothed spectrograms
3. Binary opening and binary closing
4. Filling the holes and removal of small objects (size $\leq$ 100 pixels)

Lasseck segmentation:

1. Normalisation of spectrogram to 1.0 based on the max value of the spectrogram

2. Removal of the 4 lowest and 24 highest spectrogram rows, corresponding to lowest 86 Hz and highest 516 Hz, respectively

3. Binarization via median clipping per frequency band and time frame by setting each pixel to 1 if it is above 3 times the median of its corresponding row AND above 3 times the median of its corresponding column, otherwise to 0

4. Binary closing, dilation and median filtering for further noise reduction

5. Removal of small objects (size $\leq$ 100 pixels)

For our purposes, a method that is robust to noise and does not generate noise segments is of great importance. This is due to the fact that we use the labels for each recording and assume that all segments produced derive from one or many of those labels. Hence, we implement a refinement of these segmentations based on the Lasseck segmentation, incorporating aspects of the other segmentations. First we obtain the spectrogram (time-frequency representation) of a recording via the librosa Python library (i.e. librosa.core.stft), with window size of 512, Hann window and overlap of 75%. Then, the following steps are performed for the spectrogram derived from each recording:

Our proposed segmentation:

1. Normalisation of the spectrogram values to 1.0 using its absolute max value

2. Removal of frequencies above 20 kHz and below 340 Hz. Since no bird vocalisations occurred in those frequencies, the only audio present there could be considered noise

3. Binarization via median clipping per frequency and time frame in order to eliminate any noise: we set pixel to 1 if its value was 3 times higher than the median of its corresponding row and column, otherwise it was set to 0

4. Binary closing (see (Gonzalez and Woods, 2006), pp.657-661) in order to fill any small holes in a present feature (i.e. vocalisations). Binary closing was applied in a rectangle neighbourhood of size (3,3)

5. Removal of connected components of less than 5 pixels

6. Dilation (see (Gonzalez and Woods, 2006), pp.655-657) in a rectangle neighbourhood of size (7,7). Dilation sets a pixel at (i,j) to the maximum over all pixels in the neighbourhood centred at (i,j). Dilation was applied in order to enlarge the regions that contain features (i.e. vocalisations) and remove small objects that could be considered noise

7. Median filtering of size 5

8. Removal of connected components of less than 150 pixels

9. Dilation in a circular region of radius 3

10. Defined all connected pixels as a segment (seg_i)

In our implementation, an extra step, compared to the Lasseck method, of removing small segments (step 5) is added before applying the first dilation (step 6). Since dilation enlarges regions where features are present, using dilation without first removing small objects results in expanding these regions. However, such small segments are mostly caused by noise and are not actual vocalisations. Eliminating them in this early step further reduces the noisy segments produced at the end of the segmentation process. Additionally, an extra dilation (step 9) is applied at the end of the algorithm. This second dilation has a much smaller neighbourhood (disk of radius 3) than the first one and it is used as a refined way of slightly expanding the borders of the segments detected and filling any small holes still present. This is especially helpful in larger vocalisations which are sometimes split into multiple smaller vocalisations, since this dilation can connect two vocalisations if they are close enough to each other (depending on the dilation neighbourhood). Compared to the original algorithm presented by Lasseck, this variation produces fewer noise segments and fewer, but larger, vocalisation segments.

3.1.1 Spectrogram Denoising

In order to improve segmentation, different denoising methods are tested before our proposed segmentation took place. First we test non-local means denoising (NL_Means) (Buades et al., 2011). Unlike local mean filters, which take the mean value of a group of pixels surrounding a

target pixel to smooth the image, non-local means filtering takes a mean of all pixels in the image, weighted by how similar these pixels are to the target pixel. This results in much greater post-filtering clarity, and less loss of detail in the image compared with local mean algorithms. If compared to other well-known denoising techniques, non-local means adds method noise (i.e. error in the denoising process) which looks more like white noise, which is desirable because it is typically less disturbing in the denoised product (Buades et al., 2004).

For an image Ω, with discrete pixels, the discrete NL_Means algorithm is:

$$u(p) = \frac{1}{C(p)} \sum_{q \in \Omega} v(q) f(p, q) \tag{3.1}$$

where $u(p)$ is the filtered value of the image at point p, $v(q)$ is the unfiltered value of the image at point q and $C(p)$ is given by:

$$C(p) = \sum_{q \in \Omega} f(p, q) \tag{3.2}$$

Then, for a Gaussian weighting function:

$$f(p, q) = e^{-\frac{|B(q) - B(p)|^2}{h^2}} \tag{3.3}$$

where h is the filtering parameter (i.e., standard deviation) and $B(p)$ is given by:

$$B(p) = \frac{1}{|R(p)|} \sum_{i \in R(p)} v(i) \tag{3.4}$$

where $R(p) \subseteq \Omega$ and is a square region of pixels surrounding p and $|R(p)|$

is the number of pixels in the region R.

The second method we test was Chambolle's total variation denoising method (Chambolle_TV) (Chambolle, 2004). This method minimises total variation and tends to produce piece-wise constant images. It is quite efficient for regularizing images without smoothing the boundaries of the objects. For more details see Duran et al. (2013).

3.1.2 Evaluation of Segmentation on Natural Data

In order to evaluate our proposed segmentation method and observe the effect of the two denoising methods we used a subset of the training dataset used in the Neural Information Processing Scaled for Bioacoustics (NIPS4B) bird song competition of 2013[1] (Glotin et al., 2013). The NIPS4B training dataset contains bird vocalisation recordings that have already been weakly labelled with the species present in each of them. However there are no strong labels available. Hence, for evaluation we strongly annotated a subset of the original dataset. Out of the total 87 labels present in NIPS4B, we created our subset by annotating recordings containing only 28 of the labels. This resulted in a subset of 63 recordings. A full description of the NIPS4B training dataset can be found in Chapter 4.

For the purpose of evaluating our originally proposed segmentation method and observing the effect of the denoising methods, we then ignored the bird species labels to focus on the pure detection task. Hence

[1] `http://sabiod.univ-tln.fr/nips4b/challenge1.html`

the dataset had labels denoting the presence (label=1) or absence (label=0) of a bird vocalisation in a recording. Evaluation occurred with the metrics found in DCASE 2016 for the task of sound event detection in real life audio[2]. F-measure, precision and recall are all computed based on the True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) for each time frame based on whether each segmentation method detected a bird active in the time frame or not, and the ground truth of that frame.

Precision evaluates the number of True Positives compare to the total number of positives produced by the method. The higher the precision is, the fewer False Positive results.

$$precision = \frac{TP}{TP + FP} \tag{3.5}$$

Recall, also known as sensitivity, is the fraction of True Positives over the total amount of ground truth positives. The higher recall is, the fewer ground truth positives are classified as negatives by the method.

$$recall = \frac{TP}{TP + FN} \tag{3.6}$$

Finally, F-measure is a measure of the method's accuracy. It considers both precision and recall of the method to compute the score. F-measure is the harmonic average of the precision and recall and can be used as an

[2] `http://www.cs.tut.fi/sgn/arg/dcase2016/task-sound-event-detection-in-real-life-audio`

overall metric for the methods.

$$F - measure = 2\frac{precision \cdot recall}{precision + recall} \tag{3.7}$$

Table 3.1: Evaluation of segmentation when all species labels are considered one common label (bird).

	F-measure	**precision**	**recall**
Original Method	**88%**	90%	88%
NL_Means	87%	90%	**91%**
Chambolle_TV	**88%**	**91%**	85%

In Table 3.1, the percentage results of the evaluation metrics for the three segmentation methods are depicted. By studying this table, it becomes apparent that adding a step of denoising before our originally proposed segmentation does not boost the performance of the method itself. However, the segments output from these three segmentations can vary a lot. In the following section, we describe and evaluate the classification methods proposed, and further we perform and overall evaluation of both segmentation and classification.

3.2 Classification

Using the detected segments we investigated two distinct methods for automatic classification. The first one we refer to as *deductive label refinement* and it performs segment classification by utilising a template matching algorithm on the spectrograms.The second method is based on

deep learning and more specifically convolutional neural networks that so far have proven to have a good performance for image classification. We explain both methods in detail in the following sections.

3.2.1 Deductive Label Refinement

Following segmentation, an instance based classification algorithm with no explicit training phase is implemented. In our approach, weakly labelled recordings are used. Hence, the species present are the labels of that recording ($labels\_rec$), however, we have no further information as to the specific vocalisations. For each recording, the segments that derive from the segmentation process (seg_i) are considered to be attributable to vocalisations from the bird species included in the weak labels.

For each segment, we create a list of possible labels ($labels\_seg_i$), initialized to the weak labels of the recording that contains the segment. The $labels\_seg_i$ list of a segment will later on be shortened to either one or multiple labels by the classification process via deductive elimination of the less possible labels for that segment. During classification, each segment in need of labelling is matched using normalized cross-correlation (scikit-image's *match_template* function) to different recordings in order to obtain all the possible label matches. In this process, normalized cross-correlation is used to match a template (vocalisation) to a 2D target image (spectrogram of a recording). The result is a response image of same size as the target image, with correlation coefficients between the template and target image of values between -1.0 and 1.0. The matching

value between a segment and a specific recording is found by searching for the maximum peak in the response image. Due to the number of recordings and segments detected in each of them, this process is very time consuming. However, similar bird sounds should appear in similar frequencies, hence we reduce the computational load by constraining the search to a smaller range of frequencies (within 5 frequency bins below and above the segment frequencies which for our setting correspond to approximately 200Hz above and below the segment frequencies). Furthermore, since the weak labels of a recording and a segment are already known, we only need to search for a segment match in recordings that contained at least one of the segment labels ($labels\_seg_i$).

The proposed classification has no need for a separate training set as it can classify vocalisations by finding matches within a provided set of weakly labelled recordings. The performance of the method increases as the number of recordings per species increased. The chance of the classification process finding a match for a segment increases along with the variation of each species' vocalisations. This process is implemented in three steps, namely the First Pass, Second Pass and Third Pass. All three are applied to the recordings in order, as explained in the following subsections and illustrated in Figure 3.1.

First Pass

In the First Pass of the classification, in order to best utilize the information provided by the weak labels, we create groups of recordings $recs(c_i)$ for each segment seg_i to find matches with, where c_i denotes the differ-

ent label combinations produced by the initialised labels seg_i list. The recordings in $recs(c_i)$ have label(s) c_i present in their weak labels. For each segment in need of a label the matching process searches through the list of recordings $recs(c_i)$ increasing the number of weak labels (i.e. $|c_i| = 1, 2, 3,...$) until a match is found or there are no more recordings remaining. Since *match_template* always returns a result (maximum peak in the response image), in our implementation, we consider that a match is found when the similarity rate returned by *match_template* was 0.4 or greater. The 0.4 threshold was obtained after preliminary experimentation. All the different values of the matches found in these recordings for each possible label combination c_i are summed and the label(s) with the highest sum(C_i) is(are) assigned as the segment's label(s). If no match is found in $recs(c_i)$ the Match Not Found (MNF) label is assigned to seg_i. Segments with the MNF label and segments that have more than one possible labels in $labels\_seg_i$ are classified as Unknown in our evaluation results (Section 3.2.1), even if the correct segment label is between the multiple possible labels. **Algorithm 1** describes this classification procedure.

Second Pass

The Second Pass of the process derives from the need to solve the issue of unclassified segments, MNF segments, produced during the First Pass of the classification. Since we use only weakly labelled datasets, all the labels of a recording must be assigned to at least one segment. A trivial solution for reducing the MNF segments is: when there are MNF seg-

Algorithm 1: Classification Process (First Pass)

for each segment $i = 1 : total\_segments$:
  $labels\_seg_i = labels\_rec$
  for $j = 1 : length(labels\_rec)$:
    for each combination c_i of j labels:
      $recs(c_i)$ = recordings that contain only the labels in c_i
      if match_template$(seg_i, recs(c_i)) \geq 0.4$:
        $match(c_i) = \sum$ match_template$(seg_i, recs(c_i))$
      end **if**
    end **for**
    $C_i = \arg\max(match)$
    if isempty(C_i):
      continue for
    end **if**
  end **for**
  if isempty(C_i):
    $labels\_seg_i = MNF$
  else:
    $labels\_seg_i = \cap(C_i, labels\_seg_i)$
  end **if**
end **for**

ments and labels with no corresponding segments in a recording (c_{un}), we assign the unallocated labels to all the MNF segments. This can solve the issue of unallocated labels and MNF segments in a recording but does not completely eliminate the Unknown segments (MNF segments and segments with multiple labels), since more than one label may be unallocated and thus assigned to a single segment. Case 1 in Figure 3.1 depicts what happens during the Second Pass when there is an unallocated label (label B) and an MNF segment (segment 4). In this case, the unallocated label is to be assigned to segment 4. **Algorithm 2** describes the Second Pass.

Algorithm 2: Classification Process (Second Pass)

if unallocated label(s) c_{un} **and** any $labels\_seg_i = MNF$:

$labels\_seg_i = c_{un}$, $\forall MNF$ segments

end **if**

Third Pass

After reducing the MNF segments, there may still be labels unallocated in some recordings. Hence, the Third Pass of the classification process addresses the need for all labels of a recording to get assigned to at least one segment. More specifically, in a recording for which all segments have labels but some of the weak labels of the recording are not assigned to any segments, there must be some labels that are assigned, most likely incorrectly, to more than one segment. It is possible that more than one

segment may have the same label, but when a label is unallocated then we assume that one of the segments matched to the same label is falsely classified. We search for the best match for any unallocated label among the multiple segments of the rest of the labels. If a match is found, the label of the segment it derives from is changed to the unallocated label. An example of the Third Pass is depicted in Case 2 of Figure 3.1, where all segments have labels assigned to them, however label B is not assigned to any of them. The best match with label B is found within the segments that have the same label (segments 2, 3 and 4). Segment 4 has the max match of 0.57, thus label B is assigned to it. **Algorithm 3** explains the Third Pass.

Algorithm 3: Classification Process (Third Pass)

---

if unallocated label(s) c_{un} **and** |segments with label c_i| ≥ 2, $\forall c$:

$same(c_i)$ = all segments labelled c_i

$match(c_{un})$ = match_template($same(c_i), recs(c_{un})$)

Find segment seg_i with max($match(c_{un})$)

$labels\_seg_i = c_{un}$

end **if**

Evaluation on Synthetic Data

At that point in the PhD research, there was no public dataset with strong time-frequency labelling of each bird vocalisation. Thus, in order to evaluate our proposed method, we created a synthetic dataset D

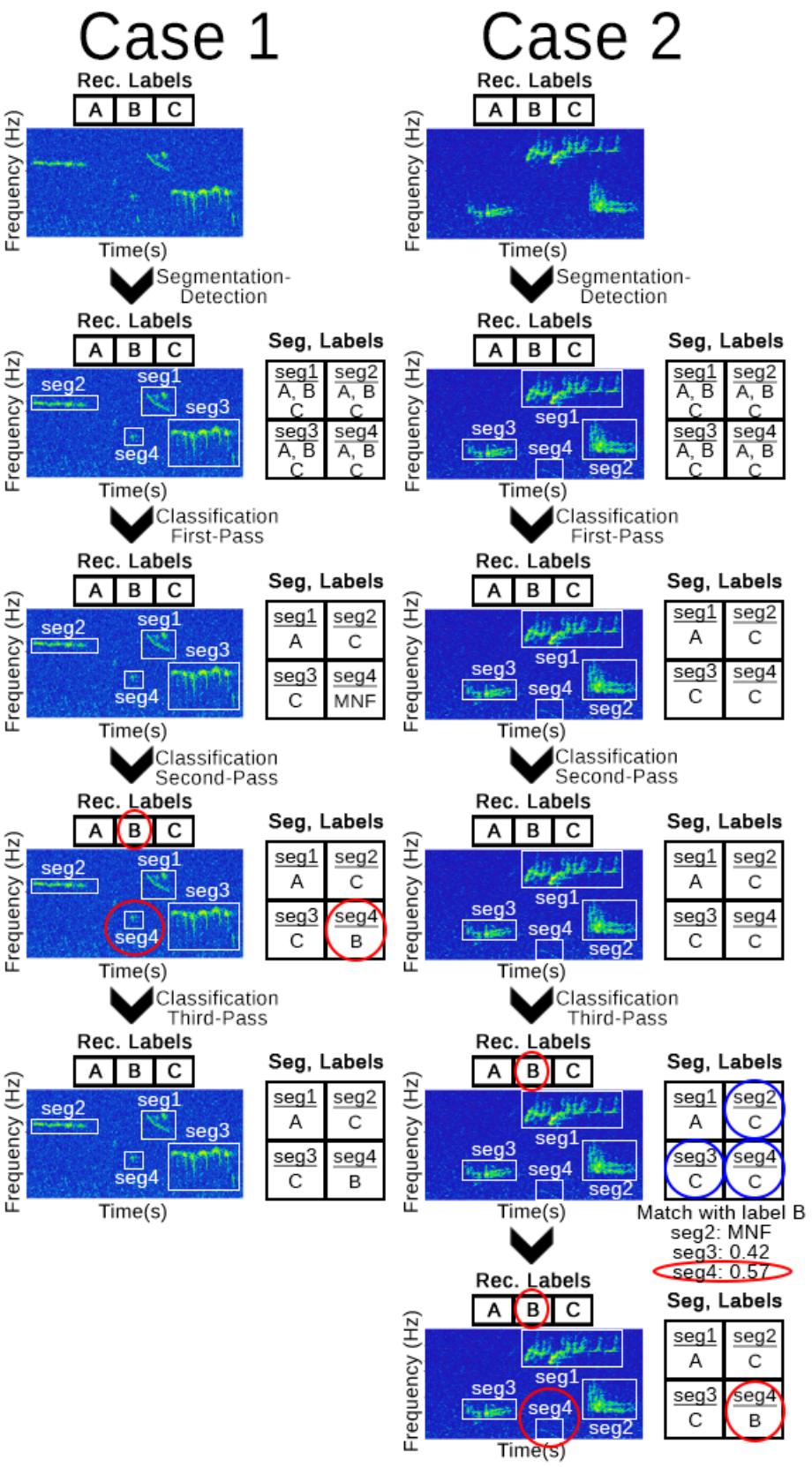


Figure 3.1: Example cases for the classification process. Case 1 describes what happens when there is an unallocated label and a segment with MNF label. Case 2 describes what happens when there is an unallocated label and multiple segments have one of the other labels.

where the boundaries of each vocalisation were known. The training audio dataset provided during the NIPS4B bird song competition of 2013 contained recordings that had already been weakly labelled. Since there was no per-unit annotation in it, we created a synthetic dataset of 50 recordings with vocalisations deriving from the single labelled recordings in the NIPS4B training dataset. Out of the 87 labels of the NIPS4B dataset, 51 had recordings that are labelled with only one species. For our synthetic dataset, each recording was 5 seconds long and it consisted of one of the recordings of NIPS4B with no labels as background, hence containing only natural background noise in it. Each synthetic recording was also allocated 2 to 4 randomly picked labels out of the above mentioned 51 labels. A source recording was randomly picked for each of the allocated labels and from that recording one segment produced by our proposed segmentation process was randomly placed in the synthetic recording by overlaying it to the background. Thus, each synthetic recording contained 2 to 4 segments and labels. The resulting dataset consisted of 50 recordings, with a total of 138 segments, hence a mean of 2.76 segments per recording. Any recording from the NIPS4B training dataset not used in the making of the synthetic data was used in order to search for the segment matches, hence providing our classification process with a broader variation of species' vocalisations than the one available by using only the synthetic dataset. The boundaries of each segment in the synthetic recordings were known, hence the following evaluation measures are only for the classification process and its different variants.

In Table 3.2, the results of the segment classification using all three

passes are depicted. The First Pass produced a correct classification of 69% and 6% of Unknown segments, the latest one included segments that were either not matched to any label, labelled as MNF, or segments that had more than one labels assigned to them. After the Second Pass of the algorithm, the percentage of Unknown segments was reduced to 4%, while the correctly classified segments were increased. Finally, after the Third Pass, we had a slight increase to the number of correct classifications, namely 4%, which led to the total result of 76% correctly classified segments.

Most of the misclassifications happened due to the fact that the segmentation process produced a lot of smaller segments that usually contained very simple vocalisations, and in many cases, fragments of vocalisations, that could be matched to multiple labels easily. In the event that the segments were part of vocalisations they were considered *out of context*. When there were out of context segments the classification results could be verified through a process of *inverse matching*. More explicitly, checking the recording where a match was found to see if it was matched to a single segment or a part of a bigger segment, by checking the area around where the match was found. If the segment was matched to part of a bigger vocalisation then it should have had the remaining of the vocalisation at a close by area in order for it to be considered a correct classification. However, inverse matching could not be applied in the synthetic dataset case, because the segments were chosen at random, so they were not placed together with the rest of the vocalisation. We discuss inverse matching in more detail in Section 3.2.1.

Table 3.2: Classification Results for D.

	Correct	**Wrong**	**Unknown**
Chance	36%	64%	—
First Pass	69%	25%	6%
Second Pass	71%	25%	4%
Third Pass	76%	20%	4%

Table 3.3: Classification Results for D_{1000}.

	Correct	**Wrong**	**Unknown**
Chance	33%	67%	—
First Pass	66%	22%	12%
Second Pass	71%	22%	7%
Third Pass	74%	19%	7%

In order to evaluate classification when the out of context problem does not occur as often, we created a second synthetic dataset D_{1000} of 50 recordings, where segment size $\geq$ 1000 time-frequency points. In this dataset, recordings were created in the same way as before and each recording contained 2 to 4 labels, with a total of 152 segments, hence a mean of 3.04 segments per recording. The results produced by the different classification steps are presented in Table 3.3. In the evaluation of the classification process using D_{1000} almost the same results as the one produced by dataset D can be noticed. This indicates that smaller segments were not the limiting factor in classification performance. In the D_{1000} results, even though the missclassifications of most of the smaller out of context segments were not present, still there were segments with simple structure (e.g. a straight line in frequency or time), which could get matched to larger vocalisations.

Inverse Template Matching

As mentioned above, for dataset D a lot of the misclassifications were caused by the smaller segments. In order to reduce these misclassifications and possibly even more caused when parts of the same vocalisation end up in different segments we propose an *inverse template matching* method. This inverse method runs after the original normalized correlation template matching process and its task is to check the broader context around a segment and try to see if it can get a match with any of the matches returned by *match_template*.

Figure 3.2 presents an example of how inverse matching works. Inverse matching considers all the matches found by *match_template* for a segment. The segmentation algorithm is performed for all of the recordings that the match is found in and it produces all the segments of that recording. Hence, we acquire the actual borders of the segment where a match is found. Next, *match_template* tries to match these segments back to the original segment in question. If an inverse match is found then the labels of this segment are considered for classification of the original segment in question (**Algorithm 1**).

Evaluation on Natural Data

It is easier to evaluate inverse matching on natural data, since it can be hard to synthesise data that preserve the broader context of vocalisation segments. Hence, in order to evaluate and compare the results of inverse matching and the original *match_template* method along with the three

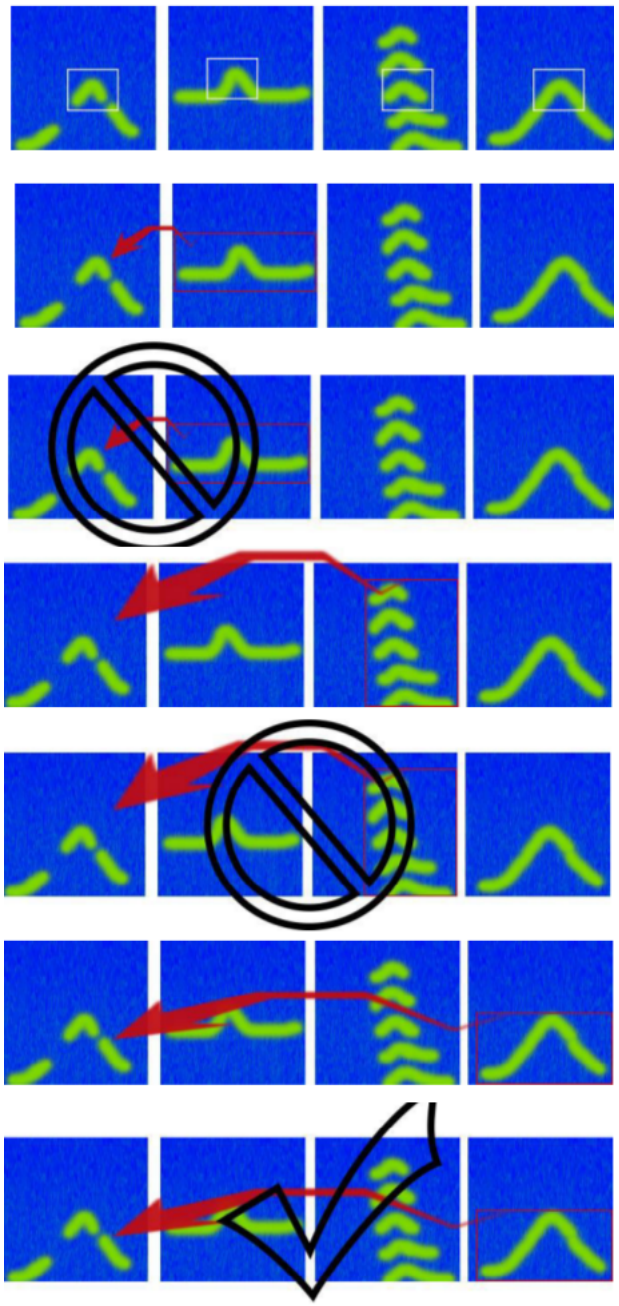


Figure 3.2: An example of inverse matching. Col 1: segment that goes through the inverse matching method. Row 1, cols 2-4: possible matches returned from *match_template*. Row 2: Inverse matching tries to match the whole segment of col 2 to col 1 segment. Row 3: no match found. Row 4: Inverse matching tries to match the segment of col 3 to col 1 segment. Row 5: no match found. Row 6: Inverse matching tries to match the segment of col 4 to col 1 segment. Row 7: match found, the labels of the recording containing this segments will be used in deductive label refinement (**Algorithm 1**).

Table 3.4: Evaluation of classification when we consider the segments produced by the segmentation method to be correct.

		Correct	Wrong	Unknown	Total Segs
No Inverse Matching	**Original Method**	49%	32%	19%	766
	NL_Means	50%	32%	**18%**	726
	Chambolle_TV	**51%**	**31%**	**18%**	710
Inverse Matching	**Original Method**	44%	**26%**	30%	766
	NL_Means	45%	27%	28%	726
	Chambolle_TV	**46%**	27%	**27%**	710

different denoised segmentations, we used the subset of NIPS4B training data defined in Section 3.1.2. For the evaluation of classification methods we used the 28 individual species labels, instead of a single positive or negative label. Two different evaluations follow, one for classification only and one for the whole process of segmentation and classification.

Firstly, we considered all the segments produced from the segmentation step to be correct and we only evaluated the classification with and without the effects of inverse template matching. Table 3.4 presents these results. The different segmentation methods produced different number of segments. Also none of them had very different results from the other two. Another thing to take into account from this table is that using inverse matching did not have the results we expected: misclassifications were reduced, however correct classifications were also reduced leading to more Unknown segments (segments with no labels or more than one labels assigned to them).

The second evaluation for the deductive label refinement algorithm measured the overall and class-wise metrics for the whole two-step process of finding the segments and assigning them with the correct label.

Table 3.5: Overall metrics for evaluation of segmentation and classification.

		Overall Metrics (micro-average)		
		F-measure	**Precision**	**Recall**
No Inverse Matching	**Original Method**	55%	54%	**56%**
	NL_Means	55%	55%	55%
	Chambolle_TV	**56%**	**56%**	55%
Inverse Matching	**Original Method**	52%	51%	**53%**
	NL_Means	52%	51%	**53%**
	Chambolle_TV	**53%**	**53%**	52%

Table 3.6: Overall metrics for evaluation of segmentation and classification.

		Class-wise metrics (macro-average)		
		F-measure	**Precision**	**Recall**
No Inverse Matching	**Original Method**	53%	65%	**53%**
	NL_Means	**55%**	**72%**	53%
	Chambolle_TV	53%	65%	52%
Inverse Matching	**Original Method**	53%	62%	52%
	NL_Means	**54%**	**66%**	**53%**
	Chambolle_TV	53%	63%	52%

Tables 3.5 and 3.6 present these results overall and class-wise. In both the overall and class-wise results the performance between different denoising methods did not have any great influence over the results, however in all cases when using inverse template matching the F-measure value dropped. This can be explained by the results of Table 3.4 where it is apparent that the number of Unknown segments was increased. All Unknown segments were considered as wrongly classified overall since no actual Unknown label is present in the dataset.

3.2.2 Convolutional Neural Network

As an alternative classification paradigm we consider supervised machine learning. Neural networks and deep neural networks with a large number of parameters are very powerful machine learning systems. Neural networks are very diverse and can learn very complicated, nonlinear relationships between their input and output. This makes them perfect for classification tasks and more specifically, in our case, vocalisation classification using the corresponding spectrogram representation.

For the second classification method we implement a convolutional neural network (CNN) (Section 2.2.3). The structure that we use is a CNN that consists of two convolutional and max-pooling stages, a fully-connected hidden layer and a fully-connected output layer as seen in Figure 3.3. First, we apply a convolution layer which consists of 32 filters of size 5x5. The nonlinearity chosen is the linear rectifier, so we obtain rectified linear units (ReLUs). In concern to the weight initialization, we choose the GlorotUniform initializer (Glorot and Bengio, 2010) that provides a uniform distribution with a carefully chosen range. We then apply max-pooling of factor 2 in both dimensions. We add another convolution and pooling stage like the ones before. Then, a fully-connected layer of 256 units with 50% dropout on its inputs is added and finally, a 51-unit softmax output layer, again with 50% dropout.

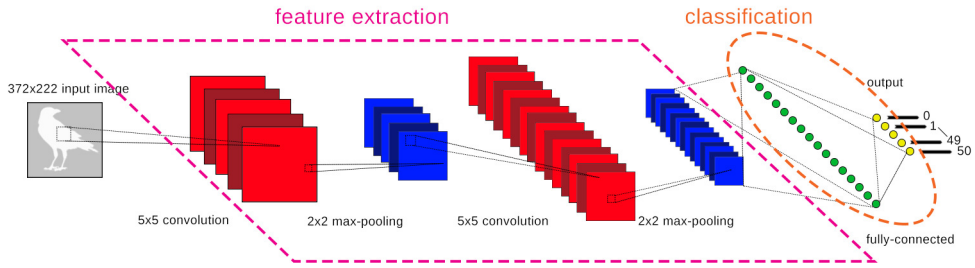


Figure 3.3: The structure of the convolutional neural network used for bird vocalisation classification.

Evaluation on Segmentation Data

As input for training our CNN we used the segments produced by our segmentation process. We zero padded all segments to the maximum width and height across all the segments produced. The segment in question was placed in the middle of the zero padding. Examples of the resulting segments that the train, validation and test datasets consist of are depicted in Figure 3.4. For each input image the corresponding target value (label) was also saved, in order to use in the training, validation and testing of the CNN.

In order to evaluate our CNN method, only segments deriving from single-labelled recordings in the dataset provided by the Neural Information Processing Scaled for Bioacoustics (NIPS4B) bird song competition of 2013, were used, because we needed to known the label of each segment in order to train the network. From the original 87 labels of the NIPS4B dataset, 51 of them were present in at least one recording where there were no other labels. In total, 230 recordings were used to extract the 1285 vocalisations that were then used to create three datasets (i.e.

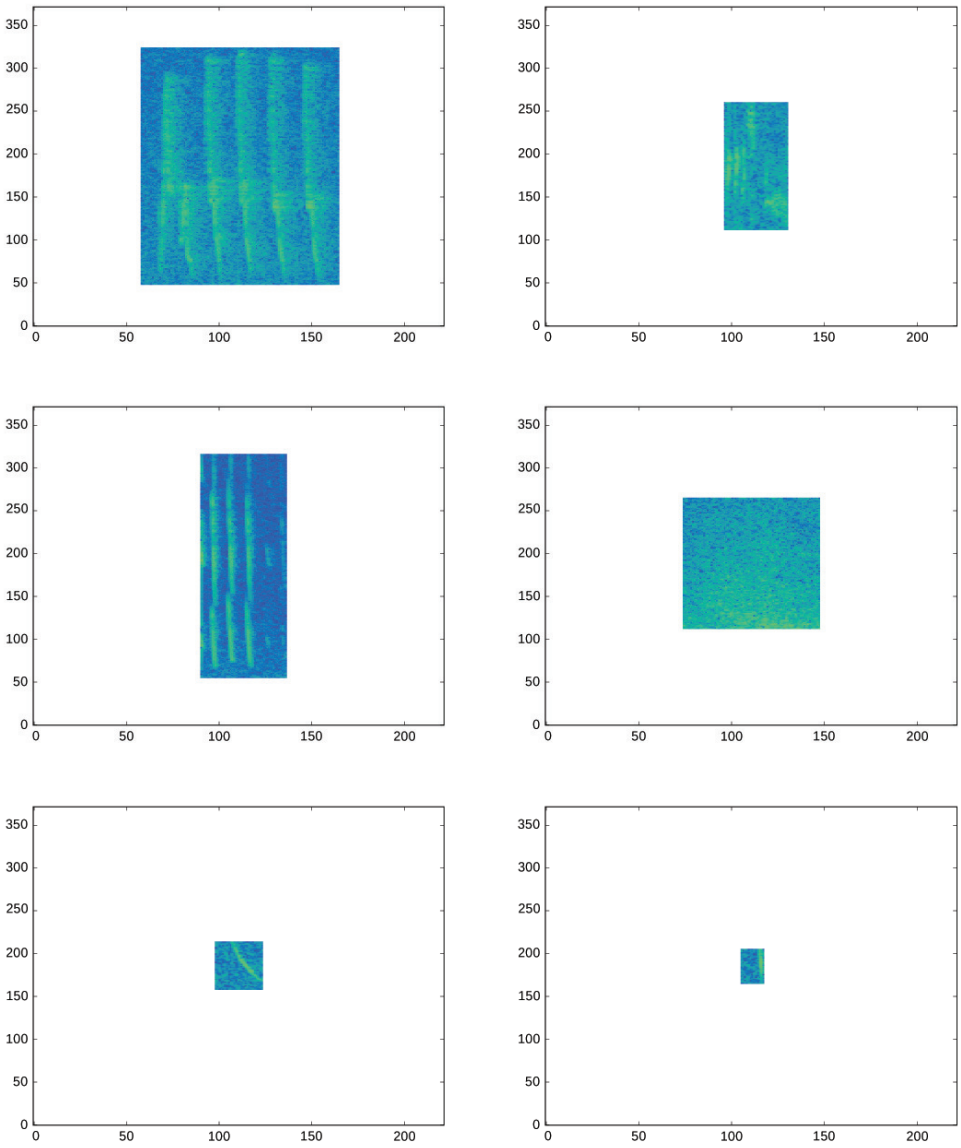


Figure 3.4: Examples segments that make up the train, validation and test datasets.

train, validation, test). The train, validation and test sets consisted of 785, 100 and 400 segments, respectively.

During training mini-batches of 50 segments were used. The loss of our network was minimized during the training and it was defined as the categorical cross-entropy loss between the network output and the targets (correct labels) and was computed as the mean of the loss over a mini-batch. For updating our CNN, we used stochastic gradient descent (SGD) with Nesterov momentum. For monitoring progress during training, after each epoch, we evaluated the network on the validation set. A slightly different loss expression was used in order to do so. The difference from the train loss was that all nondeterministic layers were switched to a deterministic implementation, so in our case, we disabled the dropout layers when computing the predictions for the validation set.

The result of the classification accuracy on the test dataset was 27.25%. This relatively low result could be most likely due to overfitting (see Figure 3.5), which was anticipated due to the small size of the dataset. Even though the results are not very satisfying, there are many different methods and deep learning architectures that could be implemented in order to obtain better results.

However, the restrictions imposed by the input of this network limit our possible improvements. Firstly, the input of this network is not ideal as can be noticed in Figure 3.1. Due to the varying sizes of the segments produced by segmentation and the restriction of the input of a network needing to be of the same shape, a lot of zero-padding was used for the input segments. Also, in order to train the network we need

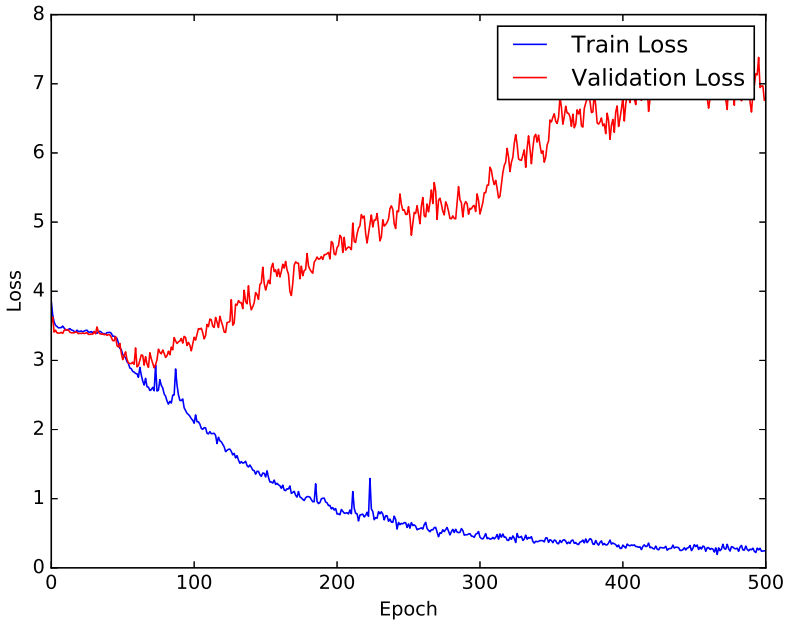


Figure 3.5: Loss value of CNN for the train dataset (blue line) and the validation dataset (red line).

segments that are produced from single-labelled recordings and not all 87 labels have single-labelled recordings. Due to these restrictions, a network that can have the whole recording as an input regardless of how many labels are assigned to it, will be more flexible and generalizable. In Chapter 5 we propose some deep learning approaches that do not face the same restrictions. However, in order for any generalizable deep learning approach to be successful we need strongly annotated data in order to evaluate the results. In the following chapter we give a detailed presentation of the NIPS4B data and also discuss our method of acquiring the strong labels for it.

Chapter 4

A Richly-Annotated Birdsong Audio Dataset

In the field of automatic birdsong monitoring, advances in birdsong detection and classification have approached a limit due to the lack of fully annotated datasets. The lack of strong annotations imposes restrictions on the methods we can use to achieve our aim. As we explained in previous chapters, acquiring strong annotations for any type of audio dataset is a laborious task, and especially for bird soundscapes where vocalisations are highly diverse, originating from multiple birds and can overlap in time. Hence, much time from an expert annotator is required for these kind of bird soundscapes. This results in many ecological datasets lacking temporal annotations of bird vocalisations even though they are vital to the training of automated methods that predict detailed annotations which could potentially remove the need for a human annotator.

Recently, BirdVox-full-night (Lostanlen et al., 2018), a dataset containing some temporal and frequency information about flight calls of nocturnally migrating birds, was released. The BirdVox-full-night dataset contains 6 far-field, full night recordings, containing 35,000 flight calls from 25 species of passerines recorded around Ithaka, New York. However, BirdVox-full-night only focuses on avian flight calls, a specific type of bird calls, that usually have a very short duration in time. The temporal annotations provided for them don't include any onset, offset or information about the duration of the calls, they simply contain a single time marker at which a flight call is active. Additionally, there is no distinction between the different bird species, hence no specific species annotations are provided: only the time and frequency location of flight calls through the duration of a recording is denoted. Hence, the dataset can provide data to train models for flight call detection but is not appropriate for models performing both event detection and classification for a variety of bird vocalisations.

In this chapter, we introduce NIPS4Bplus, the first ecological audio dataset that contains bird species tags and temporal annotations, which can be freely accessed online at `https://doi.org/10.6084/m9.figshare.6798548`, and can be used for either training supervised automated methods that perform bird vocalisation detection and classification or be used for evaluating methods that use only audio tags or no annotations for training.

4.1 Audio Data Collection

In 2013, during the Neural Information Processing Scaled for Bioacoustics (NIPS4B) challenge for bird song classification a training and testing dataset that contains multiple bird species was made public. For our previous experiments we used the NIPS4B training dataset and the weak labels provided by the challenge organisers or a smaller set that we partially annotated for our experiments. In this section we will describe both the original dataset collected for the 2013 challenge and our process of acquiring the strong annotations that we will late use for our next experiments.

The recordings that comprise the NIPS4B 2013 training and testing dataset were collected by recorders placed in 39 different locations, which can be summarised by 7 regions in France and Spain, as depicted in Figure 4.1. 20% of the recordings were collected from the Haute-Loire region in Central France, 65% of them were collected from the Pyrénées-Orientales, Aude and Hérault regions in south-central France along the Mediterranean cost and the remaining 15% of the recordings originated from the Granada, Jaén and Almeria regions in eastern Andalusia, Spain. The Haute-Loire area is a more hilly and cold region, while the rest of the regions are mostly along the Mediterranean coast and have a more Mediterranean climate.

The recorders used to acquire the recordings were the SM2BAT using SMX-US microphones. They were originally put in the field for bat echolocation call sampling, but they were also set to record for 3 hours

single channel at 44.1 kHz sampling rate starting 30 minutes after sunrise, right after bat sampling. The recorders were set to a 6 dB Signal-to-Noise Ratio (SNR) trigger with a window of 2 seconds, and acquired recordings only when the trigger was activated.

Approximately 30 hours of field recordings were collected. Any recording longer than 5 seconds was split into multiple 5 second files. SonoChiro, a chirp detection tool used for bat vocalisation detection, was used on each file to identify recordings with bird vocalisations.[1] A stratified random sampling was then applied to all acquired recordings, based on locations and clustering of features, to maximise the diversity in the labelled dataset, resulting in nearly 5000 files being chosen. Following the first stage of selection, manual annotations were produced for the classes active in these 5000 files and any recordings that contained unidentified species' vocalisations were discarded. Furthermore, the training set and testing set recordings were allocated so that the same species were active in both. Finally, for training purposes, only species that could be covered by at least 7 recordings in the training set were included in the final dataset, the rest were considered rare species' occurrences that would make it hard to train any classifier, hence were discarded. The final training and testing set consist of 687 files of total duration of less than an hour, and 1000 files of total duration of nearly two hours, respectively.

[1] `http://www.leclub-biotope.com/fr/72-sonochiro`

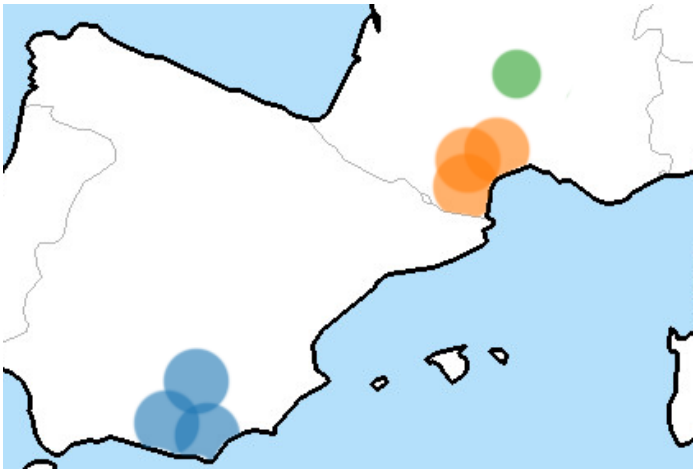


Figure 4.1: Regions where the dataset recordings were collected from. Green indicates Central France region Haute-Loire. Orange indicates Southern France regions Pyrénées-Orientales, Aude and Hérault. Blue indicates Southern Spain regions Granada, Jaén and Almeria.

4.2 Annotations

4.2.1 Tags

The labels for the species active in each recording of the training set were initially created for the NIPS4B 2013 bird song classification challenge (Glotin et al., 2013). There is a total of 61 different bird species active in the dataset. For some species we discriminate between song, call and drum. We also include some species living with these birds: 7 insects and an amphibian. This tagging process resulted in 87 classes. A detailed list of the class names and their corresponding species English and scientific names can be found in `https://doi.org/10.6084/m9.figshare.6798548`. These tags only provide information about the species active in a recording and do not include any temporal information, hence they are treated as weak labels for this dataset. In addition to the recordings

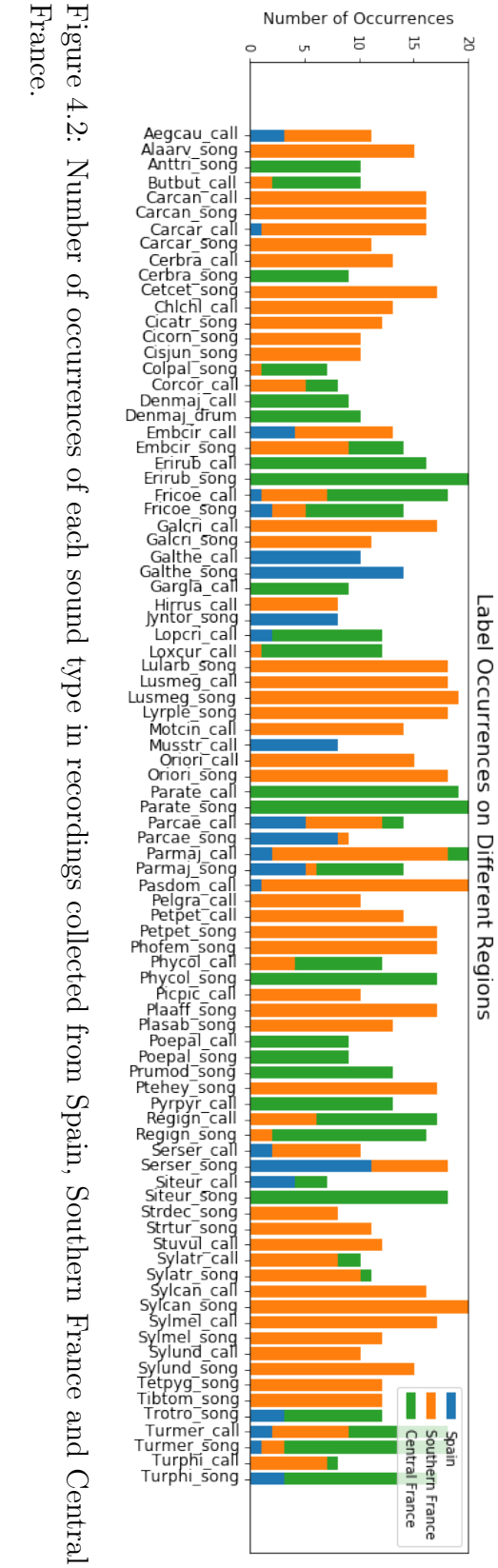


Figure 4.2: Number of occurrences of each sound type in recordings collected from Spain, Southern France and Central France.

containing bird vocalisations, some training files only contain background noise acquired from the same regions and have no bird song in them, these files can be used to tune a model during training. Figure 4.2 depicts the number of occurrences per class for recordings collected in each of the 3 different general regions of Spain, South France and Central France. Each tag is represented by at least 7 up to a maximum of 20 recordings.

Each recording that contains bird vocalisations includes 1 to 6 individual labels. These files may contain different vocalisations from the same species and also may contain a variety of other species that vocalise along with this species. Figure 4.3 depicts the distribution of the number of active classes in the dataset.

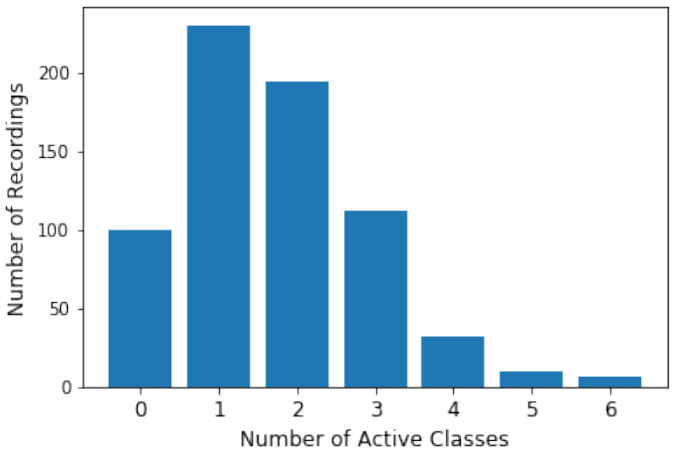


Figure 4.3: Distribution of number of active classes in dataset recordings.

Figure 4.4 depicts the number of co-occurrences between pairs of labels. We can notice that there are no notable patterns to the ways species vocalisations co-occur. One interesting thing one can notice while studying the co-occurrence heat map is that there is no strong correlation between calls and songs from the same species, this is due to the different functions between calls and songs produced. As calls may be related

to self-maintenance activities such as species identification or holding the flock together, while songs are mostly used for attracting a mate, establishing territories, intimidating enemies and learning through imitations and practising.

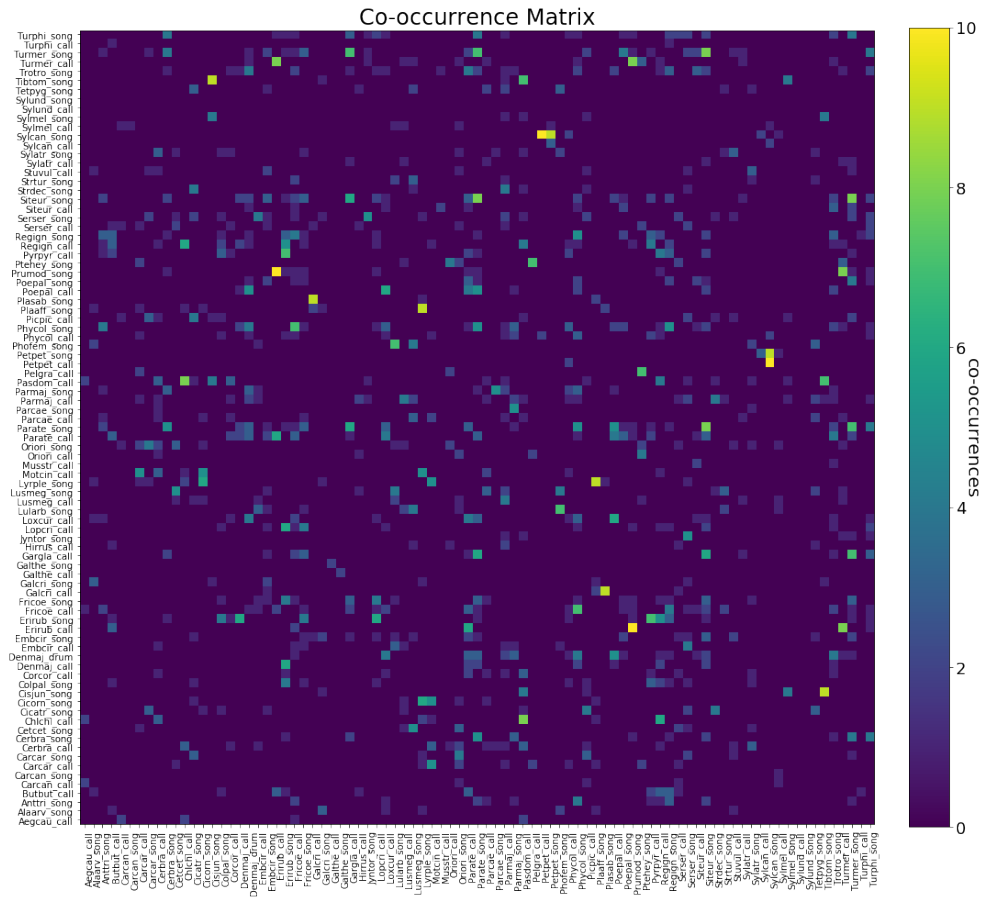


Figure 4.4: Co-occurrence heat map for the labels of the dataset.

4.2.2 Temporal Annotations

Temporal annotations for each recording in the training set of the NIPS4B dataset were produced manually using Sonic Visualiser.[2] The temporal

[2] https://www.sonicvisualiser.org/

annotations were made by a single bird expert annotator, Hanna Pamuła, and can be found in `https://doi.org/10.6084/m9.figshare.6798548`. Table 4.1 presents the temporal annotation format as is provided in NIPS4Bplus and Figure 4.5 depicts a visual representation of the temporal annotations. For the experiments in this thesis, temporal annotations and no annotations in the frequency axis were needed, hence these are the strong labels we will use in the work that follows.

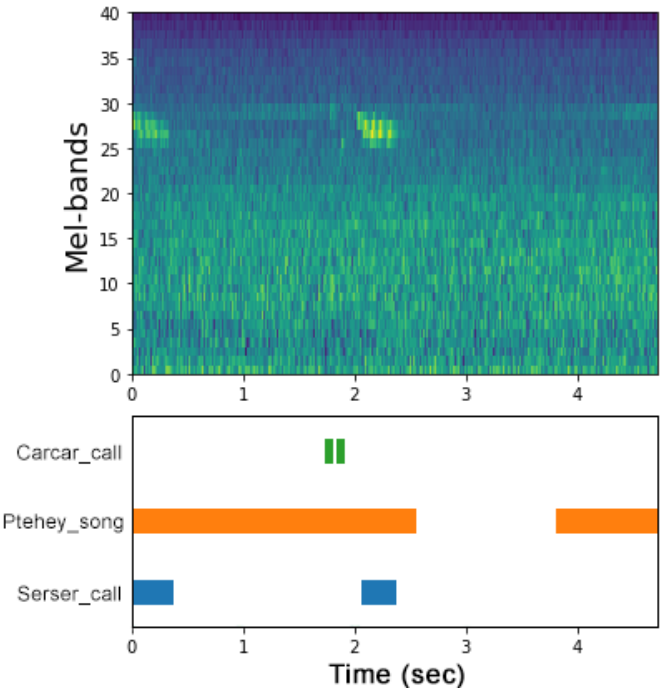


Figure 4.5: Mel-band spectrogram of a recording in NIPS4Bplus and the visual representation of the corresponding temporal annotations as noted in Table 4.1.

Regarding the temporal annotations for the dataset, we should mention the following:

- The original tags were used for guidance, however some files were judged to have a different set of species than the ones given in the

Table 4.1: NIPS4Bplus temporal annotations of the recording depicted in Figure 4.5.

Starting Time (sec)	Duration (sec)	Tag
0.00	0.37	Serser_call
0.00	2.62	Ptehey_song
1.77	0.06	Carcar_call
1.86	0.07	Carcar_call
2.02	0.41	Serser_call
3.87	1.09	Ptehey_song

original metadata. Similarly, in a few rare occurrences, despite the tags suggesting a bird species active in a recording, the annotator was not able to detect any bird vocalisation.

- An extra 'Unknown' tag was added to the dataset for vocalisations that could not be classified to a class.

- An extra 'Human' tag was added to a few recordings that have very obvious human sounds, such as speech, present in them.

- Out of the 687 recordings of the training set 100 recordings contain only background noise, hence no temporal annotations were needed for them.

- Of the remaining 587 recordings that contain vocalisations, 6 could not be unambiguously labelled due to hard to identify vocalisations, thus no temporal annotation files were produced for them.

- An annotation file for any recording containing multiple insects does not differentiate between the insect species and the 'Unknown' label was given to all insect species present.

- In the rare case where no birds were active along with the insects no annotation file was provided. Hence, 7 recordings containing only insects were left unlabelled.
- In total, 13 recordings have no temporal annotation files. These can be used when training a model that does not use temporal annotations.
- On some occasions, the different syllables of a song were separated in time into different events while in other occasions they were summarised into a larger event, according to the judgement of the expert annotator. This variety could help train an unbiased model regarding separating events or grouping them together as one continuous time event.

As mentioned above, each recording may contain multiple species vocalising at the same time. This can often occur in wildlife recordings and is important to be taken into account when training a model. Figure 4.6 presents the fraction of the total duration containing overlapping vocalisations, as well as the number of simultaneously occurring classes.

In the following chapter, we propose methods that use data such as these NIPS4Bplus recordings and annotations. We treat the dataset as a low-resource one, due to the relatively limited amount of training data and use the strong labels for evaluation purposes only, hence having only the weak labels during training.

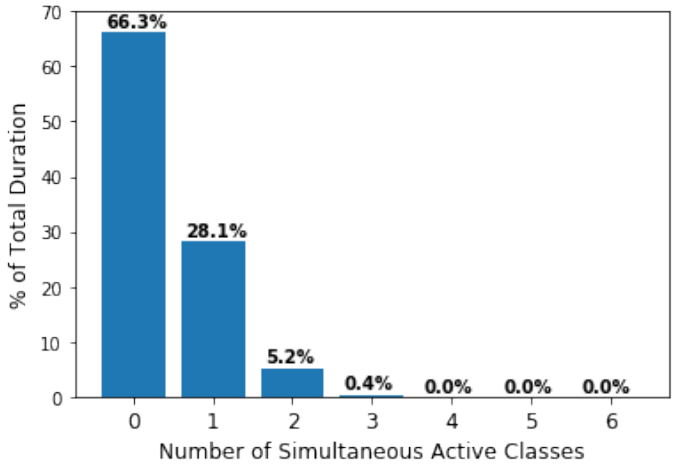


Figure 4.6: Distribution of simultaneous number of active classes on the total duration of the recordings.

Chapter 5

Deep Learning for Detection and Classification of Bird Sounds via Task Factorisation

Training a neural network to predict an audio transcription using a low-resource dataset, such as the one described in Chapter 4, can sometimes prove to be impractical. A network needs to have enough parameters to be able to predict all the different classes without ignoring any rare events, but also be small enough or have just the right amount of regularisation as to not overfit the limited amount of training data available. In Choromanska et al. (2015), the authors discuss such difficulties of training both large- and small-size networks. Predicting full audio transcription becomes even harder when the task is a weak-to-strong prediction where the network needs to predict full transcriptions from weak labels as described in Section 2.4.

Given a training problem, finding an appropriate network and input encoding for that problem is of great importance, but it is not something easily solved (Blum and Rivest, 1992). Hence, even though there is no specific way of defining a network and type of training that ensures that a transcription will be predicted successfully, a full transcription task can be redefined as multiple intermediate tasks of audio event detection and audio tagging that might be easier to train a network for, when using a low-resource dataset. A similar approach is used to enhance the performance of automatic speech transcription (Yu and Deng, 2016) by using speaker diarisation (Anguera et al., 2012; Garcia-Romero et al., 2017) and speaker recognition (Tirumala and Shahamiri, 2016) systems together in order to structure an audio stream into speaker turns and provide the speaker's true identity, respectively. However, these speech approaches are highly customised to characteristics of speech signals. Our method is focused on general low-resource audio with speech events considered just a single class amongst other audio events without distinguishing between individual speakers.

In this chapter, we first contact a pilot study of full transcription approaches using different model architectures and then we propose a factorisation of the full transcription task into multiple simpler intermediate tasks of audio event detection and audio tagging in order to predict an intermediate transcription that can be used to boost the performance on the full transcription task. For each intermediate task, we propose a training setup to optimise their performance and, finally, we train the intermediate tasks independently and in two multi-task learning settings

and compare their results. More specifically, we introduce three novel aspects: a task factorisation of the full audio transcription task for low-resource data scenarios; a new multi instance learning loss function that trains more reliably than the standard one; and an approach to multi-task learning that allows for the training data being different within each task, alleviates the issue of having to balance the values of different loss functions while maintaining all the advantages of hard parameter sharing between tasks.

5.1 Pilot Study on Full Transcription Using Current Deep Learning Architectures

As our first pilot study in predicting a full transcription in a weak-to-strong MIL setting when using a low-resource dataset, we use multiple state-of-the-art neural network architectures: a deep convolutional neural network (CNN) with similar structure to the one in Figure 3.3; a DenseNet (Huang et al., 2017) architecture as depicted in Figure 5.3; and a stacked convolutional and recurrent neural network (CRNN) with the structure of Figure 5.1. In these pilot studies many configurations failed to train effectively and thus we do not present numerical results from the pilot phase.

As input for training and validation we used the spectrograms of 513 recordings from the NIPS4Bplus data and used the remaining 187 record-

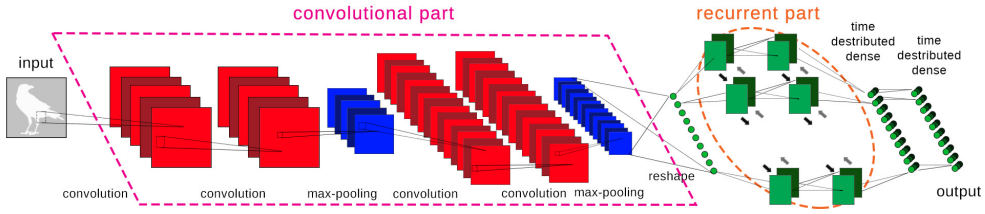


Figure 5.1: CRNN architecture used for our pilot studies on the NIPS4Bplus data. First layers perform convolutional transformations and max pooling, followed by the recurrent part of the network that consists of bidirectional GRUs, and the dense layers that predict the final transcription.

ings for evaluation. We trained and evaluated a few different configurations of all three architectures however the predictions for full transcription would usually show no distinct or useful results for different classes between recordings. This led us to the conclusion that simply training a network for the full transcription task in this kind of setting might be infeasible.

The difficulty of predicting full transcription for a low-resource dataset is mainly due to the lack of a large amount of reference recordings for each class. Furthermore in the NIPS4Bplus data, there is a large amount of different classes with large variations in the type of audio event produced by each, hence this adds an additional difficulty for the network to differentiate between them.

In order to help each network focus on a specific type of vocalisation, we explore single class transcription for each individual class. We formulate the task as a multi instance learning problem in order to make use of the weak labels provided for each recording to predict the strong temporal annotations. For each network, the spectrograms of the record-

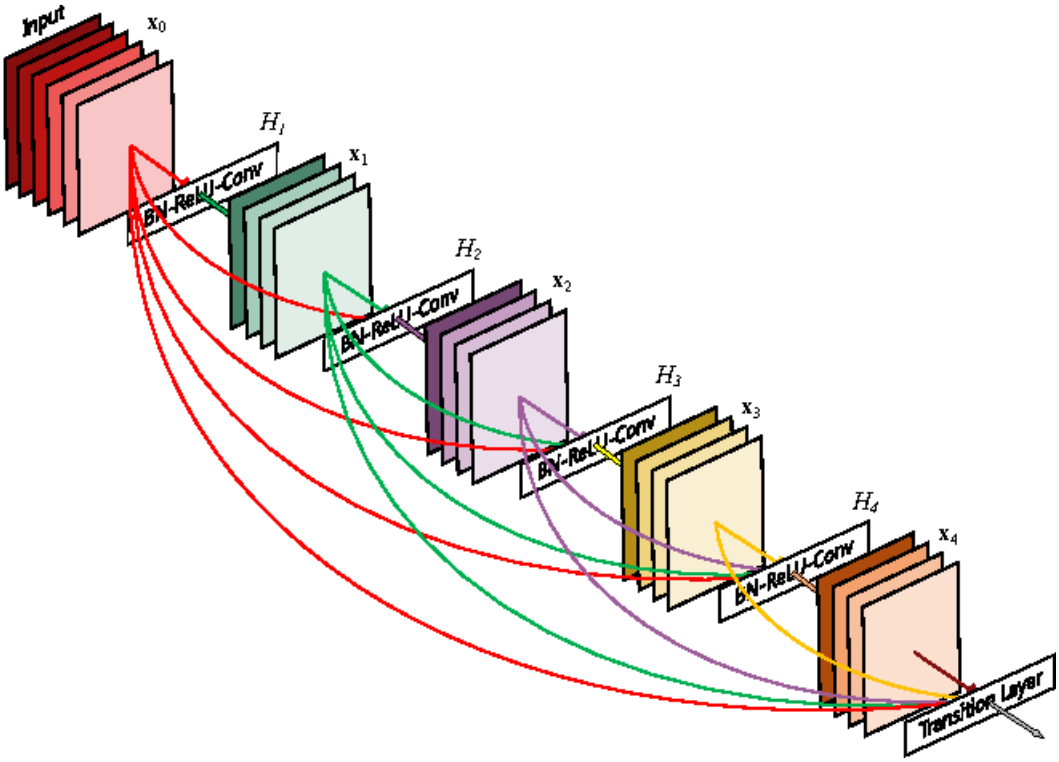


Figure 5.2: A 5-layer dense block used in our implementation of a DenseNet for our pilot studies on the NIPS4Bplus data. Each layer takes all preceding feature maps as input. *Image credit to (Huang et al., 2017)*

ings containing the class in question are used as input, and the output is a binary transcription of the recording, denoting which time frames contain a vocalisation from the class and which do not. For this pilot study on single class transcription, we use a DenseNet. DenseNets ensure maximum information flow between layers in the network, as they have dense blocks which connect all layers inside the block directly to each other. To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature maps to all subsequent layers. Figure 5.2 illustrates the layout of a dense block schematically. This produces $\frac{L(L+1)}{2}$ connections in a network with L number of layers. A possibly counter-intuitive feature of the DenseNet is that it requires fewer parameters than traditional networks, hence it was the more suitable for this pilot study compared to CNN and CRNN, due to the limited amount reference recordings for each label. Although the number of connections grows quadratically with depth, the topology encourages heavy feature reuse. In DenseNets, all layers have direct access to every feature map from all preceding layers, which means that there is no need to re-learn redundant feature maps. Consequently, DenseNet layers are very narrow and only add a small set of feature maps to the collective knowledge of the whole network while keeping the remaining feature maps unchanged. The DenseNet architecture explicitly differentiates between information that is added to the network and information that is preserved. The final classifier makes a decision based on the entire knowledge of the network. Although they follow a simple generative rule, DenseNets are very general and easy to train. One big advantage of the DenseNet is its improved flow of information and gradients throughout

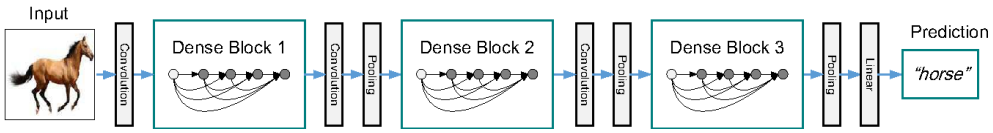


Figure 5.3: A deep DenseNet with three dense blocks used for our pilot studies on the NIPS4Bplus data. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling. *Image credit to (Huang et al., 2017)*

the network. Each layer has direct access to the gradients from the loss function and the original input signal.

For our task, we adapted the structure of Figure 5.3 into preserving the time axis, by using 1D max pooling, and changing the last layer output into a vector that represents the labels for each time frame, instead of one label for the whole recording. We trained a network for each species in the NIPS4Bplus dataset. However, due to the very limited amount of positive recordings for each species (7 to 20 positive recordings per species out of a total of 687 recordings) the results were not satisfactory. Furthermore, data augmentation, by mixing negative recordings (total number of 100 recordings) with the positive ones for each class, did not seem to boost the network performance.

Since both prediction of full transcription and prediction for each class separately fail to produce any useful results we next consider another approach to our problem. The factors causing the two previous approaches to fail can be mainly attributed to the large amount of classes and the few amount of reference recordings per class. However, for the task of bird vocalisation transcription the different vocalisations should have some similarities due to the fact that they are all produced by birds. Taking

that into account along with the nature of transcription tasks, in the following section we propose a task factorisation setting that can work in low-resource scenarios.

5.2 Factorisation

A full audio transcription task can be described as a combination of audio event detection and event classification. In order to properly train a full transcription network, we need a large amount of data that is not available in a low-resource dataset. Since it is very hard to train a network to predict full transcription on a low-resource dataset, we factorise the final task of full transcription into intermediate tasks that can predict an intermediate transcription matrix that can later be used to boost the performance of a full transcription network. Figure 5.4 depicts the overall task factorisation into the intermediate tasks and how they interact with the final task of full transcription. We define a WHEN network that performs audio event detection considering all classes as one general class; in other words, it predicts when any event was present without taking into consideration the different event classes. We also define a WHO network that performs audio tagging without predicting any temporal information. By combining the two different predictions from these networks, we create an intermediate transcription that provides us with the events present in a recording and the times where any of these events can be present in a recording. This intermediate transcription is to be used as supplementary information when training the full transcription

network in order to improve its performance by focusing its attention to the classes present in a recording and the time frames that may contain them.

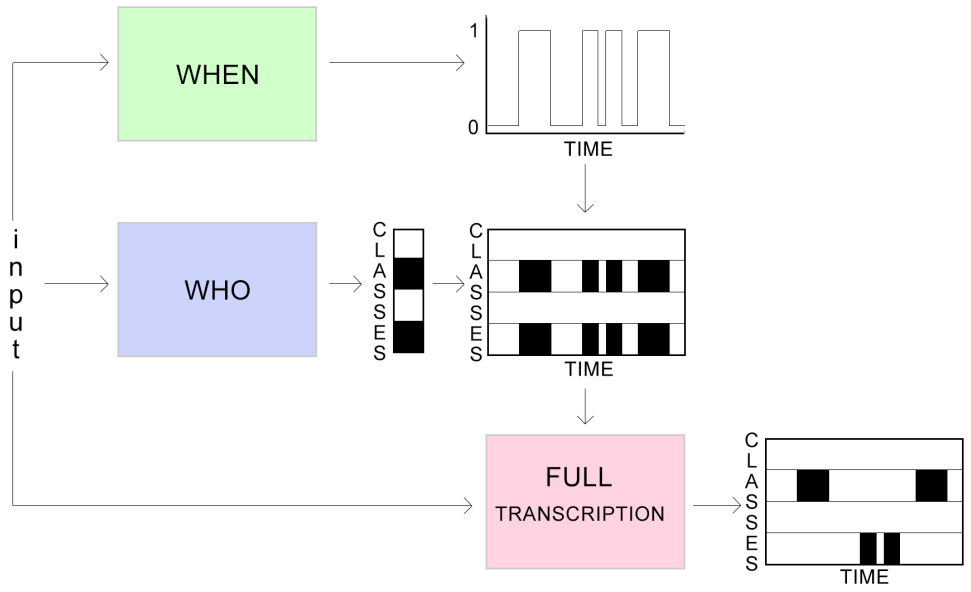


Figure 5.4: Proposed factorisation of the full transcription task into multiple simpler tasks. The WHEN network performs audio event detection considering all labels as one label. The WHO network performs audio tagging for all available labels. The predictions of WHEN and WHO produce an intermediate transcription that is used to boost the performance of the full transcription network.

When using a large enough dataset that provides satisfactory training data and has a good representation for each different class, many methods have been successful in performing both of the intermediate tasks. Examples for audio event detection can be found in Fanioudakis and Potamitis (2017); Schlüter (2016), while for audio tagging in Kong et al. (2017b); Xu et al. (2017a,b); Adavanne et al. (2017); Pons et al. (2018); Choi et al. (2016). These tasks are less challenging to train for than a full transcription task. However, using a low-resource dataset can

degrade their performance. Hence, in order to achieve a satisfactory performance when training with a low-resource dataset, we propose some specific training setups and techniques. The rest of this section describes in detail the task specific setups and techniques that we use to reach our goal.

5.3 WHEN: Audio Event Detection

In our proposed task factorisation, the WHEN network performs a single class audio event detection as the first intermediate task towards full transcription. For a multi-class dataset, training a separate network for each class in order to perform single class event detection can sometimes boost the performance of the detector as it only needs to learn the characteristics of a single class. However, in a low-resource dataset, training an audio event detector for each class can be nearly impossible. The number of classes might be too large, making it a time-consuming task. Furthermore, some of the classes might have very rare occurrences, limited to only a couple of recordings, hence making it infeasible to train a neural network for them. Nevertheless, many low-resource datasets are used for discriminating subclasses of a general class e.g., song of different bird species, sound of different car engines, barking of different dog breeds, and notes produced by an instrument. These subclasses usually share some common features and characteristics, hence, in order to achieve a good performance in the audio event detection task, we propose considering all subclasses as one general class and train a single WHEN network

to perform single class event detection. This reduces the training time compared to training one network for each subclass and also resolves any training issues caused by rare events.

5.3.1 Neural Network Architecture

From our pilot studies (see Section 5.1), we concluded that due to the nature of sounds having a structure in time it is best to have a network that can model the short-term and long-term temporal dependencies for each recording. Hence for our audio event detector we use a state-of-the-art stacked convolutional and recurrent neural network (CRNN) architecture, instead of a purely convolutional network (i.e. CNN or DenseNet). Table 5.1 describes the parameters of the proposed architecture.

The log mel-band energy feature extracted from the audio is fed to the neural network, which sequentially produces the predicted strong labels for each recording. The input to the proposed network is a $T \times M$ feature matrix. The convolutional layers in the beginning of the network are in charge of learning the local shift-invariant features of this input. We use a 3×3 receptive field and the padding arguments set as 'same' in order to maintain the same size as the input in all our convolutional layers. The max-pooling operation is performed along the frequency axis after every convolutional layer to reduce the dimension of the feature matrix while preserving the number of frames T. The output of the convolutional part of the network is then fed to bidirectional gated recurrent units (GRUs) with tanh activation to learn the short-term and long-term temporal

Table 5.1: WHEN network architecture. Size refers to either kernel shape or number of units. #Fmaps is the number of feature maps in the layer. Activation denotes the activation used for the layer and l2_reg the amount of l2 kernel regularisation used in the layer.

Layer	**Size**	**#Fmaps**	**Activation**	**l2_reg**
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Max Pooling	1×5	-	-	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Max Pooling	1×4	-	-	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Max Pooling	1×2	-	-	-
Reshape	-	-	-	-
Bidirectional GRU	64	-	tanh	0.01
Bidirectional GRU	64	-	tanh	0.01
Time Distributed Dense	64	-	ReLU	0.01
Time Distributed Dense	1	-	Sigmoid	0.01
Flatten	-	-	-	-
Trainable parameters:	320,623			

structure of audio events. Next, we apply time distributed dense layers to reduce feature-length dimensionality. Note that the time resolution of T frames is maintained in both the GRU and dense layers. A sigmoid activation is used in the last time-distributed dense layer to produce a binary prediction of whether there is an event present in each time frame. This prediction layer outputs the strong labels for a recording. The dimensions of each prediction are $T \times 1$. Finally, we calculate the loss on this output as explained in the following section.

5.3.2 Loss Function for an MIL setting

As described in Section 2.5, in a multi instance learning setting, training based on the bag (i.e. the whole recording) prediction deriving from the max prediction of the instances is the most commonly used way. However, the max predicted instance is typically the most easy to be predicted as positive for a positive bag, while it is the most difficult to be predicted as negative for a negative bag. It seems that this sets a low burden on producing a positive output but a strong burden on producing a negative output. As indicated in Amar et al. (2001), the value of a bag is fully determined by its instance with the maximal output, regardless of how many real positive or negative instances are in the bag. The burden on producing a positive or negative output is not unbalanced at bag-level. However, on an instance-level, when using max to compute the loss, only one instance per bag contributes to the backpropagated gradient, which may lead to inefficient training. Additionally, as mentioned earlier, in positive bags a predictor only has to accurately predict the label for the

easiest positive instance to reach a perfect accuracy, thus not paying as much attention to the rest of the positive instances that might be harder to accurately detect.

Using all instances in a bag for computation of the loss and back-propagated gradient is important, since the network ideally should acquire some knowledge from every instance in every training iteration. However, it is hard to find an elegant theoretical interpretation of the characteristics of the instances in a bag. In Zhang et al. (2006), the authors proposed the "noisy-or" pooling function for MIL tasks, with the output label for bag B_i defined as:

$$O_i = 1 - \prod_{1 \leq j \leq M_j} (1 - o_{ij}) \tag{5.1}$$

where o_{ij} represents the output of the network for input B_{ij}, the j-th instance in B_i, the i-th bag of training instances. The noisy-or pooling function treats the predicted o_ij as the probability of the j-th instance of the i-th bag being positive.

In Liu et al. (2017) the authors applied this pooling in a deep learning setting instead of using the max prediction. However, noisy-or has been proven to not perform as well as max for audio event detection. As discussed in Wang et al. (2018), a significant problem with noisy-or is that the label of a bag is computed via the product of the instance labels as seen in equation (5.1). This calculation relies heavily on the assumed conditional independence of instance labels, an assumption which is highly inaccurate in audio event detection. Furthermore, this can lead

the system to believe a bag is positive even though all its instances are negative. Hence there is need for improved techniques to train machine learning for MIL scenarios.

We introduce two additional assumptions about bag and instance characteristics in order to compute a loss function that takes into account all of the instance predictions. One assumption is to consider the mean of the instance predictions of a bag. If a bag is negative, the mean should be zero, while if it is positive it should be greater than zero. The true mean is unknown in weakly labelled data. A naïve assumption is to presume that approximately half of the time a specific event will be present in a recording. Even though this is not true all of the time, it takes into consideration the predictions for all instances, and also inserts a bias to the loss that will keep producing gradient for training even after the max term has reached its perfect accuracy. However, this is indeed a naïve assumption that will guide the network to predict a balanced amount of positives and negatives that may make it more sensitive to all kinds of audio events, even when they are not the ones in question.

The second simple yet typically accurate assumption is that on both negative and positive recordings the minimum predictions at an instance-level should be zero. It is possible for a positive recording to have no negative frames; however, it is extremely rare in practice. This assumption could be used in synergy with max and mean to enforce the prediction of negative instances even on positive recordings and manage a certain level of the bias that is introduced with considering mean in the computation of the loss.

We train a network on a loss function that takes into account all the above-mentioned assumptions and compute the max, mean and min from the predictions of a recording and depending on whether a recording is positive or negative we predict their divergence from different conditions.

Our proposed loss function is computed as:

$$Loss = \frac{1}{3}\big(L(max_j(o_{ij}), Y_i) + L(mean_j(o_{ij}), \frac{Y_i}{2}) + L(min_j(o_{ij}), 0)\big), \quad (5.2)$$

where $L(\cdot, \cdot)$ is a function that computes the divergence between two vectors (for our experiments we used binary cross-entropy), o_{ij} are all the predicted instance labels of bag B_i, where $j = 1...M_i$ with M_i being the total number of instances in a bag, and Y_i is the ground truth label of the bag.

We refer to this as an MIL setting using a max, mean and min (MMM) loss. For negative recordings, Equation (5.2) will compute the divergence between the max, mean and min of the predictions of the instances of a bag B_i and zero. This denotes that the predictions for all instances of a negative recording should be zero. On the other hand, for positive recordings, the predictions should span the full dynamic range from zero to one, biased towards a similar amount of positive and negative instances. Our proposed loss function is designed to balance the positive and negative predictions in a bag resulting in a network that has the flexibility of learning from harder-to-predict positive instances even after many epochs. This is due to the fact that there are no obvious local

minima to get stuck in as in the max case. We test and compare the results of using different MIL loss functions for two low-resource datasets in Section 5.6.

Following our introduction of MMM in Morfi and Stowell (2018a), this loss function has been adopted by other researchers for weak-to-strong prediction in a multi-class setting during the Detection and Classification of Acoustic Scenes and Event (DCASE) 2018 challenge (Cances et al., 2018). The authors successfully used the MMM loss in this setting by adapting it to a multi-class input.

5.3.3 Half and Half Training

In the MIL setting for weak-to-strong labelling, it is of great importance to have a good balance between positive and negative bags, in order for the network to be able to distinguish what can be considered a positive instance and what can be considered a negative one. A simple approach to achieve this kind of balanced training is to have balanced training batches as input to the network. In our approach, we implement this by duplicating negative or positive recordings randomly during training depending on which ones are fewer in the whole dataset. Thus, each batch during training will consist of the same amount of positive and negative recordings. We call this kind of input Half and Half (HnH). Please note that balanced data for the WHEN task is not necessarily balanced data for the WHO task, an issue that we will return to in Section 5.6.

5.4 WHO: Audio Tagging

The second intermediate task of our approach is the WHO network that performs audio tagging using the provided weak labels of a low-resource dataset. This task follows supervised training since the weak labels provided are the ones that the network will try to learn how to predict. Hence, the training techniques that we use for the WHO network follow standard approaches.

5.4.1 Neural Network Architecture

A similar network architecture to the one proposed for WHEN (see Table 5.1) is used for the first few layers of WHO in order to implement our proposed training approaches that we will introduce in Section 5.5. Table 5.2 describes the structure of each individual layer used in the WHO network.

Similar to the WHEN network, the log mel-band energy feature extracted from the audio is used as input with shape $T \times M$, where T is the number of time frames in a recording and M in the number of features per time instance. The convolutional layers in the beginning of the network are in charge of learning the local shift-invariant features of this input. We use a 3×3 receptive field and the padding arguments set as 'same'. Max-pooling is performed along the frequency axis after every convolutional layer to reduce the dimension for the feature matrix. Global average pooling on both the time and frequency domain is finally

Table 5.2: WHO network architecture. Size refers to either kernel shape or number of units. #Fmaps is the number of feature maps in the layer. Activation denotes the activation used for the layer and l2_reg the amount of l2 kernel regularisation used in the layer.

Layer	**Size**	**#Fmaps**	**Activation**	**l2_reg**
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Max Pooling	1×5	-	-	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Max Pooling	1×4	-	-	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Convolution 2D	3×3	64	Linear	0.001
Batch Normalisation	-	-	-	-
Activation	-	-	ReLU	-
Max Pooling	1×2	-	-	-
Global Average Pooling 2D	-	-	-	-
Dense	#labels	-	Sigmoid	0.001
Trainable parameters:	191,319			

applied to the output of the convolutional part of the network and the results are fed to a dense layer that has units equal to the number of labels for our tagging task with sigmoid activation that predict the probability of each class being present in a recording. Finally, we calculate the binary cross-entropy loss on this output and the ground truth extracted from the weak labels. Binary cross-entropy loss is used instead of categorical cross-entropy since the recordings can contain more than one class present, making this a multi-label task.

5.5 Training Methods

We investigate three different methods to train the two intermediate tasks. One is the simple and usual approach of training each network separately for each task. Additionally, two multi-task learning (MTL) methods were tested, which we describe in the following sections. All three different methods have advantages and disadvantages that we will compare in detail in this section.

MTL (Caruana, 1997) aims to improve the performance of multiple learning tasks by sharing useful information among them. MTL can be very useful when using low-resource datasets since it can exploit useful information from other learning tasks to help alleviate the issue of limited data, based on the assumption that the multiple tasks are related. MTL is similar to transfer learning (Pan and Yang, 2010) which also transfers knowledge from one task to another. However, the focus of transfer learning is to help a single target task by initially training on one or

multiple tasks while MTL uses multiple tasks to help each other. A more detailed description of MTL and the reasons why we consider it a suitable way of training for low-resource data scenarios can be found in Section 2.7.

5.5.1 Separate Training

First, we use separate training for the two tasks. As depicted in Figure 5.5, two independent networks are defined, namely WHEN and WHO with the architectures described in Sections 5.3 and 5.4, respectively. The WHEN network performs audio event detection considering all labels as a single general label, while the WHO network performs audio tagging. Different kinds of input can be used for each network. HnH input is used for WHEN and the conventional (nonHnH) input for WHO. Thus, the batches used as input for the WHO network are randomly generated without taking into account the balance of positive and negative recordings in them. Different types of input are used for each task since their performance varies greatly depending on the type of input, even though the source of training data for each one is the same.

The advantage of separate training is that each network can train with the type of input that works better for it. WHEN uses a balanced batch of positive and negative recordings (HnH) for input while WHO uses the conventional randomly generated type of batch (nonHnH). The main disadvantage of separate training is that each task trains independently of the other, which may mean wasted computation, since these two tasks

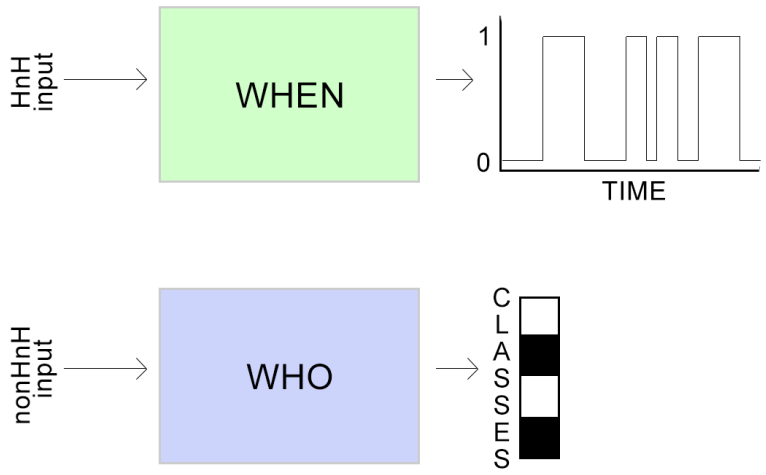


Figure 5.5: Separate training. Networks WHEN and WHO are defined and trained independently of one another for different tasks and with different types of inputs.

are somewhat related, hence they should be able to focus the attention of the network to important features and also regularise each other.

5.5.2 Joint Training

Joint training is one of the most common MTL approaches. In joint training, the same network is trained for more than one task. Usually, the network consists of a few shared layers in the beginning (most commonly convolutional to extract meaningful features from the input) followed by task specific layers before the predictions for each task. For each task, a separate loss is computed and then combined into the general loss of the network, as a weighted sum of the loss values. Joint training is a hard parameter sharing approach, since all tasks are forced to share the same early layers and weights. Figure 5.6 depicts how our intermediate tasks are adapted to the joint training approach. The *shared convolutional part*

consists of the common convolutional and max pooling layers, while the separate branches of the network consist of the task specific layers for WHEN and WHO as described in Tables 5.1 and 5.2, respectively.

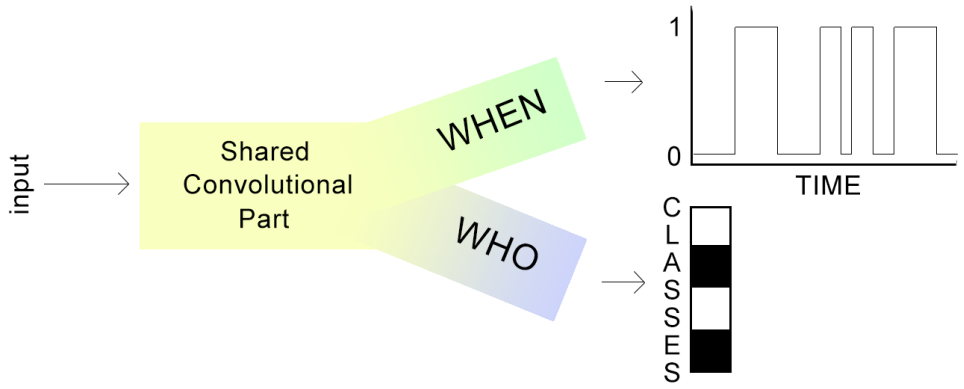


Figure 5.6: Joint training. A single network is defined for both tasks of audio event detection and audio tagging. The network consists of early shared convolutional layers between the tasks and separate task specific layers that produce the predictions. A single input type and two task specific loss functions are used while training.

The advantages of joint training are all the advantages presented by MTL. More specifically, information is shared between the tasks to help alleviate the issue of limited data. The model focuses its attention on features that are more relevant to all tasks. In addition, it reduces the risk of overfitting, since one task can act as the other's regulariser. One of the disadvantages of joint training is that both tasks train on the same input data batches, which, depending on the type (HnH or nonHnH), degrades the performance of one of the tasks (WHO or WHEN, respectively), as we will show in Section 5.6.

5.5.3 Tied Weights Training

In order to achieve the advantages of both separate and joint training without any of their disadvantages, we propose a new approach to MTL. Tied weights training follows the hard parameter training convention, where layers and their weights are shared between tasks. However, in contrast to joint training, different types of input can be used to train each task. Figure 5.7 depicts the structure of tied weights training. *Shared convolutional part* refers to the common convolutional and max pooling layers of WHEN and WHO, and the weights between the two tasks are constrained to be identical in these layers. Each task is trained consecutively for one epoch, computing a separate loss value and updating the weights of the shared layers. Using this approach, one can train each network with independent types of input as in separate training while keeping all the advantages of MTL learning. In contrast to joint training the loss is not computed as an overall weighted sum of the individual task losses, but each epoch is trained based on a different task loss specific to the input type of that epoch, hence in our setting we alternate between the two loss functions, the MIL MMM loss and a binary cross-entropy loss for multi class classification.

The results of tests conducted on all three training approaches for two low-resource data scenarios, one for birdsong and another one for mammal vocalisations, are presented in the following section.

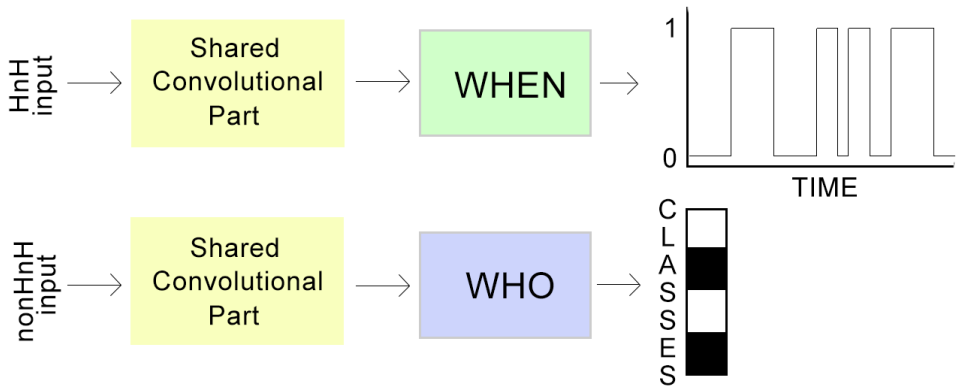


Figure 5.7: Tied weights training. A network defined per task. The weights of the initial convolutional layers are shared between the tasks. Different input is used for training each task.

5.6 Evaluation

In order to test our approach in a low-resource scenario we use two different datasets. The first one is Neural Information Processing Scaled for Bioacoustics (NIPS4B), a birdsong dataset described in Chapter 4, and the second one is a subset of one of the datasets used during the 2018 challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) for the task of large-scale weakly labelled semi-supervised sound event detection in domestic environments (task 4).[1] For the first one we trained networks that perform birdsong event detection and classification, while for the second one we focused our attention to mammal vocalisations including sound events produced by humans, dogs and cats.

For the NIPS4B dataset, along with the reference recordings, we use the weak labels provided by the organisers during training, and the NIPS4Bplus strong annotations that we acquired for evaluation purposes

[1] `http://dcase.community/challenge2018/task-large-scale-weakly-labeled-semi-supervised-sound-event-detection`

only. The dataset contains a total of 87 classes, with each being active in only 7 to 20 recordings. Each recording has 0 to 6 classes active in it. Additionally, the total amount of training time is less than one hour. All these make this dataset low-resource. A more detailed description of the dataset and the annotations can be found in Chapter 4.

For our experiments, we split the NIPS4B training dataset into a training set and a testing set. During the NIPS4B bird song competition, only the weak labels for the training dataset were released, hence we could only use these recordings for our experiments and could not make any use of the testing dataset that consisted of more recordings. For our training set, the first 499 recordings of the NIPS4B training dataset were used, while the rest were included in our testing set, excluding 14 recordings for which confident strong annotations could not be attained. Those 14 recordings were added to our training set resulting to a grand total of 513 training recordings and 174 testing recordings. Out of the 513 training recordings a small subset of them were used during training for validation purposes only. More specifically, the validation set we defined consisted of 63 recordings (55 positive, 8 negative), with the rest 450 recordings (385 positive, 65 negative) used only for training the model.

For the DCASE task 4 data, we first acquired the labelled training set provided by the organisers. The set originally contained 1578 clips deriving from YouTube videos, out of which we could only download 1537 due to copyright laws for the UK. Each clip is 10 seconds in duration. Weak annotations were verified and cross-checked by the organisers for this data. The set contains a total of 10 different classes: speech, dog,

cat, alarm/bell/ringing, dishes, frying, blender, running water, vacuum cleaner, electric shaver/toothbrush. Each of the clips contains at least one of these classes, however for our experiments having negative recordings (recordings that do not contain any of the target events) is of great importance for the WHEN task, hence we focus on a specific subset of this classes. Namely, we combine three classes, speech, dog and cat, into one general class: mammals. The remaining seven classes are ignored and only recordings that contain any mammal vocalisation are marked as positive with the rest being labelled as negative. Overall, out of the 1537 recordings, 865 of them are positive containing at least one mammal vocalisation and 672 of them are negative. The reason behind choosing to group together the three mammal classes is that even though the vocalisations produced by each class are very distinct all three of them are produced through a vocal system hence share some characteristics that distinguish them from the other seven classes that are sounds produced by objects.

The focus of our research is low-resource data scenarios, hence we randomly selected a subset of the DCASE training set in order to form the task into a low-resource one. More specifically, we randomly sampled a total of 360 recordings totalling to an hour of training data. Out of these 360 recordings, 300 were used for training our model and 60 for validation only. For the 300 training recordings, 161 were positive for mammal vocalisations and 139 were negative, while for the 60 validation recordings, 39 were positive for mammal vocalisations and the remaining 21 were negative. For testing, we acquired the test set of the challenge

that originally contained 288 recordings, though once more we could only download 230 of these clips from YouTube due to UK laws. This test set was annotated with strong labels, with timestamps (obtained by human annotators) by the organisers of the challenge.

5.6.1 Training Setup

The same general training setup and parameters are used for all networks and for both datasets. Firstly, due to the nature of neural networks, the input matrices during training should be of the same size, however for the NIPS4B dataset some recordings have different sizes. Since the max length of most recordings is 5 s, we extend the length of the other recordings to this max value by looping them in time. All recordings of the DCASE task 4 dataset have the same length of 10 s hence no reshaping is needed. As input to all our networks, log mel-band energy is extracted from audio in 23 ms Hamming windows with 50% overlap. In order to do so, the `librosa` Python library is used. In total, 40 mel-bands are used in the 0–22,050 Hz range. For a given length audio input, the feature extraction produces a Tx40 matrix ($T = 432$ for NIPS4B recordings and $T = 864$ for DCASE recordings).

The same hyper-parameters are used for training both WHEN and WHO tasks for all three different approaches. Our batchsize is equal to eight recordings. We use the Adam optimiser (Kingma and Ba, 2015) with a learning rate scheduler that reduces the initial rate of 1e-5, for NIPS4B data, and 1e-4, for DCASE data, by half every 20 epochs until it

reaches a minimum rate of 1e-8. We compare the MIL max loss function with our proposed MMM loss function for the predictions of the WHEN network in the bird vocalisation task and find the latter to perform better for both datasets (see Section 5.6.2). For training the WHO task we use a binary cross-entropy loss for multi-class predictions.

For the WHEN task, in order to efficiently use the data provided in the NIPS4B and DCASE datasets, we consider all unique bird labels and mammal labels, respectively, as one general label 'bird' and 'mammal' and train an audio event detection network for each general class.

All networks were trained using a single GeForce GTX 1080 Ti. The framework used to implement them was Keras with Tensorflow backend. Training time varied between WHEN and WHO tasks from 3 minutes to 3 seconds per epoch, respectively.

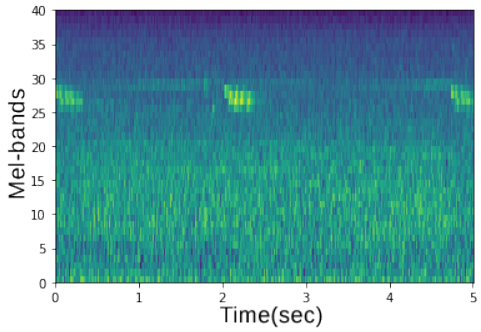
5.6.2 Results

First, we trained WHEN and WHO independently. WHEN was trained with a HnH input, since not using HnH could cause the network to ignore negative recordings. The MIL max loss and MMM loss were both used and compared, as well as false strong labelling. Additional to these loss functions, we trained two more networks using only the max and mean terms and the max and min terms of MMM to compare the impact each term had in the predictions of our audio event detector.

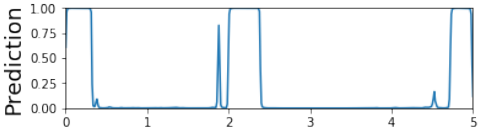
The first interesting aspect one can study is the individual results of false strong labelling, the conventional max loss and our proposed MMM

loss. Figure 5.8 depicts a positive recording from our NIPS4B testing set and the transcriptions predicted by each method. The first thing one can notice is that using false strong labelling has a tendency of pushing all the results closer to one when dealing with a positive recording. Some structure is apparent in the predicted labels, hence the network is indeed able to differentiate between positive and negative instances to some degree, however all results for positive recordings are above the usual 0.5 threshold. We attribute this primarily to the nature of the false strong labels: for a positive recording all time frames are labelled as positive. In this example, the network trained on the max loss correctly predicts the three more prominent events and then ignores all other events between them. However, the network trained with the MMM loss is starting to pick out some of the harder to detect events, due to the gradient provided by using mean. It is evident from this example that once the max loss reaches the perfect accuracy for a bag, it ignores the harder-to-predict events.

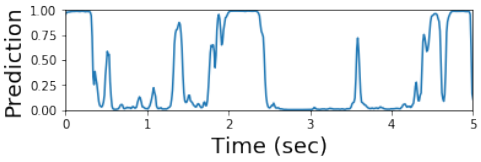
Figures 5.9a and 5.9b present the progress of the F-measure value, i.e. the harmonic average of the precision and recall of the predictions, on the NIPS4B and DCASE testing sets respectively, during training. One can notice that methods using the mean term in the loss prediction tend to reach higher scores overall. For the NIPS4B dataset, we notice that after training for a certain amount of epochs the results for most methods are decreasing: this is due to the common issue of the MIL setting which is the lack of any clear criterion for when one should stop training. No F-measure values are provided for false strong labelling since



(a) MIL using FSL

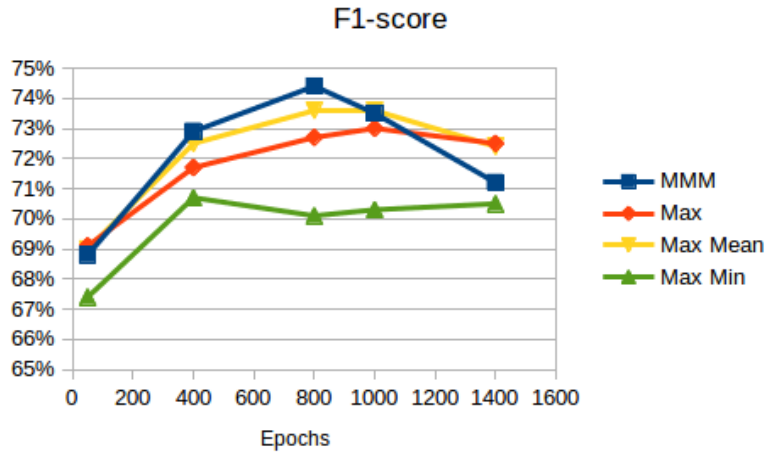


(b) MIL using max

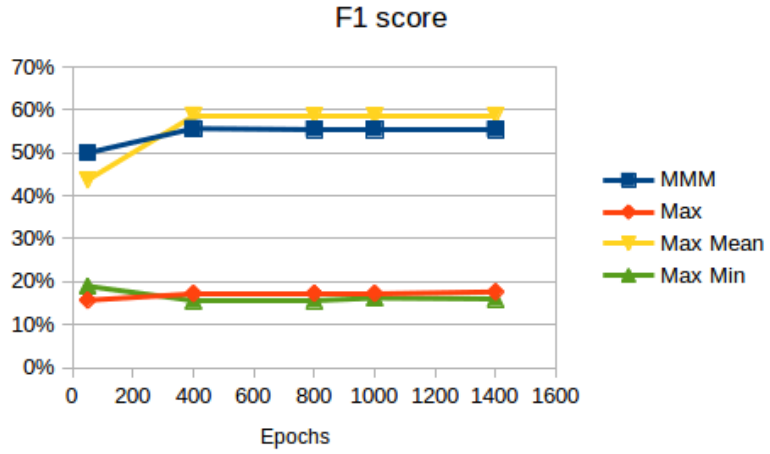


(c) MIL using MMM

Figure 5.8: Predicted transcription of a recording from the testing set on the NIPS4B dataset. 5.8a depicts the results of our WHEN network trained in a false strong labelling setting. 5.8b depicts the results of it trained with max loss. 5.8c depicts the results of it trained with MMM loss.



(a) F1 score of WHEN network predictions trained on the NIPS4Bplus data computed throughout training for different loss functions.



(b) F1 score of WHEN network predictions trained on the DCASE data computed throughout training for different loss functions.

Figure 5.9: Comparison of the progress of F1 score for our testing sets (a)NIPS4Bplus and (b)DCASE, through epochs for different loss functions, max mean min (MMM), max, max mean and max min.

all the predictions were always above the 0.5 threshold as depicted in the example of Figure 5.8, thus we do not consider false strong labelling any further.

As we showed, using the MMM loss function for training the WHEN task has advantages over all the previously proposed MIL loss functions, hence for all following experiments the results are acquired when training the WHEN task with MMM loss.

For the WHO task we trained with a conventional nonHnH input since using HnH for WHO made its performance worse, especially for the NIPS4Bplus data. This is due to the fact that the active classes are already very sparse (0 to 6 active classes out of 87 per recording) and, for both datasets, the HnH input duplicates negative recordings, hence decreasing the activation rate for each class, making it even harder to train reliably. Furthermore, for the DCASE dataset, we found it hard to train a network that would not overfit. We concluded that this was due to the size of the network and the relatively easy task of classifying between 3 classes, compared to the NIPS4Bplus task. For both datasets, we used a binary cross-entropy loss for multi-class predictions. This setup for the WHO task and the above MMM setup for WHEN were used for the separate training.

Next, we trained two versions of the joint network: one of them used a HnH input while the other a nonHnH input. In general, joint training did not provide any advantages for the NIPS4B data, but it improved the predictions for the WHO task on the DCASE data. As we mentioned above, while training for the WHO task on the DCASE data, we found it

hard not to overfit, yet it appears that joint training on both WHO and WHEN tasks acts as a regulariser. This was not the case with NIPS4B data, where the number of class is much higher and their activations are more sparse. When using HnH input while training on the NIPS4B data, the WHO predictions tended to not have a satisfactory performance due to the increase in negative recordings. When training the joint network with the nonHnH input, the WHEN task performance was degraded for both datasets. The loss value of the WHO task tended to be an order smaller than the one for WHEN, hence we trained with two different combinations of weights for the tasks. For one of them, both task losses had the same weight of 0.5, while for the other one the weight for the WHO task loss was an order larger than the WHEN; more specifically, we used weight 0.5 for WHEN loss and 5.0 for WHO loss.

Finally, we performed a tied weights training. This solved the issue of using only one type of input since it could train with both HnH and nonHnH inputs separately for each task as if the tasks were trained independently, while still sharing the weights of the shared layers like the joint training. Hence, we used the HnH input for training the WHEN task and a nonHnH input for training the WHO task.

Table 5.3 shows the area under the ROC curve (AUC) results for each training approach for the NIPS4B data. We can see that even though the tied weights training had a better overall performance compared to the joint training, separate training still had the best overall results. The best overall results for joint training were produced when using weights 0.5 and 5.0 for WHEN and WHO loss, respectively and also using nonHnH input.

Table 5.3: Area under the ROC curve (AUC) for the predictions of all training approaches on NIPS4B data. [WHEN: xx; WHO: yy] indicate the weights xx for WHEN task loss and yy for WHO task loss that were used during joint training. Best values are marked in bold.

Training Method	**Input Type WHEN — WHO**	**WHEN AUC**	**WHO AUC**
Separate	HnH — nonHnH	**0.90**	**0.94**
Joint [WHEN: 0.5; WHO: 0.5]	HnH	0.89	0.52
Joint [WHEN: 0.5; WHO: 0.5]	nonHnH	0.47	0.57
Joint [WHEN: 0.5; WHO: 5.0]	HnH	**0.90**	0.50
Joint [WHEN: 0.5; WHO: 5.0]	nonHnH	0.82	0.75
Tied Weights	HnH — nonHnH	0.87	0.77

Table 5.4: Area under the ROC curve (AUC) for the predictions of all training approaches on DCASE data. [WHEN: xx; WHO: yy] indicate the weights xx for WHEN task loss and yy for WHO task loss that were used during joint training. Best values are marked in bold.

Training Method	**Input Type WHEN — WHO**	**WHEN AUC**	**WHO AUC**
Separate	HnH — nonHnH	**0.85**	0.78
Joint [WHEN: 0.5; WHO: 0.5]	HnH	0.83	0.83
Joint [WHEN: 0.5; WHO: 0.5]	nonHnH	0.80	0.86
Joint [WHEN: 0.5; WHO: 5.0]	HnH	0.82	0.85
Joint [WHEN: 0.5; WHO: 5.0]	nonHnH	0.76	0.82
Tied Weights	HnH — nonHnH	0.84	**0.89**

Hence, we can conclude that the WHO network was sharing important information with the WHEN network that boosted its performance when enough weight was given to its loss.

Table 5.4 shows the area under the ROC curve (AUC) results for each training approach for the DCASE data. One can notice that the results for the WHO task increased when we used joint or tied weights training. This can be attributed to the fact that the information from the WHEN task is acting as a regulariser for the WHO task. On the other hand, the

performance of the WHEN task dropped, especially when using nonHnH input as we originally expected.

As mentioned before, any type of MTL training had so far been proven to outperform independent training, which was not the case in some of our experiments, especially concerning the NIPS4B data. This could be attributed to some low-resource aspects of the dataset. In contrast to the DCASE data, where MTL training produced improved results, the NIPS4B dataset contains a much larger number of classes with most of them being very sparsely active. Using MTL training introduces an additional regulariser to an already hard to train for task. For this kind of scenario a soft parameter sharing approach (Duong et al., 2015; Misra et al., 2016; Yang and Hospedales, 2017; Ruder et al., 2017) may be more suitable since the effect each task has on each other is less strong.

Chapter 6

Conclusions and Further Work

In fulfilment of our aim to achieve automatic wildlife monitoring of songbirds in low-resource scenarios, the central part of this thesis has been the development of machine learning methods to apply towards this task. We implemented and evaluated multiple methods on two main approaches: event-based segmentation and classification; and deep learning in a multi instance learning setting for whole recordings.

To conclude this thesis, we first summarise the contributions made. We then consider some potential avenues for future work and finally reflect on the results of our research.

6.1 Summary of Thesis Contributions

- In Chapter 3, we developed and evaluated a two step process of segmentation and classification of audio events. Both steps are based on image processing methods. We evaluated their performance for synthetic and natural data and acquired an insight into their limitations that further motivated the use of deep learning methods.

- In Chapter 4, we introduced the first open access ecological audio dataset that contains bird species tags and temporal annotations, NIPS4Bplus. We also presented useful statistics about the number of occurrences of each label, the distribution of active labels through the dataset and more. Finally, we described in detail the temporal annotation process. NIPS4Bplus is a low-resource ecological dataset and annotations that can be used for either training supervised automated methods that perform bird vocalisation detection and classification or be used for evaluating methods that use only audio tags or no annotations for training.

- In Chapter 5, we developed a new loss function (MMM) for deep learning in a multi-instance learning (MIL) setting for audio event detection. We compared the performance of MMM to the traditional max MIL loss function, and results showed that using MMM can detect harder to find vocalisations that max cannot.

- In Chapter 5, we also proposed and evaluated a task factorisation of the full transcription task that can be used in a low-resource MIL setting. We explored multiple ways for training the factorised

tasks and gained some insight into multi task learning. We also developed a method, namely tied weights training, that has all the advantages of separate training along with the positive aspects of joint training in MTL.

Many of these contributions are represented in international peer-reviewed conference and journal articles, as listed in Section 1.5.

6.2 Future Work

Further work that could follow on from the research of this thesis includes:

Full transcription: In this thesis we developed methods that can predict what we refer to as intermediate transcription. A way to combine this intermediate transcription with full transcription prediction methods should be investigated in future work.

Improved MIL: One of the still unsolved issues in MIL is the lack any specific criterion to stop training. Using terms that include all instances in predicting the loss to be minimised may prove to be the way to define a way of early stopping. Furthermore, future work should include investigating a way to define a more appropriate and flexible mean term in the loss function instead of the naïve 0.5 we used during our experiments.

Use in other contexts: Our experiments focused on low-resource scenarios. While the need for methods that can perform well when

there is a limited amount of data and annotations is great, the methods could also be potentially applied to boost the performance of predictors in a large scale setting.

6.3 Closing Remarks

During our research for automated wildlife monitoring the main recurring issue was the lack of strongly annotated data. These type of data are preferred when training full transcription predictors and are necessary in order to evaluate any such model. Yet in order to acquire such data human annotators are hard to find, may be expensive or slow in their progress, so we try to train unsupervised or weakly supervised models alleviate the need of human annotators. Yet we need some annotated data to evaluate these models.

This issue is a *vicious cycle*. How does one get out of such a cycle? With difficulty. Such problems are difficult to solve; since they're cyclical, one does not know where to start from.

It is hard to decide if it is better to spend time acquiring annotated data by manual labour or to spend time trying to adjust a model to a low-resource scenario. With the unlimited amount of different possible audio events there will never be enough annotated data for each individual possible audio class. Furthermore, the amount of unlabelled recordings keeps on increasing fast, so that no amount of manual work will ever be able to annotate all this data. Hence, low-resource scenarios and the

time and effort it takes to adjust models to them are a necessary evil.

The future of audio event transcription is open for methods that can learn to adapt to training with a very small amount of annotated data. These can either be methods that manage to take advantage of the huge amount of unlabelled data and transfer that knowledge to the low-resource data or methods that find a way to train and make an efficient use of small amounts of data like the ones we explored in this thesis. And we have shown, even methods such as deep neural networks, that were originally thought to only behave well when there is a vast amount of data, can train in low-resource scenarios when the correct training setup is in place.

Bibliography

Adavanne, S., Drossos, K., Çakir, E., and Virtanen, T. (2017). Stacked convolutional and recurrent neural networks for bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1729–1733.

Adavanne, S. and Virtanen, T. (2017). Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network. In *Detection and Classification of Acoustic Scenes and Events (DCASE 2017)*.

Aide, T. M., Corrada-Bravo, C., Campos-Cerqueira, M., Milan, C., Vega, G., and Alvarez, R. (2013). Real-time bioacoustics monitoring and automated species identification. *PeerJ*, 1:e103.

Amar, R., Dooly, D. R., Goldman, S. A., and Zhang, Q. (2001). Multiple-instance learning of real-valued data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, '01, pages 3–10, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Andrén, H. (1994). Effects of habitat fragmentation on birds and mam-

mals in landscapes with different proportions of suitable habitat: A review. *Oikos*, 71(3):355–366.

Anguera, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G., and Vinyals, O. (2012). Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370.

Armitage, D. W. and Ober, H. K. (2010). A comparison of supervised learning techniques in the classification of bat echolocation calls. *Ecological Informatics*, 5:465–473.

Bardeli, R., Wolff, D., Kurth, F., Koch, M., Tauchert, K.-H., and Frommolt, K.-H. (2010). Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring. *Pattern Recognition Letters*, 31:1524–1534.

Baxter, J. (1997). A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Blum, A. L. and Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117 – 127.

Briggs, F., Lakshminarayanan, B., Neal, L., Fern, X., Raich, R., Hadley, S. J. K., Hadley, A. S., and Betts, M. G. (2012). Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach. *Journal of the Acoustic Society of America*, 131:4640–4650.

Buades, A., Coll, B., and Morel, J. M. (2004). On image denoising methods. Technical report, Centre de Mathematiques et de Leurs Applications (CMLA).

Buades, A., Coll, B., and Morel, J.-M. (2011). Non-local means denoising. *Image Processing On Line*, 1:208–212.

Cakir, E., Adavanne, S., Parascandolo, G., Drossos, K., and Virtanen, T. (2017). Convolutional recurrent neural networks for bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1744–1748.

Campos Cerqueira, M. and Aide, T. M. (2016). Improving distribution data of threatened species by combining acoustic monitoring and occupancy modeling. *Methods in Ecology and Evolution*, 7:1340–1348.

Cances, L., Pellegrini, T., and Guyot, P. (2018). Sound event detection from weak annotations: Weighted GRU versus multi-instance learning. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE 2018)*.

Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 41–48. Morgan Kaufmann.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

Chambolle, A. (2004). An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97.

Chapin, F. S., Zavaleta, E. S., Eviner, V. T., Naylor, R. L., Vitousek, P. M., Reynolds, H. L., Hooper, D. U., Lavorel, S., Sala, O., Hobbie, S. E., Mack, M. C., and Díaz, S. (2000). Consequences of changing biodiversity. *Nature*, 405(6783):234–242.

Cho, K., van Merrinboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Choi, K., Fazekas, G., and Sandler, M. B. (2016). Automatic tagging using deep convolutional neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*.

Choromanska, A., Henaff, M., Mathieu, M., Ben Arous, G., and LeCun, Y. (2015). The loss surfaces of multilayer networks. *Journal of Machine Learning Research*, 38:192–204.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In

Proceedings of the 25th International Conference on Machine Learning (ICML), '08, pages 160–167, New York, NY, USA. ACM.

Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8609–8613.

Damoulas, T., Henry, S., Farnsworth, A., Lanzone, M., and Gomes, C. P. (2010). Bayesian classification of flight calls with a novel dynamic time warping kernel. In *The 9th International Conference on Machine Learning and Applications (ICMLA)*, pages 424–429. IEEE Computer Society.

Dawson, D. K. and Efford, M. G. (2009). Bird population density estimated from acoustic signals. *Journal of Applied Ecology*, 46(6):1201–1209.

de Camargo, U. M., Somervuo, P., and Ovaskainen, O. (2017). Protaxsound: A probabilistic framework for automated animal sound identification. *PLOS ONE*, 12(9):1–15.

Deng, L., Hinton, G., and Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8599–8603.

Dieleman, S. and Schrauwen, B. (2014). End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71.

Drake, K. L., Frey, M., Hogan, D., and Hedley, R. (2016). Using digital recordings and sonogram analysis to obtain counts of yellow rails. *Wildlife Society Bulletin*, 40(2):346–354.

Duong, L., Cohn, T., Bird, S., and Cook, P. (2015). Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 845–850. Association for Computational Linguistics (ACL).

Duran, J., Coll, B., and Sbert, C. (2013). Chambolle's Projection Algorithm for Total Variation Denoising. *Image Processing On Line*, 3:311–331.

Eronen, A. J., Peltonen, V. T., Tuomi, J. T., Klapuri, A. P., Fagerlund, S., Sorsa, T., Lorho, G., and Huopaniemi, J. (2006). Audio-based context recognition. *Transactions on Audio, Speech and Language Processing*, 14(1):321–329.

Fanioudakis, L. and Potamitis, I. (2017). Deep networks tag the location of bird vocalisations on audio spectrograms. arXiv:1711.04347.

Farley, B. and Clark, W. (1954). Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, 4(4):76–84.

Fearn, E., Redford, K. H., Woods, W., and Wildlife Conservation Society (New York, N.Y.) (2008). *State of the wild, 2008-2009 : a global portrait of wildlife, wildlands, and oceans*. Island Press.

Fearn, E., Redford, K. H., Woods, W., and Wildlife Conservation Society (New York, N.Y.) (2010). *State of the wild 2010-2011: a global portrait*. Island Press.

Ferraz, G., Marinelli, C. E., and Lovejoy, T. E. (2008). Biological monitoring in the Amazon: Recent progress and future needs. *Biotropica*, 40(1):7–10.

Fischer, F. P., Schulz, U., Schubert, H., Knapp, P., and Schmger, M. (1997). Quantitative assessment of grassland quality: Acoustic determination of population sizes of orthopteran indicator species. *Ecological Applications*, 7(3):909–920.

Fodor, G. (2013). The Ninth Annual MLSP competition: First place. *International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–2.

Frommolt, K.-H. (2017). Information obtained from long-term acoustic recordings: applying bioacoustic techniques for monitoring wetland birds during breeding season. *Journal of Ornithology*, 158:659–668.

Furnas, B. J. and Callas, R. L. (2015). Using automated recorders and occupancy models to monitor common forest birds across a large geographic region. *The Journal of Wildlife Management*, 79(2):325–337.

Garcia-Romero, D., Snyder, D., Sell, G., Povey, D., and McCree, A.

(2017). Speaker diarization using deep neural network embeddings. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4930–4934.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Girshick, R. (2015). Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS). Society for Artificial Intelligence and Statistics*.

Glotin, H., LeCun, Y., Artières, T., Mallat, S., Tchernichovski, O., and Halkias, X. (2013). Proc. neural information processing scaled for bioacoustics, from neurons to big data. USA. NIPS Int. Conf.

Goëau, H., Glotin, H., Vellinga, W.-P., Planqué, R., and Joly, A. (2016). LifeCLEF Bird Identification Task 2016: The arrival of Deep Learning.

Goëau, H., Glotin, H., Vellinga, W.-P., Planqué, R., and Joly, A. (2017). LifeCLEF Bird Identification Task 2017.

Goëau, H., Glotin, H., Vellinga, W.-P., Planqué, R., Rauber, A., and Joly, A. (2015). LifeCLEF Bird Identification Task 2015.

Goetze, S., Schroder, J., Gerlach, S., Hollosi, D., Appell, J.-E., and

Wallhoff, F. (2012). Acoustic Monitoring and Localization for Social Care. *Journal of Computing Science and Engineering*, 6(1):40–50.

Gonzalez, R. C. and Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Grill, T. and Schlüter, J. (2017). Two convolutional neural networks for bird detection in audio signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1764–1768.

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K. (2017). Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Huang, G., Liu, Z., and Weinberger, K. Q. (2017). Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1(4):364–378.

Johnston, A., Newson, S., Risely, K., J Musgrove, A., Massimino, D., Baillie, S., and W Pearce-Higgins, J. (2014). Species traits explain variation in detectability of UK birds. *Bird Study*, 61:340–350.

Kamp, J., Oppel, S., Heldbjerg, H., Nyegaard, T., and F. Donald, P. (2016). Unstructured citizen science data fail to detect long-term population declines of common birds in Denmark. *Diversity and Distributions*, 22:1024–1035.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations (ICLR), San Diego*.

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In Shannon, C. and McCarthy, J., editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ.

Knight, E., C. Hannah, K., J. Foley, G., D. Scott, C., Brigham, R., and Bayne, E. (2017). Recommendations for acoustic recognizer performance assessment with application to five common automated signal recognition programs. *Avian Conservation and Ecology*, 12(2):14.

Kolen, J. and Kremer, S. (2001). *A Field Guide to Dynamical Recurrent Networks*. Wiley.

Kong, Q., Xu, Y., and Plumbley, M. D. (2017a). Joint detection and classification convolutional neural network on weakly labelled bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1749–1753.

Kong, Q., Xu, Y., Wang, W., and Plumbley, M. D. (2017b). A joint detection-classification model for audio tagging of weakly labelled data. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 641–645.

Kroodsma, D. (2005). *The Singing Life of Birds: Audio CD*. The Singing Life of Birds: The Art and Science of Listening to Birdsong. Houghton Mifflin.

Kumar, A. and Raj, B. (2016). Audio event detection using weakly labeled data. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 1038–1047, New York, NY, USA. ACM.

Kumar, A. and Raj, B. (2017). Deep CNN framework for audio event recognition using weakly labeled web data. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*.

Lakshminarayanan, B., Raich, R., and Fern, X. (2009). A syllable-level probabilistic framework for bird species identification. In *2009 8th International Conference on Machine Learning and Applications (ICMLA)*, pages 53–59.

Lambert, K. T. A. and McDonald, P. G. (2014). A low-cost, yet simple and highly repeatable system for acoustically surveying cryptic species. *Austral Ecology*, 39(7):779–785.

Lasseck, M. (2015). Towards automatic large-scale identification of birds in audio recordings. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF 2015, Toulouse, France, September 8-11, 2015, Proceedings*, pages 364–375.

LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). *Object Recognition with Gradient-Based Learning*, pages 319–345. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lee, C.-H., Han, C.-C., and Chuang, C.-C. (2008). Automatic classification of bird species from their sounds using two-dimensional cepstral coefficients. *Transactions Audio, Speech and Language Processing*, 16(8):1541–1550.

Liu, D., Zhou, Y., Sun, X., Zha, Z., and Zeng, W. (2017). Adaptive pooling in multi-instance learning for web video annotation. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 318–327.

Liu, J., Dietz, T., Carpenter, S. R., Alberti, M., Folke, C., Moran, E., Pell, A. N., Deadman, P., Kratz, T., Lubchenco, J., Ostrom, E., Ouyang, Z., Provencher, W., Redman, C. L., Schneider, S. H., and Taylor, W. W. (2007). Complexity of coupled human and natural systems. *Science*, 317(5844):1513–1516.

Lostanlen, V., Salamon, J., Farnsworth, A., Kelling, S., and Bello, J. P. (2018). Birdvox-full-night: a dataset and benchmark for avian flight call detection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 266–270.

Luther, D. (2008). Signaller: Receiver coordination and the timing of communication in amazonian birds. *Biology Letters*, 4:651–654.

Luther, D. and Wiley, R. (2009). Production and perception of communicatory signals in a noisy environment. *Biology Letters*, 5:183–187.

Mac Swiney Gonzlez, M. C., M. Clarke, F., and Racey, P. (2008). What you see is not what you get: The role of ultrasonic detectors in increas-

ing inventory completeness in neotropical bat assemblages. *Journal of Applied Ecology*, 45:1364–1371.

MacArthur, R. H., MacArthur, J. W., and Preer, J. (1962). On bird species diversity. II. prediction of bird census from habitat measurements. *The American Naturalist*, 96(888):167–174.

Marques, T. A., Thomas, L., Martin, S. W., Mellinger, D. K., Ward, J. A., Moretti, D. J., Harris, D., and Tyack, P. L. (2013). Estimating animal population density using passive acoustics. *Biological Reviews*, 88(2):287–309.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003.

Morfi, V., Bas, Y., Pamula, H., Glotin, H., and Stowell, D. (2018). NIPS4Bplus: a richly annotated birdsong audio dataset. *PeerJ CS (under review)*, abs/1811.02275.

Morfi, V. and Stowell, D. (2017). Deductive refinement of species labelling in weakly labelled birdsong recordings. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 656–660.

Morfi, V. and Stowell, D. (2018a). Data-efficient weakly supervised learning for low-resource audio event detection using deep learning. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE 2018)*, pages 123–127.

Morfi, V. and Stowell, D. (2018b). Deep learning for audio event detection and tagging on low-resource datasets. *Applied Sciences*, 8(8):1397.

Nandwana, M. K. and Hasan, T. (2016). Towards smart-cars that can listen: Abnormal acoustic event detection on the road. In *17th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 2968–2971, San Francisco, California, USA.

Pacifici, K., Simons, T. R., and Pollock, K. H. (2008). Effects of vegetation and background noise on the detection process in auditory avian point-count surveys. *The Auk*, 125(4):998–998.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *30th International Conference on Machine Learning (ICML)*, volume 28.

Pellegrini, T. (2017). Densely connected cnns for bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1734–1738.

Pijanowski, B. C., Villanueva-Rivera, L. J., Dumyahn, S. L., Farina, A., Krause, B. L., Napoletano, B. M., Gage, S. H., and Pieretti, N.

(2011). Soundscape ecology: The science of sound in the landscape. *BioScience*, 61(3):203–216.

Pons, J., Nieto, O., Prockup, M., Schmidt, E. M., Ehmann, A. F., and Serra, X. (2018). End-to-end learning for music audio tagging at scale. In *2018 19th International Society for Music Information Retrieval Conference (ISMIR)*.

Potamitis, I. (2015). Unsupervised dictionary extraction of bird vocalisations and new tools on assessing and visualising bird activity. *Ecological Informatics*, 26(3):6–17.

Prechelt, L. (2012). *Early Stopping — But When?*, pages 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ribeiro Jr, J., Sugai, L., and Campos Cerqueira, M. (2017). Passive acoustic monitoring as a complementary strategy to assess biodiversity in the Brazilian Amazonia. *Biodiversity and Conservation*, 26:2999–3002.

Rochester, N., Holland, J., Haibt, L., and Duda, W. (1956). Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions on Information Theory*, 2(3):80–93.

Roger, V., Bartcus, M., Chamroukhi, F., and Glotin, H. (2018). Unsupervised bioacoustic segmentation by hierarchical dirichlet process hidden markov model. In *Multimedia Tools and Applications for Environmental & Biodiversity Informatics*, pages 113–130. Springer.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2017). Sluice

networks: Learning what to share between loosely related tasks. abs/1705.08142.

Ruiz-Muñoz, J. F., Orozco-Alzate, M., and Castellanos-Dominguez, G. (2015). Multiple instance learning-based birdsong classification using unsupervised recording segmentation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2632–2638. AAAI Press.

Salamon, J. and Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24:279–283.

Schlüter, J. (2016). Learning to Pinpoint Singing Voice from Weakly Labeled Examples. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, USA.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117.

Scott Brandes, T. (2008). Automated sound recording and analysis techniques for bird surveys and conservation. *Bird Conservation International - BIRD CONSERV INT*, 18:S163–S173.

Sovern, S. G., Forsman, E. D., Olson, G. S., Biswell, B. L., Taylor, M., and Anthony, R. G. (2014). Barred owls and landscape attributes influence territory occupancy of northern spotted owls. *The Journal of Wildlife Management*, 78(8):1436–1443.

Stowell, D. and Clayton, D. (2015). Acoustic event detection for multiple overlapping similar sources. In *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5.

Stowell, D., Morfi, V., and Gill, L. F. (2016a). Individual identity in songbirds: signal representations and metric learning for locating the information in complex corvid calls. In *17th Annual Conference of the International Speech Communication Association (Interspeech)*, volume 2, pages 2607–2611, San Francisco, California, USA.

Stowell, D., Wood, M., Stylianou, Y., and Glotin, H. (2016b). Bird detection in audio: A survey and a challenge. *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6.

Thakur, A., andW. P. Rajan, R. J., and Dileep, A. D. (2017). Rapid bird activity detection using probabilistic sequence kernels. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1754–1758.

Tirumala, S. S. and Shahamiri, S. R. (2016). A review on deep learning approaches in speaker identification. In *Proceedings of the 8th International Conference on Signal Processing Systems*, ICSPS 2016, pages 142–147, New York, NY, USA. ACM.

Vitousek, P. M., Mooney, H. A., Lubchenco, J., and Melillo, J. M. (1997). Human domination of earth's ecosystems. *Science*, 277(5325):494–499.

Wang, Y., Li, J., and Metze, F. (2018). Comparing the max and noisy-or pooling functions in multiple instance learning for weakly supervised sequence learning tasks. In *19th Annual Conference of the Interna-*

tional Speech Communication Association (Interspeech), pages 1339–1343.

Werbos, P. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University.

Wrege, P., D. Rowland, E., Keen, S., and Shiu, Y. (2017). Acoustic monitoring for conservation in tropical forests: Examples from forest elephants. *Methods in Ecology and Evolution*, 8:1292–1301.

Xu, Y., Huang, Q., Wang, W., Foster, P., Sigtia, S., Jackson, P. J. B., and Plumbley, M. D. (2017a). Unsupervised feature learning based on deep models for environmental audio tagging. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1230–1241.

Xu, Y., Kong, Q., Huang, Q., Wang, W., and Plumbley, M. D. (2017b). Convolutional gated recurrent neural network incorporating spatial features for audio tagging. *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3461–3466.

Yang, Y. and Hospedales, T. (2017). Trace norm regularised deep multi-task learning. In *5th International Conference on Learning Representations Workshop (ICLR)*.

Yu, D. and Deng, L. (2016). *Automatic Speech Recognition*. Springer.

Zhang, C., Platt, J. C., and Viola, P. A. (2006). Multiple instance boosting for object detection. In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18*, pages 1417–1424. MIT Press.

Zhang, M. L. and Zhou, Z. H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.

Zhang, Y. and Yang, Q. (2018). An overview of multi-task learning. *National Science Review*, 5(1):30–43.

Zheng, T. F., Zhang, G., and Song, Z. (2001). Comparison of different implementations of mfcc. *Journal of Computer Science and Technology*, 16:582–589.

Zhou, Z.-H. and Zhang, M.-L. (2002). Neural networks for multi-instance learning. In *Proceedings of the International Conference on Intelligent Information Technology, Beijing, China*, pages 455–459.