Machine learning for acoustic mosquito detection

Ivan Kiskin

A thesis presented for the degree of

Doctor of Philosophy

Machine Learning Research Group

Supervised by Professor Steve Roberts

The University of Oxford

Hilary 2020

Machine learning for acoustic mosquito detection

Abstract

Mosquitoes are responsible for over one billion cases of disease and over one million deaths each year. The data produced during mosquito surveillance are needed to identify emerging insecticide resistance, facilitate effective and evidence-led insecticide intervention programmes as well as model current and future vector-borne disease transmission. Traditional mosquito survey methods are time-consuming, expensive, and spatially limited. Consequently, many mosquito distribution models that map the range of these insects rely on small quantities of poorly spatially distributed occurrence data. There is therefore an urgent need to develop new mosquito survey methods that can provide real-time species-specific occurrence and abundant data without human risk. Here we consider an acoustic detection paradigm, in which the distinctive 'buzz' of mosquito flight is used as a characteristic signature for detection and subsequent species identification. We show it is possible to achieve high classification accuracy even in data-scarce scenarios, using a combination of deep learning and wavelet encoding. Additionally, we develop and deploy a smartphone app that allows mosquito detection at scale and can discriminate between species with high accuracy. We garner low-resolution labelling for parts of our data via crowdsourcing, supplementing the high-resolution labels obtained from experts and publicly release a baseline model and dataset.

The technical materials that underpin this thesis detail development of machine learning approaches for detecting and identifying events in data, with the primary focus of finding mosquito flight tones in acoustic time series. Solutions specific to such audio detection are developed, tested and applied to field-gathered data. Although the research is specific to one focal application domain, the approaches developed are generic and were motivated by canonical problems of low signal-to-noise ratio, sparse data, multiple resolution labelling, class imbalance, and decision bias. They thus apply to a far wider set of detection problems, in audio time series and beyond.

Acknowledgements

I would like to begin by acknowledging that this work has been a by-product of the opportunities that have been given to me in life by my family, as well as a healthy dose of luck.

First, I would like to express my gratitude towards Professor Steve Roberts who has always found positivity in both my work (even when I could not) and in life. Steve has been proactively supporting me irrespective of the circumstances we have faced. Steve's selfless dedication to his students, evidenced from my time as an undergraduate under his supervision, ultimately led me to pursue a DPhil.

I thank all of my collaborators, including, but not limited to, Henry Chan, Udeepa Meepagama, Yunpeng Li, Lawrence Wang, and Davide Zilli. Additionally, I would like to thank the HumBug team for their work and vision, and wish them every success in further stages of the project.

I would like to express an enormous thanks to Adam Cobb for being invaluable to my personal and professional development, as well as an excellent collaborator no matter what obstacles we have encountered together. Adam has been faced with probably the toughest of challenges out there – a seat next to me for four years. It comes at no surprise that he is now stationed over 6,000 km away.

A very special thanks also extends to Ahsan Alvi, and Luca Laurenti, whose support in life have meant more to me than they can imagine. Their encouragement, desire to help, and healthy mindsets have helped me overcome countless hurdles, for which I will be forever grateful.

Last, but not least, I want to thank my girlfriend for re-igniting my enthusiasm, drive, and sense of purpose during my DPhil studies. In particular, I consider myself most fortunate to have spent the coronavirus lockdown in a safe and sound environment with her ever-present care by my side.

This work has been made possible through my sponsorship by the Autonomous Intelligent Machines and Systems (AIMS) Centre for Doctoral Training and the Engineering and Physical Sciences Research Council (Grant No. EP/L015897/1).

Contents

Declaration viii

List of Figures x

List of Tables xiv

Nomenclature xvi

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Problem statement . . . 2
- 1.3 Main contributions . . . 4
- 1.4 Structure . . . 5

2 Machine learning for signal detection 6
- 2.1 Overview . . . 6
- 2.2 Foundational principles . . . 7
  - 2.2.1 Time series . . . 7
  - 2.2.2 What is a signal? . . . 7
  - 2.2.3 Regression . . . 9
  - 2.2.4 Detection . . . 9
- 2.3 Uncertainty in machine learning . . . 10
  - 2.3.1 Measurement error . . . 11
  - 2.3.2 Incomplete domain coverage . . . 12
  - 2.3.3 Model error . . . 13
  - 2.3.4 Bayesian decision theory . . . 14
- 2.4 Introduction to model classes . . . 15
  - 2.4.1 Decision trees . . . 17

2.4.2 Random forests . . . 19
2.4.3 Support vector machines . . . 20
2.4.4 Neural networks . . . 23
2.4.4.1 Convolutional neural networks . . . 26
2.5 Audio data analysis . . . 29
2.6 Chapter summary . . . 34

3 The HumBug project 35
3.1 Overview . . . 35
3.2 Acoustic detection prior work . . . 35
3.3 Timeline . . . 37
3.3.1 A: Data collection . . . 39
3.3.2 B: Smartphone app . . . 40
3.3.3 C: Research output . . . 41
3.4 Chapter summary . . . 45

4 Bioacoustic classification with wavelet-conditioned neural networks 46
4.1 Overview . . . 46
4.2 Introduction . . . 47
4.3 Related work . . . 49
4.3.1 Applications . . . 50
4.3.2 Feature representation and learning . . . 51
4.4 Method . . . 53
4.4.1 The wavelet transform . . . 54
4.4.2 Neural network configurations . . . 57
4.4.3 Traditional classifier baseline . . . 57
4.5 Experimental details . . . 58
4.5.1 Datasets . . . 58
4.5.2 Cross-validated parameter search . . . 60
4.5.3 Computational details . . . 61
4.6 Classification performance . . . 63
4.6.1 Mosquito detection . . . 63
4.6.2 Bird classification . . . 64
4.7 Visualising discriminative power . . . 65
4.8 Chapter summary . . . 68

5 HumBug: data acquisition and smartphone applications 70
5.1 Overview . . . 70

5.2 Mosquito detection with low-cost smartphones: data acquisition for malaria research . . . 71
5.2.1 Mobile devices . . . 72
5.2.2 Data acquisition . . . 72
5.2.3 Classification pipeline . . . 73
5.2.4 Experiments and results . . . 74
5.3 Fast mosquito acoustic detection with field cup recordings: an initial investigation . . . 76
5.3.1 Field experiments and data summary . . . 76
5.3.2 Model . . . 77
5.3.3 Training strategy . . . 79
5.3.4 Experiment configuration . . . 80
5.3.5 Results . . . 81
5.3.6 Initial investigation discussion . . . 84
5.4 Cost-sensitive detection with variational autoencoders for acoustic sensing . . . 84
5.4.1 Method . . . 85
5.4.2 2ν-SVM . . . 86
5.4.3 Cost-sensitive ensemble selection . . . 87
5.4.4 Data . . . 88
5.4.5 Parameter values . . . 88
5.4.6 Results . . . 90
5.4.7 Scaling to a larger dataset . . . 91
5.5 HumBug Zooniverse: a crowdsourced acoustic mosquito dataset . . . 91
5.5.1 Related work . . . 92
5.5.2 Data acquisition . . . 93
5.5.3 Data labelling . . . 94
5.5.4 Baseline . . . 96
5.6 Chapter summary . . . 97

6 Loss-calibrated Bayesian neural networks for noisy acoustic mosquito detection 99
6.1 Overview . . . 99
6.2 Bayesian decision theory for Bayesian neural networks . . . 100
6.2.1 Probabilistic inference . . . 101
6.2.2 Decision . . . 101
6.3 Bayesian deep learning . . . 102
6.3.1 Bayesian neural networks . . . 102
6.3.2 Inference in Bayesian neural networks . . . 103

6.3.3 Loss-calibrated approximate inference in Bayesian neural networks . . . 105
6.4 Acoustic mosquito detection with loss-calibrated Bayesian neural networks . . . 107
6.4.1 Problem statement . . . 107
6.4.2 Prior work on audio detection . . . 108
6.4.3 The challenge of data imbalance . . . 109
6.4.4 Experiment . . . 110
6.4.4.1 Utility function . . . 110
6.4.4.2 Model . . . 110
6.4.5 Results . . . 111
6.4.6 Utility calibration . . . 113
6.5 Chapter summary . . . 114

7 Machine learning with variable label resolution data 116
7.1 Overview . . . 116
7.2 Introduction . . . 117
7.3 Background . . . 118
7.4 Methodology . . . 120
7.4.1 Framework overview . . . 120
7.4.2 Feature extraction and selection . . . 121
7.4.3 Gaussian kernel density estimation . . . 122
7.4.4 Convolutional neural network . . . 123
7.4.5 Traditional classifier baselines . . . 124
7.5 Experiments . . . 125
7.5.1 Description of datasets . . . 125
7.5.2 Data pre-processing . . . 126
7.5.3 Experimental design . . . 126
7.6 Classification performance . . . 127
7.7 Chapter summary . . . 132

8 Conclusion and further work 133
8.1 Conclusion . . . 133
8.2 Future work . . . 135
8.2.1 The future of HumBug . . . 135
8.2.2 Label super-resolution . . . 136
8.2.3 HumBug smartphone app . . . 137
8.2.4 Feature learning . . . 138

Bibliography 139

Declaration

The work in this thesis is based on research carried out at the Machine Learning Research Group, Department of Engineering Science, University of Oxford, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification, and it is the sole work of the author unless referenced to the contrary in the text.

Publications

Some of the work presented in this thesis has been published in journals and conference proceedings. Publications not featured in this thesis are starred (*). (**) denotes equal contribution in the author list.

1. **Mosquito Detection with Neural Networks: The Buzz of Deep Learning**, **I Kiskin**, BP Orozco, T Windebank, D Zilli, M Sinka, K Willis, S Roberts, *arXiv pre-print, submitted to ECML 2017*, 2017 (Kiskin et al., 2017).

2. **Bioacoustic detection with wavelet-conditioned convolutional neural networks**, **I Kiskin**, D Zilli, Y Li, M Sinka, K Willis, S Roberts, *Springer Neural Computing and Applications Special Issue in Deep learning for Music and Audio*, 2018 (Kiskin et al., 2018).

3. **Mosquito detection with low-cost smartphones: Data acquisition for malaria research**, Y Li, D Zilli, H Chan, **I Kiskin**, M Sinka, S Roberts, K Willis, *NeurIPS 2017 Workshop on Machine Learning for the Developing World*, 2017 (Li et al., 2017b).

4. **Cost-sensitive detection with variational autoencoders for environmental acoustic sensing**, Y Li, **I Kiskin**, D Zilli, M Sinka, H Chan, K

Willis, S Roberts, *NeurIPS 2017 Workshop on Machine Learning for Audio Signal Processing*, 2017 (Li et al., 2017a).

5. **Fast Mosqutio Acoustic Detection With Field Cup Recordings: An Initial Investigation**, Y Li, **I Kiskin**, M Sinka, D Zilli, H Chan, E Herreros-Moya, T Chareonviriyaphap, R Tisgratog, K Willis, S Roberts, *Detection and Classification of Acoustic Scenes and Events 2018 (DCASE) 2018 Workshop*, (Li et al., 2018).

6. **HumBug Zooniverse: a crowd-sourced acoustic mosquito dataset**, **I Kiskin**, AD Cobb, L Wang, S Roberts, *2019 NeurIPS Machine Learning for the Developing World workshop, 2020 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019–2020 (Kiskin et al., 2019, 2020).

7. **Super-resolution of Time-series Labels for Bootstrapped Event Detection**, **I Kiskin**, U Meepegama, S Roberts, *2019 ICML Time-series workshop*, 2019 (Kiskin et al., 2019).

8. ***Semi-separable Hamiltonian Monte Carlo for inference in Bayesian neural networks**, AD Cobb, AG Baydin, **I Kiskin**, A Markham, SJ Roberts, *2019 NeurIPS Bayesian Deep Learning Workshop*, 2019 (Cobb et al., 2019).

9. ***An Overview of Gaussian Process Regression for Volatility Forecasting**, B Liu**, **I Kiskin****, S Roberts, *IEEE International Conference on Artificial Intelligence in Information and Communication, 2020*, 2020 (Liu et al., 2020).

Hilary 2020

Ivan Kiskin

List of Figures

2.1 Neural network architecture. For clarity the diagram displays connections for a few units. Each layer is fully connected with activation functions. The number of layers are denoted $l = 1, \ldots, L$, with $L = 4$. The activations at layer l are denoted $\mathbf{a}^l$. The input $\mathbf{x}$ forms activation layer 1, $\mathbf{a}^1$. We denote the number of units in the second and third layers as P and Q respectively. 23

2.2 Example of a typical convolutional neural network used to detect two classes from a three-channel (RGB) image of size 32×32, parameterised as detailed in the figure. 27

2.3 Mosquito WAVE audio file (top), with feature representations of increasing saliency from top to bottom. 30

3.2 Timeline of the HumBug project organised by **A: Data collection**, **B: Smartphone app** development, and **C: Research output** publications. The endpoints are defined as reaching a stage where dependent work may be carried out, and not necessarily where development for each specific element ends. 38

4.1 Illustration of key properties of the bump wavelet, constructed from Equations (4.2) and (4.3). 56

4.2 The CNN pipeline. 1.5 s spectrogram of mosquito recording is partitioned into single-channel images ($c = 1$) of dimensions $h_1 \times w_1$. The images serve as input to a convolutional layer with N_k filters with kernel size $k \times k$. Feature maps are formed with dimensions reduced to $h_2 \times w_2$ following convolution. These maps are fully connected to N_d units in the dense layer, fully connected to 2 units in the output layer. 58

4.3 STFT (top) and wavelet (middle) representations of signal with $h_1 = 256$ frequency bins and wavelet scales respectively. Corresponding varying class labels (bottom) as supplied by human labellers. The wavelet representation shows greater contrast in horizontal constant frequency bands that correspond to the mosquito tone. 60
4.4 **BirdCLEF** subset: boxplots of mean accuracy per class (F_1 score) for $n = 30$ trials of the CNN and SVM methods, grouped by feature combination. 65
4.5 **Culex mosquito** dataset: plot of normalised feature coefficient against STFT frequency bin, (a), and wavelet centre frequency, (b), for the 10 % most confident predicted outputs over a test dataset. The learned spectra $\mathbf{x}_{i,\text{test}}(f)$ for the highest N scores closely match the labelled class spectra $\mathbf{x}_{i,\text{train}}(f)$. 67
4.6 **BirdCLEF** subset: plot of normalised feature coefficient against STFT frequency bin, (a), and wavelet centre frequency, (b), for the 10 % most confident predicted outputs over a test dataset. The learned spectra $\mathbf{x}_{i,\text{test}}(f)$ closely match the labelled class spectra $\mathbf{x}_{i,\text{train}}(f)$. 68

5.1 The MozzWear android app. The latest version is available on https://github.com/HumBug-Mosquito 73
5.2 An example recording and associated spectral features. Mosquito flight tones are present from 21 s to 31 s for cow capture #242, 35 s to 40 s from cow capture #243, 42 s to 52 s from cow capture #251. Other segments of the recording (including the high amplitude sections) are either background noise or human voices. 79
5.3 Example CNN architecture. Input to the CNN is a mel-frequency spectrum computed from a 0.1 second audio clip with $c = 1$ channel and dimension $h_1 \times w_1$. The CNN has N_k filters with kernel $K \in \mathbb{R}^{k \times k}$ thus it reduces the input dimension to $h_2 \times w_2$ following convolution with each filter. These feature maps are flattened (i.e. vectorised) before being fully connected to N_d hidden units in a dense layer. The last fully-connected layer produces the classification output. 80
5.4 Boxplots showing out-of-sample classification accuracy and F_1 scores across 100 randomised trials. 82
5.5 Confusion matrices of out-of-sample classification performance for models operating on the mel-frequency spectrum. The first value in each entry is the corresponding mean value among 100 simulation trials, while the value in the parenthesis reports the standard deviation. 83

5.6 The ensemble selection approach. There are M different VAE network candidate structures and N different combinations of classifier hyper-parameter values. 88
5.8 User interface for the classification page of the Zooniverse HumBug project, found at https://www.zooniverse.org/projects/yli/humbug/. A short-time Fourier transform spectrogram representation is offered above the audio file. 93
5.9 Statistics of the crowdsourced Zooniverse dataset. The dataset is comprised of four sources of data, labelled "**A**", "**B**", "**C**", and "**D**", described in Section 5.5.2. The total number of classifications made is 195,434. This is made on 80,101 overlapping 2 second chunks, each with a unique `audio_id`, which create a 22 hour dataset of unique audio. 57,710 (72 %) of the audio samples contain more than one label, of which 10,487 (18 %) contain disagreement. In total, approximately 1 in 10 recordings contains an audible mosquito, with the distribution given in (c). 94
5.10 Mean confusion matrices of classifications made with the CNN baseline on ten folds of the dataset, given in the form of $\mu \pm \sigma$, where μ is the mean, and σ is the standard deviation. The two subfigures show two weighted cross-entropies which encode our utility in detecting either class. The trade-off of false negative and true positive rate is evidenced when we change the weighting of each class in the loss function. 97

6.1 Confusion matrices of the decisions with respect to the utility function of Table 6.1 with $FP = 0, FN = 20$, and their corresponding expected utility, $\mathcal{U}$. The LCBNN achieves a lower false positive rate and therefore a higher utility. 113
6.2 A comparison of the effect of SNR on the predicted utility of choosing the mosquito. **Top:** The LCBNN's behaviour is better calibrated as it only expects a high utility for high SNR. **Bottom:** Above a threshold, the LCBNN expects higher utility, aligned with our preference for reduced false negatives. By jointly training the model with the utility, the LCBNN is able to calibrate its behaviour to our preferences in a way the weighted model cannot. 114

7.1 Framework comprising a feature extraction & selection layer, an inner classifier and an outer classifier. The arrows represent data flows. 120
7.2 Top: log-mel spectrogram of 100 seconds of audio data at a signal-to-noise ratio of -15 dB. The KS-selected features are shown as green dashed lines (see text for details). Bottom: the corresponding fine and weak labels for the log-mel spectrogram. 121

7.3 The CNN architecture. Spectrogram of mosquito recording fed as input to convolutional layer with 32 filters and kernel $2 \times 2 \times 1$. Generated feature maps are down-sampled by a max-pooling layer from 127×9 to 63×4. It is then connected to a dense layer with 256 neurons and finally the output with 2 neurons. 124
7.4 Boxplots of F_1 scores for **Experiments 1** and **2**. The benefit of training on coarse 0 data in (b) for the generative models such as the KDE is demonstrated relative to the baseline scores of (a). The discriminative approaches tend to achieve lower F_1 scores on the whole. 128
7.5 Boxplots of F_1 scores for **Experiments 3** and **4**. LDA, SVM, RF and MLP are trained on finely labelled data only whilst the Gaussian KDE is trained on the finely labelled data and coarse class 0 data. The KDE most strongly benefits from median and rejection filtering, due to its reasonable distribution of positive and negative predictive accuracy and principled handling of uncertainty, respectively. 130
7.6 Ratio of data rejected during **Experiment 4**, grouped by SNR. 130

List of Tables

2.1 An example of a utility matrix with elements $\mathcal{U}_{kj}$ for a cancer treatment problem. The rows correspond to the true class, whereas the columns correspond to the decision. 14

4.1 Results for grid search over hyperparameter values. The cross-validated F_1 score is reported for optimal hyperparameters found (in bold). 62
4.2 Execution time given in seconds for feature-classifier pipelines trained on 15 minutes (900 s) and evaluated over 15 minutes of audio data sampled at 8000 Hz. Run times are an average of three passes on a mid-range desktop with an Intel i7-4790k CPU with 16 GB DDR4 RAM and an NVIDIA GTX 970 GPU. 63
4.3 **Mosquito detection**: summary classification metrics reported as means ± the standard deviation from $n = 30$ random hold out dataset splits with 50 % training data, and 50 % test data. 64
4.4 **BirdCLEF subset**: summary classification metrics reported as means ± the standard deviation from $n = 30$ random dataset splits with 50 % training data, and 50 % test data. 64

5.1 Statistics of detection accuracies with 7 mosquito species. Results are generated from 100 random splits of balanced data. 75
5.2 Number of captured mosquito individuals and durations of recorded mosquito flight tones for different species. 78

6.1 **Utility A** matrix to favour precision (passive mode). Applies to scenarios where false positives are too costly. In our application, this is the requirement not to trigger too many recordings and disturb the user to the point where the app is disabled for being too intrusive or consuming too much power. 111

6.2 **Utility B** matrix to favour recall (active mode). Applies to scenarios where false negatives have potentially lethal consequences. In our application, this is the requirement not to miss detection of a deadly species. 111
6.3 Results over 10 folds of cross-validated data for two utilities. Acc. corresponds to the results after the integration. We show that the LCBNN achieves both the highest accuracy in decisions as well as the highest utility with consistently lower standard deviation in comparison to standard and weighted cross-entropy methods. 112

7.1 **Experiment 5**: CNN outer classifier: metrics reported as means ± one standard deviation at an SNR of −19.8 dB for 40 iterations. 131

Nomenclature

ANN Attention Neural Network

AR Autoregressive

BNN Bayesian Neural Network

CDC Center for Disease Control and Prevention

CNN Convolutional Neural Network

CWT Continuous Wavelet Transform

DCT Discrete Cosine Transform

FT Fourier Transform

GNB Gaussian Naïve Bayes

HMM Hidden Markov Model

KDE Kernel Density Estimator

KL Kullback–Leibler

KS Kolmogorov–Smirnov

LCBNN Loss-Calibrated Bayesian Neural Network

LDA Linear Discriminant Analysis

LPCC Linear Prediction Cepstral Coefficients

MAP Maximum A Posteriori

MFCC Mel-Frequency Cepstral Coefficient

MIL Multiple Instance Learning

MLE Maximum Likelihood Estimate

MLP Multi-Layer Perceptron

NB Naïve Bayes

NN Neural Network

PCA Principal Component Analysis

PCM Pulse-Code Modulation

RBF Radial Basis Function

ReLU Rectified Linear Unit

RF Random Forest

RFE Recursive Feature Elimination

ROC Receiver Operating Characteristic

SED Sound Event Detection

SGD Stochastic Gradient Descent

SNR Signal-to-Noise Ratio

STFT Short-Time Fourier Transform

SVM Support Vector Machine

USAMRU-K United States Army Medical Research Unit, Kenya

VAE Variational Autoencoder

Chapter 1

Introduction

1.1 Overview

Mosquitoes are responsible for over one billion cases of disease and over one million deaths each year. Malaria alone kills more than 400,000 people each year (World Health Organization et al., 2019), and viruses carried by mosquitoes, such as yellow fever, dengue, chikungunya and more recently zika, are spreading and increasingly impacting on human health.

The goal of this thesis is to produce tools to detect malaria-bearing mosquito species from their acoustic signature. In doing so, we are able to create systems that help deliver aid to malaria-stricken areas most in need. Furthermore, in developing these tools, we are also able to improve on machine learning applications to audio. These improvements take place in the feature transformation space, the application of deep learning to scarce and partially labelled data, and at the intersection of acoustics and Bayesian utility theory.

Mosquito occurrence data is naturally difficult to obtain. However, large-scale machine learning typically require large volumes of data (Sun et al., 2017). As a lack of data hinders long-term progress to the overall objective, we dedicate resources towards novel solutions to acquire data. We achieve this by deploying a data-efficient algorithm on a low-cost smartphone app in both laboratory and field

conditions. In parallel to the data acquisition, we develop offline machine learning solutions that are limited in structural complexity. We bridge together signal processing and deep learning to make use of wavelet transformations as mid-level feature representations. As a result, we outperform human experts on detecting mosquitoes from expertly labelled acoustic data, where we take a gold-standard labeller as ground truth. Additionally, we show transferability to other bioacoustic applications with no hyperparameter re-tuning. Moreover, we show that the model learns the salient parts of the data and does not classify based on secondary factors such as the microphone frequency response.

With early prototype systems deployed in Thailand demonstrating proof of concept, we involve citizen science to accumulate data labels, which allows us to address a broader range of research problems. We publish data and baseline models to accelerate research in the wider community. With this additional information we are able to scale our approach to utilise more sophisticated Bayesian neural network models to address the data imbalance and user preference on error types when making classifications. Finally, we investigate ways of extracting maximal information from the accumulated mixture of data and label sources.

More specifically, the overall goal of the thesis is motivated by the HumBug project, which we first outline in Section 1.2. Our main contributions are described in Section 1.3, and we describe the overall structure of the thesis in Section 1.4.

1.2 Problem statement

Traditional mosquito survey methods, such as human landing catches (where the person conducting the survey uses themselves as bait to attract the mosquitoes), are time-consuming, expensive, and spatially limited. Furthermore, the surveyor is exposed to the risk of catching the diseases they are trying to prevent. Consequently, many mosquito distribution models that map the range of these insects rely on small quantities of poorly spatially distributed occurrence data. The data produced

during mosquito surveillance are needed to identify emerging insecticide resistance, facilitate effective and evidence-led insecticide intervention programmes as well as model current and future vector-borne disease transmission. There is therefore an urgent need to develop new mosquito survey methods that can provide real-time species-specific occurrence and abundant data without human risk.

The HumBug project* is a multidisciplinary collaboration between the University of Oxford and Royal Botanic Gardens, Kew. As part of the project deliverables, we have developed a novel mosquito survey tool that transforms a budget smartphone into a sensor that detects and identifies mosquitoes using the acoustic signature of their flight tone.

HumBug can generate unprecedented levels of urgently needed high-quality, spatially accurate mosquito occurrence data without incurring any risk to those conducting the surveys. It is low cost and can be used on other wearable smart devices as well as low-energy acoustic loggers. The sensor records the time and location, along with the mosquito flight tone and uploads the data to a central server where the species are identified using a suite of algorithms.

HumBug was originally funded via a 2014 Google Impact Award and Orchid, and has, as of 2020, been awarded funding from the Bill and Melinda Gates Foundation, allowing further development of this innovative system in selected field sites within Tanzania and the Democratic Republic of Congo.

Working with established partners in Ifakara Health Institute (Tanzania) and the School of Public Health at the University of Kinshasa (Democratic Republic of Congo), HumBug will be tested in rural communities to provide high-quality, real-time data on the diversity, distribution and abundance of malaria-transmitting mosquitoes. Combining these data with environmental variables from remote sensing satellite data will also allow us to create dynamic real-time heat maps of their distribution, providing information to i) healthcare practitioners to better

*humbug.ox.ac.uk

target mosquito control methods, ii) health policymakers to enhance more effective intervention policies and, iii) the academic community, providing unprecedented levels of occurrence, behavioural and ecological mosquito data.

1.3 Main contributions

This thesis showcases successful application of machine learning systems of appropriate complexity to a range of challenges faced in real-world detection. These challenges arise out of necessity following our solution to the problem statement of Section 1.2. We summarise our main contributions as follows:

1. A low-cost smartphone real-time data collection system, with a support vector machine operating on mel-frequency cepstral coefficient features. We thus facilitate safe and efficient audio recording of mosquitoes (Li et al., 2017b).

2. Showcasing the effectiveness of wavelet transformations as feature representations when allied with convolutional neural networks in bioacoustic applications (Kiskin et al., 2018).

3. Supplying machine learning baselines and labelled data of mosquito audio recordings thanks to a concentrated collaborative research effort with citizen science (Kiskin et al., 2019, 2020).

4. Developing and applying a framework that combines Bayesian decision theory with Bayesian neural networks to perform detection with user-defined utility functions on imbalanced class data (Kiskin et al., 2020).

5. Extracting maximal information from a mixture of label qualities in various time resolutions for the purpose of maximising classification performance (Kiskin et al., 2019).

1.4 Structure

In Chapter 2, we introduce the reader to necessary background material to cover the thesis. We define signals and time series, which we refer to frequently. We also describe the basics of machine learning modelling that we build on in the thesis, as well as give detail on audio data analysis. In Chapter 3 we supply a timeline of the HumBug project, with emphasis on key events in the data collection process, algorithmic design, and resulting academic literature. In Chapter 4 we present early works which show how deep learning can be used successfully in a data-scarce scenario when allied with wavelet transformations of audio data streams. We show that our model remains effective when transferring to other problem domains and gives interpretable results. In Chapter 5 we explore the research output which revolves around data collection and modelling under data and computational constraints. In Chapter 6 we proceed to address a new key theme – uncertainty in classification, where we talk about why this knowledge is important and suggest mechanisms for how to best incorporate it. With the availability of uncertainty, we show that it is possible to incorporate a user utility into the training objective of a Bayesian neural network. In Chapter 7 we discuss further work which combines sources of labels at different temporal resolution, as well as accuracies. Finally, we conclude in Chapter 8 with our main contributions and future research directions.

Chapter 2

Machine learning for signal detection

2.1 Overview

In this chapter we introduce the literature which defines the context necessary for the research presented in the thesis. In Section 2.2 we discuss foundational principles which recur throughout the thesis. We define our understanding of a signal in the context of time series in Section 2.2.2. We then describe the tasks of regression and classification. We show how characterising parts of time series can be approached in the classification framework in Section 2.2.4.

We then discuss the role that uncertainty plays in machine learning in Section 2.3, and introduce the statistical underpinnings which we re-visit in Chapter 6. We show the effect of measurement noise, incomplete domain coverage and model error.

We examine Bayesian decision theory in Section 2.3.4, which we build on to create end-to-end systems that take asymmetrical utility into account later in Chapter 6.

In Section 2.4 we introduce a set of model types for classification. We focus on classifiers which are used as baselines throughout the thesis. In particular, we cover random forests, the support vector machine, and neural networks.

We narrow our focus to consider the sub-discipline of audio data analysis in Section 2.5 and describe the higher level data descriptions commonly used in audio processing and analysis. In Section 2.6 we summarise and conclude.

2.2 Foundational principles

2.2.1 Time series

A *time series* is a series of data points indexed in time order. Thus it is a sequence of discrete-time data. A *stationary* time series is a stochastic process whose statistical properties such as the mean and variance all do not change when shifted in time (Box et al., 2015).

A stochastic model for a time series will generally reflect the fact that observations close together in time will be more closely related than observations further apart (e.g. kernel methods with an exponentiated quadratic, log-linear models such as autoregressive methods). In addition, time-series models will often make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values.

With the main objective of the thesis being signal detection, we define our understanding of a signal more specifically in the following section.

2.2.2 What is a signal?

We define a *signal* as information content which characterises an event, or phenomenon of interest. We define an *observation* as an instance of data, such as an audio stream. In this thesis we are specifically concerned with extracting signals from streams of audio data – which typically contain a lot of *noise*. The underlying concept of noise in machine learning is fundamental, as noise is present in all observations made in the real world. We define noise as anything present in the

observation that is spurious and extraneous to the information source of interest. Often noise is introduced due to the limitations of the capturing process. Under this definition, noise refers to a *residual*, which is used to explain all the variance in the data stream that does not bear information regarding the variables of interest. We discuss sources of noise in more depth in Section 2.3.1.

When the signal $s(t)$ is a stationary stochastic process, its power is defined to be the value of its correlation function $R_s(\tau)$ at the origin:

$$R_s(\tau) \equiv \mathbb{E}[s(t)s(t+\tau)]; \quad P_s = R_s(0), \tag{2.1}$$

where $\mathbb{E}$ refers to the expected value. The noise power P_n is similarly related to its correlation function:

$$P_n = R_n(0) = \mathbb{E}[n^2(t)]. \tag{2.2}$$

Therefore, the Signal-to-Noise Ratio (SNR) is the ratio of mean squares of s and n:

$$\text{SNR} = \frac{\mathbb{E}[s^2]}{\mathbb{E}[n^2]}. \tag{2.3}$$

In a non-stationary time series the signal may manifest itself through a change in dynamics, and hence the statistics of the time series. We may be interested in tracking the behaviour of our variable of interest over time to predict future behaviour, or alternatively be on the lookout for a change in behaviour of the variable. For the former, it is common to introduce data transforms to remove non-stationarity to match the assumptions made by many models (Box et al., 2015). In the second scenario, which is known as *change point detection*, we are interested to discover the point where an event may have occurred that caused a change in the system dynamics. Formally, in statistical analysis, change point detection tries to identify times when the probability distribution of a stochastic process or the underlying dynamics of a time series change. In general the problem concerns both detecting whether or not a change has occurred, or whether several changes might have occurred, and identifying the times of any such changes. Examples of this include safety-critical applications, where component failure may have a knock-on effect on the dynamics of a system.

We now discuss in more depth common approaches that underpin information extraction and event detection in signals.

2.2.3 Regression

In many problem domains, we may be interested in predicting the values of the time series in the future (or past) based on the data we currently have available. Prediction within the range of values in the dataset used for model-fitting is commonly referred to as *interpolation*. Prediction outside this range of the data is known as *extrapolation*. Both interpolation and extrapolation are performed by *regression*.

Regression models predict a continuous value, or probability distribution, of the independent variable given known observations of the dependent variables.

2.2.4 Detection

In the context of the thesis, we consider detection as classification by regressing continuous data onto a set of indicator variables, one per class, over which we wish to infer probabilities. To do so, we use functions that provide mappings from the feature space to class labels, $c \in C$, where C is the set of all possible classes. In order to achieve this goal we use a likelihood function to convert latent regression variables to instances of class probabilities. If we set K to be the number of classes, then the function output $f(x)$ can be passed through an activation function*, which squashes the output to an instance of a discrete probability distribution, thus mapping $\mathbb{R}^K \mapsto [0,1]^K$.† In binary classification, we only require a one-dimensional output. This is equivalent to scaling the output to lie between 0 and 1, such that we can treat the output as the class probability p_0 of class 0, and $p_1 = 1 - p_0$ as the class probability of class 1. The activation function goes hand-in-hand with

*The inverse of the activation function is called the link function.

†We note that as $\sum_{i=1}^{K} p_i = 1$, the output space is strictly fully defined in $[0,1]^{K-1}$.

the likelihood. In two-class classification with logistic regression, the outcome (or *response variable*) can be modelled by a Bernouilli distribution* (Hosmer Jr et al., 2013). Similarly, the likelihood function for multi-class classification can be derived from a generalised Bernoulli distribution (also known as the *categorical distribution*). The categorical distribution is a discrete probability distribution that describes the possible results of a random variable that can take on one of K possible categories, with the probability of each category separately specified (Murphy, 2012). We give a Bayesian interpretation of the detection problem with multiple classes with the Bayesian deep learning framework in Chapter 6.

When we make classifications, irrespective of model choice, it is helpful to quantify predictive uncertainty. The following section addresses various sources of uncertainty and discusses the frameworks that can help us quantify and propagate it correctly for making decisions.

2.3 Uncertainty in machine learning

Uncertainty is inherent to the application of machine learning to real-world data, and can be coherently managed using the mathematics of probability. Probability theory provides a consistent framework for the quantification and manipulation of uncertainty and forms one of the central foundations for pattern recognition (Bishop, 2006, p. 12).

To structure this discussion, we can decompose the sources of uncertainty in the context of an acoustic detection problem as *measurement noise*, *incomplete domain coverage*, and *model error*. Probabilistic reasoning helps address the three key areas outlined as follows.

- To address noisy observations, probability theory helps quantify the expected value and variability in our observations.

*Or more generally, a beta-binomial distribution.

- In terms of the incomplete coverage of the domain, probabilistic modelling allows us to quantify the expected distribution and density of observations even in regions of the domain in which observations are more sparse. This lets us quantify the uncertainty which such sparse data naturally induces.
- In terms of model error, probability helps to understand and quantify the expected capability and variance in performance of our predictive models when applied to new data, typically in terms of expected generalisation performance.

We explore each of these sources of error in more depth in the following sections.

2.3.1 Measurement error

The effect of observation error is to diminish the useful information content, which we refer to as the signal. In the context of audio, errors stem from various sources. We can further distinguish between *random* and *systematic* error. For example, random additive noise from the environment (in the form of vehicles, wind, etc.) may mask the signal of interest. On the other hand, a systematic error can be caused by an inaccurate reproduction in frequencies due to the microphone itself (microphone spectral response properties can often be obtained from manufacturers but are very difficult to correct in practice, due to the interplay of acoustics in an environment). This attenuates various frequencies by different amounts, causing misrepresentation of the informative signal contents.

The variability impacts not just the inputs or measurements but also the outputs. A noisy observation may lead to an incorrect class label being assigned for the purposes of training. For example, our crowdsourced data used in Section 5.5 has entirely unknown ground truth labels. This can in part be circumvented through an estimate of labeller accuracy, which may be obtained by calibrating the crowd voting against carefully constructed training labels. However, this requires additional resources and still can only provide an estimate. We therefore have multiple sources of uncertainty (with varying magnitude) stemming from both hand-labelling signal sections, as

well as from obtaining labels from volunteers, whose behaviour is unknown (and we may even encounter spam or adversarial behaviour (Difallah et al., 2012)).

2.3.2 Incomplete domain coverage

Furthermore, uncertainty stems from the data collection procedure itself. In statistics, a *random sample* refers to a collection of observations chosen from the domain without systematic bias. In practice, it will be difficult to sample without bias. For example, in the case of mosquito detection, due to the species variation over geographical regions, we are unable to sample from all possible regions to obtain unbiased data in pilot studies. By taking biased observations from a domain, decisions may be made from a model output operating on a sampled expectation (which does not match the true expectation) of the data.

Data collection is often expensive, taking a significant amount of time to amass quality datasets for any fairly specific domain. For example, when trying to classify mosquito species, we are uncertain whether the variability in the statistics of the sample data demonstrated is due to the species itself, or other contributing factors such as wingspan length (which is related to fundamental frequency (Clements, 1999)), age or gender. To determine the species, we therefore require a large number of samples to cover the domain. As the volume of the space required to cover by samples increases, in order to obtain a statistically sound result, the quantity of data needed to support a result grows exponentially with the dimensionality. This is an example of the curse of dimensionality, first introduced as a concept by Bellman (1961).

These factors combined lead to uncertainty in how well our sample represents the full dataset, which we need to carefully account for when choosing models and modelling techniques.

2.3.3 Model error

A further source of uncertainty is model error. A common aphorism attributed to British statistician George Box states that *"all models are wrong but some are useful"* (Box, 1976). This refers to the notion that a model is an approximation of a real-world phenomenon, which further constitutes a source of uncertainty. We differentiate this from uncertainty in the parameters (or hyperparameters) of the model, which introduces output uncertainty. With reference to our mosquito detection problem, when using convolutional neural networks the error is compounded with imperfect hyperparameter search and non-global optimisation. Regardless, the model may be good enough to distinguish classes, according to a user-defined error metric such as precision–recall (Davis and Goadrich, 2006). However, quantifying how certain the model is in its predictions puts the practitioner in a better position to make informed decisions based on the model outputs.

To summarise, we have uncertainty stemming from measurement noise, an incomplete coverage of the domain through the nature of observations, as well as modelling errors. The modelling errors are due to both inductive bias in model choice as well as uncertainty in the associated parameters of that model.

Given these limitations, we turn in the next section to a Bayesian framework for modelling. Many classifiers are deterministic in their formulation, which fundamentally limits their ability to be applied to certain problem aspects involving producing a *well-calibrated* probability distribution over the output.* Given our ability to understand and quantify probabilities, we would like to make use of models that output well-calibrated probabilities to make decisions in a principled manner. This leads to the consideration of Bayesian decision theory.

*By *well-calibrated* we mean that the distribution captures the underlying uncertainty of the system. Well-calibrated probability distributions are obtained by propagating uncertainty both in parameters and directly via the input, which Bayesian frameworks naturally accomplish.

Table 2.1: An example of a utility matrix with elements $\mathcal{U}_{kj}$ for a cancer treatment problem. The rows correspond to the true class, whereas the columns correspond to the decision.

		Decision	
	Utility	Cancer	Normal
True	Cancer	0	-1000
	Normal	-1	0

2.3.4 Bayesian decision theory

Bayesian decision theory seamlessly combines with Bayesian statistics in a manner that ensures uncertainty is appropriately incorporated in decision-making tasks. In using this framework, we are able to quantify the trade-off between various decisions, by making use of, and propagating, probabilities. An agent operating under such decision theory uses the concepts of Bayesian statistics to estimate the expected value of its actions, and update its expectations based on new information. These agents can be referred to as estimators (Berger, 1985). Any agent expected to make a decision for a specific task must be informed as to how their choices are valued. We may imagine that in a task such as medical disease diagnosis, the cost incurred by the agent in missing a cancer in a patient (a false negative) is greater than that of diagnosing a false positive (which can be managed and mitigated by further checks). Therefore, a clear way of defining the goal of a task is to define a *loss function* that captures the way in which correct decisions are valued, and incorrect ones penalised (Bishop, 2006, p. 41). We note that we use the negated loss function, the *utility* function ($\mathcal{U} = -L$), to avoid confusion between the loss function of a neural network, and a loss function associated with decision theory (which we jointly optimise in Chapter 6).

Suppose that, for a new value of $\mathbf{x}$, the true class is C_k and that we assign $\mathbf{x}$ to class C_j (where j may or may not be equal to k). In doing so, we gain some unit of utility that we denote by $\mathcal{U}_{kj}$, which we can view as the k, j^{th} element of a utility matrix. For instance, in our cancer example, we might have a utility matrix of the

form shown in Table 2.1. This particular utility matrix says that there is no utility (and hence no penalty incurred) if the correct decision is made, there is a utility of -1 if a healthy patient is diagnosed as having cancer, whereas there is a utility of -1000 (and hence a large penalty of 1000) if a patient having cancer is diagnosed as healthy. The optimal solution is the one which maximises the utility function. However, the utility function depends on the true class, which is unknown. For a given input vector $\mathbf{x}$, our uncertainty in the true class is expressed through the joint probability distribution $p(\mathbf{x}, C_k)$ and so we seek instead to maximise the average utility, where the average is computed with respect to this distribution, which is given by

$$\mathbb{E}[\mathcal{U}] = \sum_k \sum_j \int_{\mathcal{R}_j} \mathcal{U}_{kj} p(\mathbf{x}, C_k)\, \mathrm{d}\mathbf{x}. \tag{2.4}$$

Each $\mathbf{x}$ can be assigned independently to one of the *decision regions* $\mathcal{R}_j$. By definition, a decision region $\mathcal{R}_j$, one for each class, divides the input space such that all points in $\mathcal{R}_j$ are assigned to class C_j.* Our goal is to choose the regions $\mathcal{R}_j$ in order to maximise the expected utility (2.4), which implies that for each $\mathbf{x}$ we should maximise $\sum_k \mathcal{U}_{kj} p(\mathbf{x}, C_k)$. We can use the product rule $p(\mathbf{x}, C_k) = p(C_k|\mathbf{x})p(\mathbf{x})$ to eliminate the common factor of $p(\mathbf{x})$. Thus the decision rule that maximises the expected utility is the one that assigns each new $\mathbf{x}$ to the class j for which the quantity

$$\sum_k \mathcal{U}_{kj} p(C_k|\mathbf{x}) \tag{2.5}$$

is a maxmimum. This rule is trivial, once we know the posterior class probabilities $p(C_k|\mathbf{x})$. In the following section we relate classification to the decision framework defined here.

2.4 Introduction to model classes

The goal in classification is to take an input vector $\mathbf{x}$ and to assign it to one of K discrete classes C_k where $k = 1, \ldots, K$. In the most common scenario, the classes

*Note that each decision region need not be contiguous but could comprise some number of disjoint regions.

are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into decision regions whose boundaries are called *decision boundaries* or decision surfaces. We break down the classification problem into two separate stages, the *inference* stage in which we use training data to learn a model for $p(C_k|\mathbf{x})$, and the subsequent decision stage in which we use these posterior probabilities to make optimal class assignments. An alternative possibility would be to solve both problems together and simply learn a function that maps inputs $\mathbf{x}$ directly into decisions. Such a function is called a *discriminant function*.

In fact, we can identify three distinct approaches to solving decision problems, all of which have been used in practical applications. These are given, in increasing order of complexity:

1. Constructing a discriminant function that directly assigns each vector $\mathbf{x}$ to a specific class.

2. Modelling the conditional probability distribution $p(C_k|\mathbf{x})$ in an inference stage, and then subsequently using this distribution to make optimal decisions. By separating inference and decision, we gain numerous benefits, as discussed in Bishop (2006, p. 43). One technique is to model $p(C_k|\mathbf{x})$ directly, for example by representing the conditional probabilities as parametric models and then optimising the parameters using a training set.

3. Alternatively, we can adopt a generative approach in which we model the class-conditional densities given by $p(\mathbf{x}|C_k)$, together with the prior probabilities $p(C_k)$ for the classes, and then we compute the required posterior probabilities using Bayes' theorem:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}. \tag{2.6}$$

While some models may be learned by re-formulating the learning objective to suit either of these three approaches (e.g. logistic regression, which would fall under a conditional probability approach, can be re-cast as learning isotropic Gaussians in

a generative model), we give examples of common learning approaches that can be interpreted within these three paradigms.

To elaborate on the first approach, a discriminant is a function that takes an input vector $\mathbf{x}$ and assigns it to one of K classes, denoted C_k. For the case of a linear discriminant, the decision surface is a hyperplane. Examples of this approach include the margin optimisation objective in support vector machines, which we discuss in Section 2.4.3.

Examples of the second approach include tree-based methods such as decision trees or random forests, where we learn class-conditional densities. We examine how basic decision trees work, how individual decisions trees are combined to make a random forest, and offer insight into why random forests are commonly used in classification in Section 2.4.1. Furthermore, we view neural network training under this category. We introduce neural networks in Section 2.4.4, and expand on the convolutional neural network architecture, which we utilise throughout Chapters 4 to 7. Similarly, when we use Bayesian neural networks for making decisions in Chapter 6, the class-conditional probabilities are given by the output of the softmax function. We re-visit this in detail in Section 6.3.

Examples of generative approaches include naïve Bayes, which learns an approximate joint distribution, which we re-visit in Chapter 7, and variational autoencoders which we utilise in Section 5.4.

The classification algorithms we describe here are used throughout the thesis, either as baselines or due to their computational efficiency for real-world applications.

2.4.1 Decision trees

A decision tree is a flow-chart-like structure, where each internal (non-leaf) *node* denotes a test on an attribute, each *branch* represents the outcome of a test, and each *leaf* (or terminal) node holds a class label. The topmost node in a tree is the root node.

A tree is built by splitting the source set, constituting the root node of the tree, into subsets – which constitute the successor children. The splitting is based on a set of splitting rules based on classification features (Quinlan, 1987). This process is repeated on each derived subset by *recursive partitioning*. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the prediction according to a range of possible stopping criteria (Hothorn et al., 2006).

In this thesis we describe the quality of split metric which is used in the `scikit-learn` implementation, the *Gini impurity*. The Gini impurity, I_G, is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset.* The Gini impurity can be computed by summing the probability p_i of an item with label i being chosen multiplied by the probability of a mistake in categorising that item, $\sum_{k \neq i} p_k$. It reaches its minimum (zero) when all cases in the node fall into a single target category. For a set of items with K classes, suppose $i \in \{1, 2, ..., K\}$, and let p_i be the fraction of items labelled with class i in the set. Then,

$$I_G(p) = \sum_{i=1}^{K} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{K} p_i(1 - p_i) = \sum_{i=1}^{K} p_i - \sum_{i=1}^{K} {p_i}^2 = 1 - \sum_{i=1}^{K} {p_i}^2. \tag{2.7}$$

A drawback to common approaches with decision trees is that trees that are grown very deep tend to overfit their training sets, i.e. have low bias, but very high variance.† Random Forests (RFs) provide a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance (Friedman et al., 2001, pp. 587–588). This comes at the expense of a small increase in the bias and some loss of interpretability, but generally boosts the performance in the final model. We discuss these in more depth in the following section.

*In this sense, the Gini impurity, is a variation of the usual entropy measure for decision trees.

†The bias-variance decomposition under a mean square error states that the predictive error can be decomposed into a fixed sum of irreducible noise error, a bias term, and a variance term (Domingos, 2000).

2.4.2 Random forests

The training algorithm for random forests* applies the general technique of bootstrap aggregating, or *bagging*, to decision tree learners. Given a training set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with responses $\mathbf{Y} = \{y_1, \ldots, y_N\}$, bagging repeatedly ($B$ times) selects a random sample with replacement of the training set and fits trees to these samples:

1. Sample, with replacement, N training examples from $\{\mathbf{X}, \mathbf{Y}\}$; call these $\{\mathbf{X}_b, \mathbf{Y}_b\}$.
2. Train a classification tree $\mathbf{f}_b$ on $\{\mathbf{X}_b, \mathbf{Y}_b\}$.

After training, predictions for unseen samples $\mathbf{x}^*$ can be made by taking the majority vote from all the individual classification trees on $\mathbf{x}^*$.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees; bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

In addition to the bagging procedure described above, random forests use a modified tree learning algorithm that selects a random subset of the features at each candidate split in the learning process. In the ordinary bootstrap procedure, if one or a few features are very strong predictors for the response variable, these features will be selected in many of the B trees, causing them to become correlated. Due to the random feature selection, the trees are more independent of each other compared to regular bagging, which often results in better predictive performance (Pal, 2005).

*The word *forest* itself implies the existence of many trees.

2.4.3 Support vector machines

A Support Vector Machine (SVM) is a discriminant classifier that attempts to find the optimal hyperplane between the classes. Optimality here is defined as the maximal margin between the classes and the hyperplane as defined by the support vectors. Leeway is given for some misclassification either side of the margin in the soft margin version of the SVM. The SVM is also part of a larger set of methods known as kernel methods in which the feature space can be transformed into a high-dimensional feature space through the use of a kernel function. The hyperplane is then found in this feature space. In this way, the SVM can learn non-linear decision boundaries.

As defined under the categorisation in Section 2.4 that we adopt from Bishop (2006), the SVM directly obtains a discriminant function and so does not natively provide posterior probabilities.

We begin our discussion of SVMs by considering the two-class classification problem using linear models of the form

$$y(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + b, \tag{2.8}$$

where $\boldsymbol{\phi}(\mathbf{x})$ denotes a fixed feature-space transformation, and we have made the bias parameter b explicit. Note that we shall shortly introduce a dual representation expressed in terms of kernel functions, which avoids having to work explicitly in feature space. The training dataset comprises N input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$, with corresponding target values $t_1, \ldots, t_N$, where $t_n \in \{-1, 1\}$, and new data points $\mathbf{x}$ are classified according to the sign of $y(\mathbf{x})$. We shall assume for the moment that the training dataset is linearly separable in feature space, so that by definition there exists at least one choice of the parameters $\mathbf{w}$ and b such that a function of the form (2.8) satisfies $y(\mathbf{x}_n) > 0$ for points having $t_n = +1$ and $y(\mathbf{x}_n) < 0$ for points having $t_n = -1$, so that $t_n y(\mathbf{x}_n) > 0$ for all training data points.

The SVM approaches this problem through the concept of the margin, which is

defined to be the smallest distance between the decision boundary and any of the samples. In SVMs the decision boundary is chosen to be the one for which the margin is maximised.

As shown in Bishop (2006, Ch. 7), the maximum margin classification problem can be formulated in its *primal* form as the constrained optimisation

$$\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2, \tag{2.9}$$

with the condition that

$$t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \ldots, N. \tag{2.10}$$

We can introduce Lagrange multipliers $a_n \geq 0$, with one multiplier a_n for each of the constraints in (2.10), giving the Lagrangian function

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b) - 1. \tag{2.11}$$

Eliminating $\mathbf{w}$ and b from $L(\mathbf{w}, b, \mathbf{a})$, through differentiation with respect to the parameters (Bishop, 2006, Ch.7), gives the *dual* representation of this problem, in which we maximise

$$\hat{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \tag{2.12}$$

with respect to $\mathbf{a}$, subject to the constraints

$$a_n \geq 0 \quad n = 1, \ldots, N, \tag{2.13}$$

$$\sum_{n=1}^{N} a_n t_n = 0, \tag{2.14}$$

where the *kernel function* is defined by $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$. The maximisation takes the form of a quadratic programming problem in which a quadratic function is optimised subject to a set of inequality constraints. Approaches to solve this are covered in Bishop (2006, Ch. 7).

The solution to a quadratic programming problem in M variables in general has computational complexity that is $\mathcal{O}(M^3)$. By re-writing as the dual formulation of

Equation (2.12), we have turned the original optimisation problem of minimising $||\mathbf{w}||$ over M variables* into an optimisation over N variables. For a fixed set of basis functions, whose number M is smaller than the number N of data points, the move to the dual problem appears disadvantageous. However, it allows the model to be reformulated using kernels, and so the maximum margin classifier can be applied efficiently to feature spaces whose dimensionality exceeds the number of data points, including infinite feature spaces.

SVMs provide a solution to the application of models of fixed basis functions to large-scale problems. One advantage of SVMs is that, although the training involves nonlinear optimisation, the objective function is convex, and so the solution of the optimisation problem is relatively straightforward. The number of basis functions in the resulting models is generally much smaller than the number of training points, although it is often still relatively large and typically increases with the size of the training set.

An alternative approach is to fix the number of basis functions in advance but allow them to be adaptive – in other words to use parametric forms for the basis functions in which the parameter values are adapted during training. The most successful model of this type in the context of pattern recognition is the feed-forward neural network, also known as the Multi-Layer Perceptron (MLP)†, discussed in the following section. For many applications, the resulting model can be significantly more compact, and hence faster to evaluate, than a support vector machine having the same generalisation performance. The price to be paid for this compactness is that the likelihood function, which forms the basis for network training, is no longer a convex function of the model parameters. In practice, however, it is often worth investing substantial computational resources during the training phase in order to obtain a compact model that is fast at processing new data. We elaborate on such models in the following section.

*M is the dimensionality of $\boldsymbol{\phi}(\mathbf{x})$, as well as $\mathbf{w}$.

†In fact 'MLP' is really a misnomer, because the model comprises multiple layers of regression and non-linearities rather than multiple perceptrons (Bishop et al., 1995, p.226).

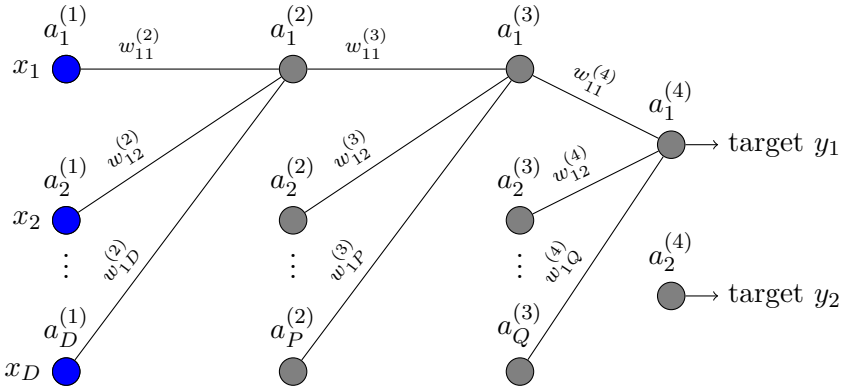


Figure 2.1: Neural network architecture. For clarity the diagram displays connections for a few units. Each layer is fully connected with activation functions. The number of layers are denoted $l = 1, \ldots, L$, with $L = 4$. The activations at layer l are denoted $\mathbf{a}^l$. The input $\mathbf{x}$ forms activation layer 1, $\mathbf{a}^1$. We denote the number of units in the second and third layers as P and Q respectively.

2.4.4 Neural networks

In a feed-forward neural network, each layer consists of a number of units (or *neurons*) that compute a weighted linear combination, $\mathbf{z} = \mathbf{w}^\top \mathbf{x} + \mathbf{b}$, of the layer input, followed by an element-wise non-linearity, $\sigma(\mathbf{z})$. The model parameters consist of the combined set of neural network weights, $\mathbf{W}$, and biases, $\mathbf{b}$.

To be precise, we use w_{jk}^l to denote the weight for the connection from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer. We show an example four-layer configuration in Figure 2.1.

Using vector notation, we can describe the activation at layer l as a function of the previous layer as follows:

$$\mathbf{a}^l = \sigma(\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l), \tag{2.15}$$

where the activation vector $\mathbf{a}^l$ has components that are the activations a_j^l. The entries of the weight matrix $\mathbf{W}^l$ are the weights connecting to the l^{th} layer of neurons, that is, the entry in the j^{th} row and k^{th} column is W_{jk}^l. Similarly, for each layer l we define a bias vector, $\mathbf{b}^l$. Traditionally, choices for the activation function σ included the family of sigmoidal functions, with the most prominent member

being the logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.16}$$

We will consider a network with L layers. The network input is represented by $\mathbf{x} = \mathbf{a}^1$, and its output by $\mathbf{a}^L$, the activation of the output layer L. We denote the remaining intermediate layers from $l = 2, \ldots, L-1$ as *hidden* layers. The network computes a function of the input. The output $\mathbf{a}^L$ of this function is a prediction of one or more quantities of interest. We will use $\mathbf{y}$ to represent the desired output (target) corresponding to the network input $\mathbf{x}$.

During training, the parameters of all layers of the network are jointly optimised to make the output $\mathbf{a}^L$ approximate the desired output $\mathbf{y}$ as closely as possible. As a result, the hidden layers will learn to produce representations of the input data that are useful for the task at hand, and the output layer will learn to predict the desired output from these representations. We can write the prediction error in the output layer L as $\boldsymbol{\delta}^L = \mathbf{a}^L - \mathbf{y}$. To quantify how well we are achieving this goal, we define a cost function*, $C(\mathbf{W}, \mathbf{b})$.

The idea is to use gradient descent to find the weights w_k and biases b_l which minimise our cost function C. We can re-state the gradient descent update rule in terms of the weights and biases as components:

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k}, \tag{2.17}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l}, \tag{2.18}$$

where η is a hyperparameter, the learning rate. Given these quantities, via *back-propagation* it is possible to derive the expressions for the derivative of the cost function with respect to the parameters. We refer the reader to Nielsen (2015) for a detailed derivation. For the purpose of the thesis, we summarise the process of training a neural network with gradient descent over a dataset of size N as follows:

1. Input a set of N training examples, $\mathbf{X}$.

*Sometimes referred to as a loss or objective function.

2. For each training example $\mathbf{x}$: set the corresponding input activation, $\mathbf{a}^1$, and perform the following steps:

   a) Feedforward: for each $l = 2, 3, , \ldots, L$ compute $\mathbf{z}^{x,l} = \mathbf{w}^l \mathbf{a}^{x,l-1} + \mathbf{b}^l$ and $\mathbf{a}^{x,l} = \sigma(\mathbf{z}^{x,l})$.

   b) Compute the output error, $\boldsymbol{\delta}^{x,L}$, which is derived via the chain rule at the last layer as $\boldsymbol{\delta}^{x,L} = \nabla_a C_x \odot \sigma'(\mathbf{z}^{x,L})$.*

   c) Backpropagate the error: for each previous layer $l = L-1, L-2, \ldots, 2$ compute the error at layer l, $\boldsymbol{\delta}^{x,l}$.

3. Gradient descent: for each $l = L, L-1, \ldots, 2$ update the weights according to the rule $\mathbf{w}^l \to \mathbf{w}^l - \frac{\eta}{N} \sum_x \boldsymbol{\delta}^{x,l} (\mathbf{a}^{x,l-1})^\top$, and the biases according to the rule $\mathbf{b}^l \to \mathbf{b}^l - \frac{\eta}{N} \sum_x \boldsymbol{\delta}^{x,l}$.

The steps above capture the essence of training neural networks. Improvements can be made to the training procedure by choosing an appropriate cost function C_x, *regularising* the weights, and employing techniques for dealing with difficulties in training associated with gradients. For example, a common issue in neural networks trained with sigmoidal activation functions is the vanishing gradient problem. In essence, when backpropagating the error from the final layers to the initial layers, the gradient, which is calculated via the product of the errors, has a tendency to vanish due to the errors $\boldsymbol{\delta}$ becoming increasingly small (due to the derivative of the sigmoid function). Commonly, an alternative choice to the sigmoid function is the rectified linear function

$$\sigma(z) = \max(z, 0), \tag{2.19}$$

which leads to Rectified Linear Units (ReLUs) (Nair and Hinton, 2010). Increasing the weighted input, z, to a ReLU will never cause it to saturate, and so there is no corresponding learning slowdown from vanishing gradients. On the other hand, when the weighted input to a ReLU is negative, the gradient vanishes, and so the

*The $\odot$ operator (Hadamard product) denotes element-wise multiplication.

neuron stops learning entirely. These are just two of the many issues that make it non-trivial to understand when and why ReLU perform better than the sigmoid (Nielsen, 2015). For a deeper discussion on different activation functions, we refer the reader to Karlik and Olgac (2011).

An extension on regular gradient descent is to use stochastic gradient descent (SGD). In SGD, the true gradient of the cost function is approximated by a gradient at a single example:

$$w \to w - \eta \nabla C_i(w). \tag{2.20}$$

A compromise between computing the true gradient and the gradient at a single example is to compute the gradient against more than one training example (called a mini-batch) at each step. This can perform significantly better than SGD as in (2.20), because the code can make use of vectorisation libraries rather than computing each step separately. It may also result in smoother convergence, as the gradient computed at each step is averaged over a greater number of training examples.

Furthermore, for architectures used throughout the thesis, the optimiser *Adam* (Kingma and Ba, 2014) can be looked at as a combination of SGD with momentum and squared gradients to scale the learning rate (Kadambe and Srinivasan, 2006).

2.4.4.1 Convolutional neural networks

By definition, Convolutional Neural Networks (CNNs) are neural networks that use *convolution* in place of general matrix multiplication in at least one of their layers. They can be used when the input data exhibits some kind of topological structure, like the ordering of image pixels in a grid, or the temporal structure of an audio signal. In both of these media we expect there to be a strong correlation between neighbouring spatial values (or pixels). CNNs contain two types of layers with restricted connectivity: convolutional layers and pooling layers. A convolutional layer takes a stack of feature maps as input and convolves each of these with a set

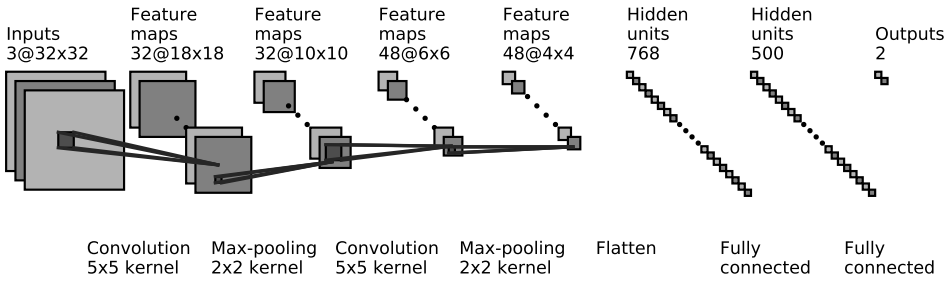


Figure 2.2: Example of a typical convolutional neural network used to detect two classes from a three-channel (RGB) image of size 32×32, parameterised as detailed in the figure.

of learnable filters to produce a stack of output feature maps. This is efficiently implemented by replacing the matrix-vector product $\mathbf{W}^l\mathbf{a}^{l-1}$ in Equation (2.15) with a sum of convolutions. The set of output activations from one feature map, $\mathbf{a}^l$ can be expressed as:

$$\mathbf{a}^l = \sigma(\mathbf{b} + \mathbf{W}^l * \mathbf{a}^{l-1}), \tag{2.21}$$

where the input feature map* is denoted by $\mathbf{a}^{l-1}$, and $*$ is the convolution operation, and $\mathbf{W}$ now refers to the learnable *filters*.

For example, in an RGB image with 3 channels, when we arrange the neurons in a three channel two-dimensional grid as shown in Figure 2.2, for the j, k^{th} hidden neuron in the feature map the output of the convolution with the filter $\mathbf{W} \in \mathbb{R}^{5x5}$ is:

$$\sigma\left(b + \sum_{l=1}^{5}\sum_{m=1}^{5} w_{l,m}a_{j+l,k+m}\right), \tag{2.22}$$

where b is the shared value for the bias, $w_{l,m}$ is a 5×5 array of shared weights, and $a_{x,y}$ denotes the input activation at position (x, y).

By replacing the matrix product with a sum of convolutions, the connectivity of the layer is effectively restricted to take advantage of the input structure and to reduce the number of parameters. Each unit is only connected to a local subset of the units in the layer below, and each unit is replicated across the entire input.

*The input layer activation $\mathbf{a}^1$ is the original data point $\mathbf{x}$.

This reduction in parameters can drastically improve generalisation performance and make the model scale to larger input dimensionalities. Because convolutional layers are only able to model local correlations in the input, the dimensionality of the feature maps is often reduced between convolutional layers by inserting pooling layers. This allows higher layers to model correlations across a larger part of the input, albeit with a lower resolution. A pooling layer reduces the dimensionality of a feature map by computing some aggregation function (typically the maximum or the mean) across small local regions of the input (Boureau et al., 2010).

For example, we can think of max-pooling as a way for the network to ask whether a given feature is found anywhere in a region of the image. It then throws away the exact positional information. The intuition is that once a feature has been found, its exact location is not as important as its rough location relative to other features. A big benefit is that there are many fewer pooled features, and so this helps reduce the number of parameters needed in later layers. The pooling also makes the model invariant to small translations of the input, which is a desirable property for modelling images and many other types of data.

By alternating convolutional and pooling layers, higher layers in the network see a progressively more coarse representation of the input. As a result, these layers are able to model higher-level abstractions more easily because each unit is able to see a larger part of the input.* CNNs constitute the state of the art in many computer vision problems. Since their effectiveness for large-scale image classification was demonstrated, they have been ubiquitous in computer vision and audio analysis research (Krizhevsky et al., 2012; Razavian et al., 2014; Dieleman and Schrauwen, 2014; Hershey et al., 2017)

Having given an overview of the fundamentals, as well as a range of models and their applicability, we narrow down our focus to the domain of audio data analysis, as subsets of time series and detection as defined in Sections 2.2.1 and 2.2.4.

*This concept can be extended further to *dilated* convolutions (van den Oord et al., 2016).

2.5 Audio data analysis

Audio signals are a proxy representation of sound. Sound is a vibration that propagates as an acoustic wave, through a transmission medium such as a gas, liquid or solid. Audio signals are continuous in both time and amplitude. To store them in digital form as time series, however, the signals need to be quantised in both of these dimensions. This is referred to as Pulse-Code Modulation (PCM). In a PCM stream, the amplitude of the analogue signal is sampled regularly at uniform intervals, and each sample is quantised to the nearest value within a range of digital steps. The process of sampling, given a sample rate at twice the maximum frequency of interest* fully preserves (*lossless transform*) a band-limited signal. However, quantising to discrete values in amplitude is a *lossy* transform (creates an approximation of the original).

The commonly used WAVE file format (`*.wav`) is a form of storing the unmodified sequence of PCM numbers. Other audio file formats, such as MP3 (`*.mp3`), or OGG vorbis (`*.ogg`) (Moffitt, 2001) compress the sequences with lossy schemes that remove information the human ear is insensitive to, to reduce the amount of required storage space. We show an example mono WAVE audio signal of a mosquito in flight for five seconds, which is sampled at 44.1 kHz in Figure 2.3. The following description of feature representations in audio will refer back to this figure for illustrations of each transformation.

In the field of audio signal analysis, *mid-level* representations are often extracted from digital audio signals and operated on instead (Muller et al., 2011). The representations typically match more closely how humans perceive sound – as varying patterns of frequencies over time. Mid-level representations derive their name by holding the middle ground between low-level representations (i.e. raw audio signals) and high-level symbolic representations of music such as musical notation. The most commonly used mid-level representations are time-frequency

*Plus a buffer for imperfect anti-aliasing filters.

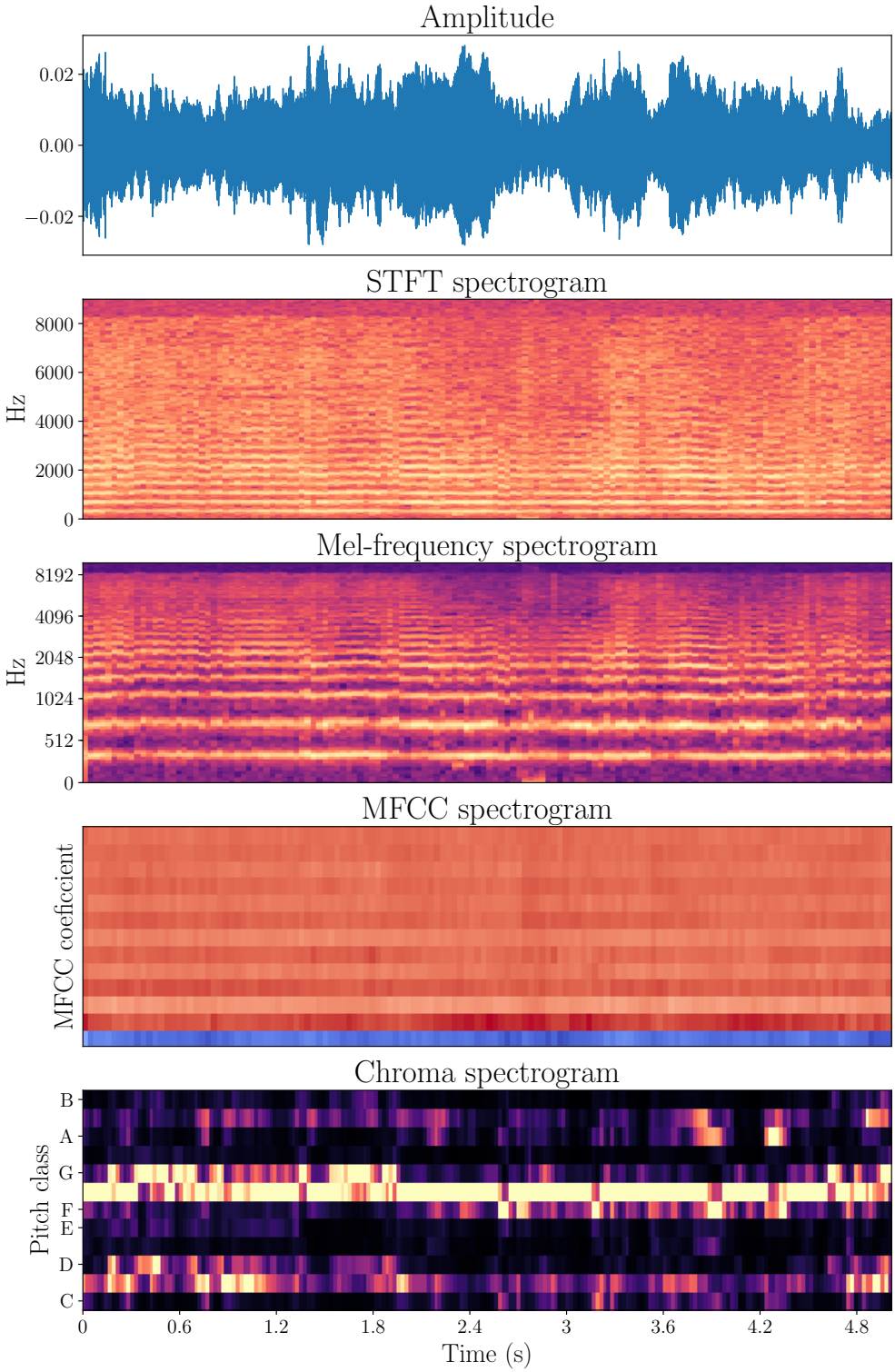


Figure 2.3: Mosquito WAVE audio file (top), with feature representations of increasing saliency from top to bottom.

representations. Instead of describing the amplitude of the signal as it varies over time, they describe the amplitudes or energies of the signal in various frequency bands, as they vary over time on a much coarser timescale (Zölzer, 2011). In effect, these representations trade temporal resolution for frequency resolution.*

Many commonly used time-frequency representations are based on *spectrograms*. A spectrogram of a signal can be obtained by computing the short-time Fourier transform (STFT) of short windows that usually overlap partially. From these transformed windows, power spectra can be obtained that indicate the energy in various frequency bands within each window. Each of these constitutes a frame. The successive frames are concatenated into a matrix to form the spectrogram. To illustrate the STFT and provide insight, we begin by describing the Fourier transform (FT)†, which is the fundamental transformation upon which the majority of mid-level representations are formed:

$$\mathrm{FT}\{x(t)\}(\omega) \equiv X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} \, \mathrm{d}t. \tag{2.23}$$

The FT decomposes a function (often a function of time, or a signal) into its constituent frequencies. The term Fourier Transform refers to both the frequency domain *representation* and the mathematical *operation* that associates the frequency domain representation to a function of time. The FT of a function of time is itself a complex-valued function of frequency, whose magnitude (modulus) represents the amount of that frequency present in the original function, and whose argument is the phase offset of the basic sinusoid in that frequency.

Because we are interested in how these frequency components vary over time, the function to be transformed $x(t)$ is multiplied by a window function which is nonzero for only a short period of time. The FT (a one-dimensional function) of the resulting signal is taken as the window is slid along the time axis, resulting in a

*This is a direct result of the Heisenberg uncertainty principle, which states that the product of time and frequency resolution is fixed, which we return to in Chapter 4.

†We give all definitions in continuous time, but note that the integral is replaced with a finite sum in discrete time for real signals.

two-dimensional representation of the signal. Mathematically, this is written as:

$$\mathrm{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-i\omega t}\,\mathrm{d}t, \tag{2.24}$$

where $w(\tau)$ is the window function, commonly a Hann window or Gaussian window centred around zero. $X(\tau,\omega)$ is essentially the FT of $x(t)w(t-\tau)$, a complex function representing the phase and magnitude of the signal over time and frequency.

We showcase the STFT, with 1024 frequency bins, with the Hann window function, of our WAVE mosquito recording in Figure 2.3.

Spectrograms are often post-processed further to better match the properties of human perception. The frequencies in the spectrogram are often converted to a logarithmic scale, to match our logarithmic perception of loudness at different pitches (Varshney and Sun, 2013). One such example is the mel spectrogram, which is commonly used in speech processing. The mel scale* is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The equation for mapping the frequency, f, in Hz to the mel scale, m, is given as

$$m = 2595\log_{10}\left(1+\frac{f}{700}\right). \tag{2.25}$$

We show a mel-encoded spectrogram of our mosquito in flight in Figure 2.3. The tiling of the frequencies can be contrasted to that of the STFT, which we plot in a linear frequency scale for clearer comparison between the two. The mel scale assigns greater frequency resolution to low-frequency events, and higher time resolution to high-frequency events. A more general form, from which Fourier-related windowed features can be derived, can be expressed with wavelet transforms. With wavelets we have the flexibility to tile the time-frequency plane arbitrarily, and are not restricted to a rectangular grid. We explore, contrast, and compare wavelet features in more depth in Chapter 4.

The mel spectrogram can then be further de-correlated by taking a discrete cosine transform (DCT) to form mel-frequency cepstral coefficients (MFCCs). The DCT

*The name mel is derived from the word melody to indicate that the scale is based on pitch comparisons.

de-correlates the energies which means diagonal covariance matrices can be used to model the features in e.g. a Hidden Markov Model (HMM) classifier. Furthermore, commonly only a subset of DCT coefficients are kept. This is because the higher DCT coefficients represent fast changes in the filterbank energies, which often are attributed to noise in the higher energy spectra.

Further specialised high-level features exist, such as the chroma representation which partitions frequencies into twelve categories, whose tuning approximates to the equal-tempered scale. One main property of chroma features is that they capture harmonic and melodic characteristics of music, while being robust to changes in timbre and instrumentation (Müller, 2007). These would be particularly useful for classifying music pieces written in that scale into notation. For the example mosquito signal in Figure 2.3, it can be shown that the mosquito holds a near constant pitch at F# throughout its flight path, with slight warbles in pitch.*

We note a common theme amongst levels of feature representation. In general, the less capable the classifier is at forming latent representations, the more salient the level of features tend to be used. For example, this applies to MFCCs, which have been paired with classifiers that make strong assumptions about the underlying data (such as de-correlated features in the case of the HMM). Furthermore, it has been common practice to stack together a set of high-level feature representations into feature vectors, fine-tuned to specific applications. Humphrey et al. (2013) and Espi et al. (2015) have attributed the stagnation to performance in the field of audio analysis to this overly hand-crafted approach.

In contrast, more general Fourier or wavelet features pair well with models such as convolutional neural networks, which discover hierarchical feature representations.

As a final note, there has been some limited success in working directly with raw time series to discover spectral-like representations (Dieleman and Schrauwen, 2014). In our work, due to the difficulty of obtaining sufficient labelled training data for

*The fundamental frequency of the mosquito is approximately 370 Hz which corresponds to the fourth overtone of F#0 = 23.12 Hz, F#4 = 369.99 Hz.

full end-to-end learning, empirically we observed performance degradation when not utilising mid-level representations. This thesis therefore does not focus on learning from raw time series directly. We note, however, that further progress made in this area may lead to raw time series becoming the input representation of choice in the future.

2.6 Chapter summary

In this chapter we have provided an overview of machine learning approaches for signal detection. Section 2.2.2 defines our understanding of signals. Section 2.2.4 offers an introduction to time series, and discusses how classification can be performed based on regressing from a latent space of real numbers to class probabilities. We have covered the paradigms for training supervised classifiers, and common approaches to modelling that recur in later chapters of the thesis. The methods include decision trees and random forests (Section 2.4.1), the support vector machine (Section 2.4.3), and neural networks (Section 2.4.4).

In Section 2.5 we covered the foundational principles related to the signal processing aspects of audio. Additionally, we have reviewed popular feature representations that have been ubiquitous in signal processing applications. This chapter has therefore provided us with the prerequisite knowledge to apply machine learning at the intersection of signal processing.

Chapter 3

The HumBug project

3.1 Overview

In this chapter we provide an overview to the background and context of the project that motivated the core research of this thesis. We begin with prior research that was used as the foundation for further work in Section 3.2. We describe the timeline of the project in Section 3.3. In this timeline we explore the evolution of the project, breaking down the work into key events in data collection, deployment applications, and corresponding research contributions. This chapter provides motivating context for the remainder of the thesis and indicates the reasoning behind the research directions taken in the later chapters.

3.2 Acoustic detection prior work

In its infancy in 2015, without data available specific to the HumBug project, mosquito detection methods were operating on what was known about the acoustics of mosquitoes in the literature. We refer the reader to Warren et al. (2009) which gives a detailed breakdown of how the sound plays an important role in mosquito behavioural patterns, such as location and mating. Importantly, the wingbeat produces a tone that is proportional the wingbeat frequency, which forms audible sound with harmonic content (Clements, 1999). Table 3.1 shows that this

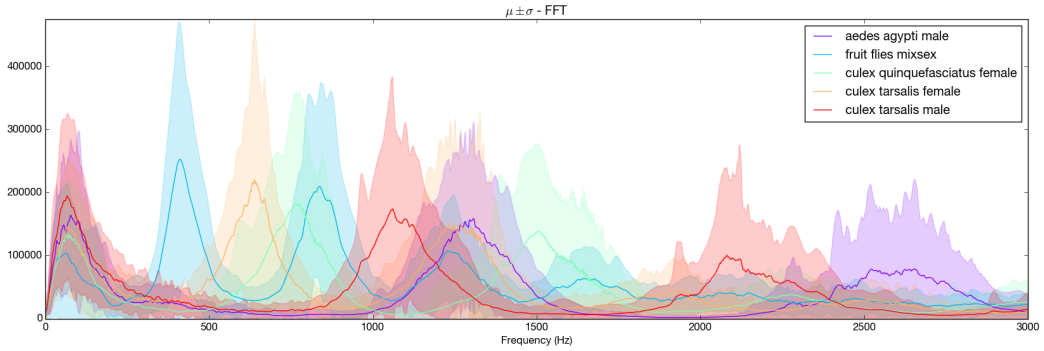


Figure 3.1: FFT of 5 different classes of mosquitoes (4 species) (Chen et al., 2014).

frequency can vary greatly between species and sex, giving rise to the opportunity to discriminate based on the emitted sound. Additionally, Figure 3.1 shows the spectral components of the wingbeat of 500 different specimens belonging to the *culex quinquefasciatus* (female), *aedes agypti* (male), *culex tarsalis* (male and female) mosquito species, as well as the spectrum of fruit flies for reference. It can be seen that despite both fruit flies and the *aedes agypti* male producing very similar sound around 1250 Hz, the two entities do not overlap elsewhere over the spectral range of the plot. Thus, from the full frequency spectrum it is possible to distinguish the two clearly. It is noteworthy that the wingbeat frequency of *aedes agypti* given in Table 3.1 and the apparent fundamental frequency visible in Figure 3.1 do not correspond. This can occur due to the fundamental frequency being inaudible when measured, and what we see instead in Figure 3.1 is the second harmonic at double the wingbeat frequency (approximately $600 \times 2 = 1200$ Hz). The microphone on the mobile is a sensor that measures the changes in the air pressure. If a mosquito flies right underneath or above the microphone, then the microphone will be able to pick up the fundamental frequency clearly. However, if the microphone is placed sideways relative to the mosquito wings, then it will only be able to detect the pressure variations twice as fast as the wingbeat frequency, leading to a high peak at the second harmonic (Belton, 1986).

We note that similar fundamental frequency characteristics (e.g. *aedes agypti* vs *anopheles subpictus* in Table 3.1) could still lead to a ratio of harmonics that differs

	Wingbeat frequency (Hz)	
Species	Male	Female
Aedes agypti	$557 - 600$	$414 - 453$
Aedes diamtaeus	538 ± 20	330 ± 13
Anopheles subpictus	$520 - 580$	$330 - 385$
Anopheles gambiae	$660 - 900$	$420 - 600$

Table 3.1: A range of mosquito wingbeat frequencies reproduced from Clements (1999).

between these mosquito species. Based on the findings of Raman et al. (2007), simple approaches, involving sensing the fundamental frequency (first harmonic) and second harmonic, were capable of detecting insects, but generated large numbers of false positives because of other ambient sounds. In contrast, combining information from the first four harmonics, from the inter-harmonic regions, and from the sound envelope, reduced false positives greatly.

We accept that in practice the exact ratio of harmonics would be difficult to identify due to the low SNR emitted by a mosquito's flight tone (as evidenced by the rate of disagreement amongst experts in Chapter 4 when marking onset and offset times). However, combining the difference in observed fundamental frequencies, harmonics, and potentially harmonic ratios, with additional variables such as the geographical location, gives us sufficient prior evidence that it may be possible to recognise and distinguish mosquito species from data-driven approaches.

3.3 Timeline

The HumBug project has presented multiple challenges, both in data acquisition and in the development and deployment of acoustic detection algorithms. We will now look at key events of the HumBug project, as well as explain their relevance within the literature. Figure 3.2 provides an overview of this timeline.

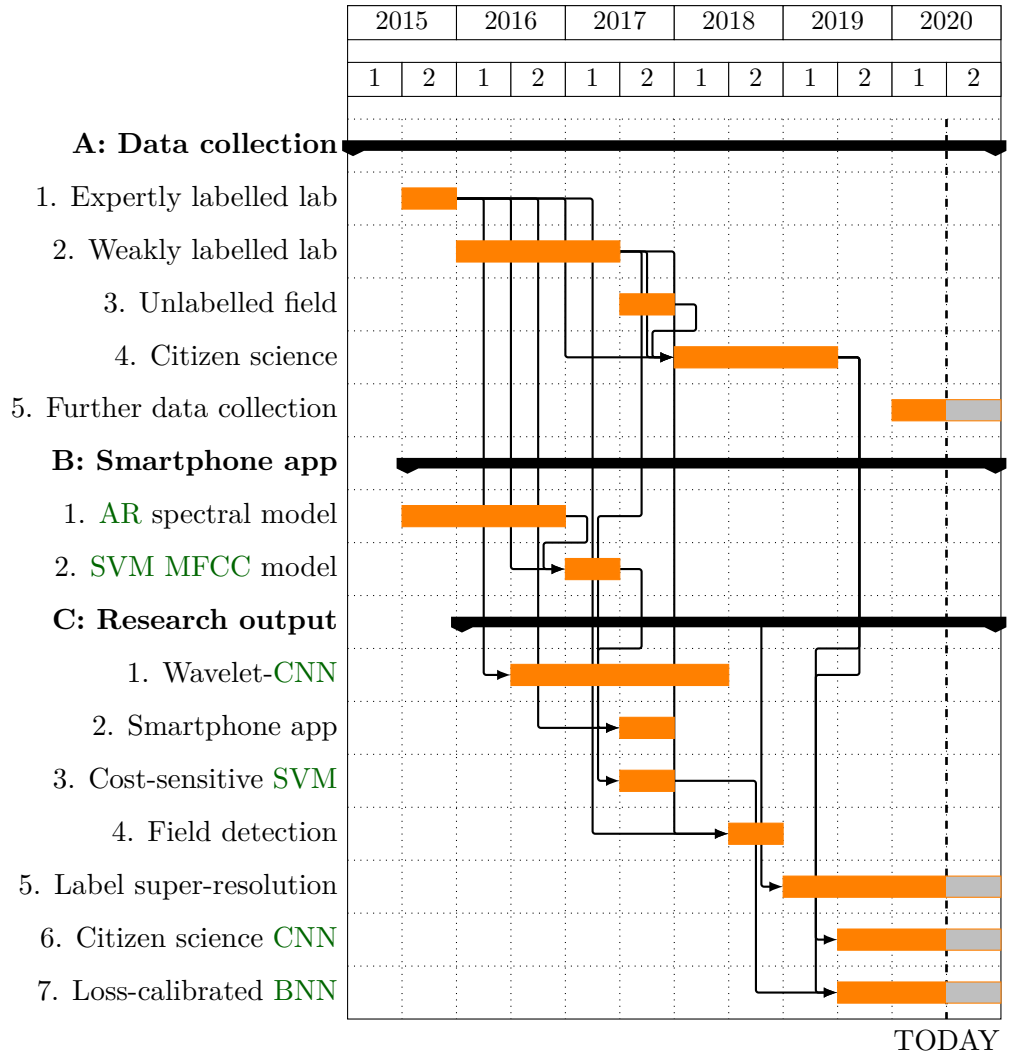


Figure 3.2: Timeline of the HumBug project organised by **A: Data collection**, **B: Smartphone app** development, and **C: Research output** publications. The endpoints are defined as reaching a stage where dependent work may be carried out, and not necessarily where development for each specific element ends.

3.3.1 A: Data collection

1. Expertly labelled lab data

In the early stages of the project in 2015, one of the first studies undertaken was the collection and labelling of a small dataset of mosquitoes which were cultured specifically in cages, and maintained under controlled conditions (Figure 3.2: **A**1). The data was recorded with low-cost smartphone microphones, which were intended to be used in conjunction with the app in development. The labels were initially created by humans by listening and utilising the spectrogram as a visual aid to more accurately determine onset and offset times. A range of label qualities was supplied, based on the time commitment available to the human experts. The resultant labelled dataset consisted of approximately one hour of recordings.

2. Weakly labelled lab data

The weakly labelled data (Figure 3.2: **A**2) refers to audio recordings which were made available through the project collaborators. The dataset contains significant volumes of caged mosquitoes, that have been supplied by the Center for Disease Control and Prevention (CDC), and the United States Army Medical Research Unit, Kenya (USAMRU-K). Most recordings however contain at most one species tag per length of each individual recording – of which the majority is noise. Our approach to making classifications with this data became more nuanced as the project progressed through its stages. We describe the first steps made towards simplifying this problem when discussing dataset **A**4.

3. Unlabelled field data

This dataset represents the first recordings that are made with mosquitoes captured in the wild in sites in Thailand. This data was collected with the help of the second version of the smartphone app, and contains no tags with onset or offset times.

For a better idea of the information content, we subject this set to citizen science labelling to form part of a larger dataset, **A**4.

4. Citizen science data

With this influx of further data (Figure 3.2: **A**2, **A**3.), it was clear that the scale at which labelling the audio was required was not sustainable to individual human experts. This led to the organisation of a citizen science project on Zooniverse, a popular crowdsourcing platform (Simpson et al., 2014). The initial setup was first published by Li et al. (2017b) (Figure 3.2: **C**2). Following the deployment over time, gathered votes were released alongside the data in work outlined in Section 5.5 (Figure 3.2: **C**6).

5. Further data collection

Finally, further data collection is underway through the concerted efforts of the new collaboration established in 2020 with Tanzania and the Democratic Republic of Congo.

3.3.2 B: Smartphone app

1. Autoregressive spectral method

With prior knowledge of the acoustics of mosquitoes (as laid out in Section 3.2), the research team deployed the first version of the smartphone app (Figure 3.2: **B**1). The app used an Autoregressive (AR) spectral smoothing algorithm (Roberts, 2010), combined with a peak-finding algorithm to re-cast detection as a binary classification when searching for harmonic structure in a pre-specified frequency range (Lin and Zilli, 2015). This method relied on searching for a prior range of frequencies which the user believed to be either the fundamental frequency, or its second harmonic, based on the known physics of mosquitoes, such as reported

by Clements (1999). While the algorithm is robust to noise through the spectral smoothing, another mosquito that shares a similar frequency (or other ambient sound with a fundamental frequency in the same range), but with a different ratio of amplitudes of the harmonic peaks, would not be distinguishable. This is because the peak-finding algorithm identifies subsequent harmonics within a range of frequencies due to the potential inaccuracy of exactly determining the fundamental. This provides no mechanism for discriminating between the relative ratios of harmonic peaks and inter-peaks. This drawback called upon more sophisticated data-driven approaches.

2. SVM MFCC model

To combat this drawback we utilised the expert-labelled dataset **A**1. The relatively scarce dataset inspired the use of data-efficient learning methods for the development of the second revision of the smartphone app (Figure 3.2: **B**2 (Li et al., 2017b)). This improved algorithm utilised the prediction stage of an SVM, which was trained offline on 13 MFCCs of the available mosquito data at this point. More details regarding the design choices and parameter specifics are given in Section 5.2.

3.3.3 C: Research output

1. Bioacoustic detection with wavelet-conditioned convolutional neural networks

In Chapter 4 we utilise the dataset **A**1 to trial a range of traditional established machine learning methods against deep learning architectures more recently employed. In this data domain, where we have a low number of training examples, hand-crafted approaches tended to dominate. However, by utilising windowing appropriate to the dynamics of the signal and the size of the dataset, we show how it is possible for deep learning approaches to be effective in spite of this data scarcity. We perform this by utilising an often overlooked feature transform – the wavelet

transform. We give details of the dataset and algorithms in Chapter 4 (Kiskin et al., 2018). The best-performing algorithm, outperforming human experts with access to the same recordings, was shown to be the wavelet-CNN. We furthermore show the ability to generalise to other datasets, an advantage unique to paradigms that do not rely on explicitly hand-crafted feature representations.

2. Mosquito detection with low-cost smartphones: Data acquisition for malaria research

From the work of **C**1, the best-performing computationally efficient algorithm was selected for the revision of the smartphone app (**B**2). This spawned our publication co-authored with Li et al. (2017b) (Figure 3.2: **C**2), which outlines Section 5.2. We detail the technology that goes into the data collection and tagging process, whose output is crucial to the success of the concentrated research efforts to understand mosquito prevalence for the purpose of malaria prevention. Additionally, we show that it is possible to distinguish between seven species of mosquitoes from the recordings available at this stage of the project.

3. Cost-sensitive detection with variational autoencoders for environmental acoustic sensing

In Section 5.4 (Figure 3.2: **C**3), we consider the effects of false positives and negatives, and show how an SVM combined with a variational autoencoder can be used in this framework. We wish to employ algorithms that place a different relative importance on errors made in classification. This stems from the requirements of a detection system, which may work in two modes. In the *passive* mode, the system listens for a potential positive candidate. On detection, the system records the audio and uploads to a central database. In this mode, in order not to excessively disturb the user and conserve hardware resources, we wish only to trigger a detection with a high confidence event, thereby promoting a low false positive rate. We hence

penalise false positives more strongly than false negatives. In the second (*active*) mode, we are more concerned with not missing a disease-bearing species, especially if deployed in an area at a critical risk of malaria. We therefore wish to maximise the recall of positive events, at the expense of incurring a higher false positive rate. To target the smartphone app, this led to the utilisation of cost-sensitive SVMs, which were further combined with a variational autoencoder to reduce the feature dimensions (Li et al., 2017a). The variational autoencoder, which was trained offline, further optimised the computational efficiency of the SVM-MFCC smartphone app by creating a latent space of four features. This reduces the demand on the prediction of the SVM, which is used in real-time to record and store data on detection. This was deemed of critical importance for the use of a low-power device such as a smartphone.

4. Fast mosquito acoustic detection with field cup recordings: an initial investigation

With a greater availability of data, we were able to conduct further experiments, resulting in the research output of Section 5.3 (Li et al., 2018) (Figure 3.2: **C**4). In this section, we show that it is possible to distinguish 6 different species of mosquitoes from data recorded in the field in Thailand. This was an important step in the feasibility assessment of the project, and gave the team confidence moving forward. Similarly to the wavelet-CNN, we present a less hand-crafted approach to make predictions on the log-mel spectrogram with CNNs. Considerations behind the design choices are given in detail in Section 5.3.

5. Super-resolution of time-series labels for bootstrapped event detection

Chapter 7 describes ongoing research which addresses further challenges with the availability of data at a multitude of time resolutions, which is both expertly and

weakly labelled. This led to the development of a framework which aims to extract maximal information for the purpose of maximising overall classification accuracy. This made use of a two-stage Bayesian framework. In this framework, we use expertly labelled data as prior information to create a probability density estimate of a feature space. With Bayes' rule, we combine weakly labelled data to update our label belief. These new labels are then used to train a supervised classifier (Figure 3.2: **C**5) (Kiskin et al., 2019).

6. HumBug Zooniverse: a crowdsourced acoustic mosquito dataset

With the influx of further data (**A**3), it was clear that the scale at which labelling the audio was required was not sustainable to individual human experts. Following the deployment of the citizen science project over time, gathered votes were released alongside the data in work outlined in Chapter 5.5 (Figure 3.2: **C**6). Furthermore, we supplied a baseline CNN which showed that there was sufficient information content to reliably predict mosquito events. We also briefly touch upon correcting for the heavy class imbalance present in that labelled dataset by utilising a weighted cross-entropy approach (Kiskin et al., 2019, 2020).

7. Loss-calibrated Bayesian neural networks for noisy acoustic mosquito detection

We then address this trade-off in a more Bayesian manner through the use of a utility function with Bayesian decision theory. Further difficulties with the dataset presented themselves in the form of a large imbalance in the distribution of the classes, a high uncertainty in the inputs, and furthermore the absence of a ground truth. To leverage the best of Bayesian probability frameworks and the discriminative power of CNNs, we develop and apply Loss-Calibrated Bayesian Neural Networks (LCBNNs) (Figure 3.2: **C**7) in Chapter 6 (Kiskin et al., 2020).

3.4 Chapter summary

In this chapter we have broken down the overarching goals of this interdisciplinary collaboration to combat malaria through improved malaria vector modelling. We have described the project according to the timeline according to three categories: data collection, versions of the smartphone app, and the resulting research output. For each category, we have described the development in chronological order. As we gained access to a larger quantity of data, correspondingly the complexity of the solution increases. In order to allow the use of more complex models, we have described the technology we use to collect the data, and how this presented new challenges that we have addressed through the innovative application of machine learning. The data in turn has been used to both improve the practical smartphone app, as well as answer emerging research challenges.

In the following chapter we begin working with a small expert-labelled dataset, which poses challenges with generalisation, especially with the use of data-intensive algorithms. We pose the mosquito detection problem in the more broad framework of time-series analysis in scarce data environments.

Chapter 4

Bioacoustic classification with wavelet-conditioned neural networks

4.1 Overview

Many real-world time-series analysis problems are characterised by low signal-to-noise ratios and compounded by scarce data. Solutions to these types of problems often rely on hand-crafted features extracted in the time or frequency domain. Recent high-profile advances in deep learning have improved performance across many application domains, however they typically rely on large datasets that may not always be available. This chapter presents an application of deep learning for acoustic event detection in a challenging, data-scarce, real-world problem. We show that CNNs, operating on wavelet transformations of audio recordings demonstrate superior performance over conventional classifiers that utilise hand-crafted features. Our key result is that wavelet transformations offer a clear benefit over the more commonly used short-time Fourier transforms. Furthermore, we show that features, hand-crafted for a particular dataset, do not generalise well to other datasets. Conversely, CNNs trained on generic features are able to achieve comparable results

across multiple datasets, along with outperforming human labellers. This chapter presents our results on the application of both detecting the presence of mosquitoes and the classification of bird species.

The work herein is published in the *Springer Neural Computing and Applications Special Issue on Deep Learning for Music and Audio*, (Kiskin et al., 2018), which expands on previous work that was specific to mosquito detection within HumBug (Kiskin et al., 2017).

4.2 Introduction

The timely and accurate detection of animals, birds and insects is of critical importance for conservation, ecology and epidemiology. We consider the effective analysis of the natural soundscape as a constituent component of this analysis. In this chapter we focus on bioacoustic classification, with a particular emphasis on mosquito detection. As part of showcasing the methods developed for this application, we describe how they can also, with minimal alteration, offer robust results in other bioacoustic classification domains.

Mosquitoes are responsible for hundreds of thousands of deaths every year due to their capacity to vector* lethal parasites and viruses, which cause diseases such as malaria, lymphatic filariasis, zika, dengue and yellow fever (World Health Organization et al., 2016, 2014). Their ability to transmit diseases has been widely known for over a hundred years, and several practices have been put in place to mitigate their impact on human life. Examples of these include insecticide-treated mosquito nets (Lengeler, 2004; Bhatt et al., 2015) and insect sterilisation techniques (Alphey et al., 2010). However, further progress in the battle against mosquito-vectored disease requires a more accurate identification of species and their precise location – not all mosquitoes are vectors of disease, and some non-vectors are

*In epidemiology, a disease vector is any agent which carries and transmits an infectious pathogen into another living organism (John, 2001).

morphologically identical to highly effective vector species. Current surveys rely either on human-landing catches or on less effective light traps. In part this is due to the lack of cheap, yet accurate, surveillance sensors that can aid mosquito detection. Acoustic monitoring of mosquitoes proves compelling, as the insects produce a sound both as a by-product of their flight and as a means for communication and mating. Detecting and recognising this sound is an effective method to locate the presence of mosquitoes and even offers the potential to categorise by species. Nonetheless, automated mosquito detection presents a fundamental signal processing challenge, namely the detection of a weak signal embedded in noise. Current detection mechanisms rely heavily on domain knowledge, such as tuning models to likely fundamental frequency and harmonics, and often extensive hand-crafting of features, frequently similar to traditional speech representation methods. Over the last decade there have been increasingly impressive performance gains achieved by the paradigm shift to deep learning, including bioacoustics (Joly et al., 2016). An opportunity hence emerges to exploit and expand upon these advances to tackle our application problem.

Deep learning approaches, however, tend to be effective only once a critical number of training samples has been reached (Chen et al., 2014). Consequently, data-scarce problems are not well suited to this paradigm. As with many other domains, the task of data labelling is expensive in both time requirement for hand labelling and associated ambiguity – namely that multiple human experts will not be perfectly concordant in their labels. Furthermore, recordings of free-flying mosquitoes in realistic environments are scarce (Mukundarajan et al., 2017) and hardly ever labelled.

This chapter presents a novel approach for classifying events from acoustic data using scarce training data. Our approach is based on a convolutional neural network classifier conditioned on wavelet representations of the raw data. By exploiting the high sample rates of audio recordings we are able to create sufficient training data for deep learning to remain highly effective. The network architecture and associated

hyperparameters are however still strongly influenced by constraints in dataset size. We compare our methods to well-established classifiers, trained on both hand-crafted features and the STFT, as well as human-made labels of mosquito audio recordings. We show that wavelet-conditioned CNN classifications are consistently made more accurately and confidently than on the STFT. The majority of our algorithms are able to more reliably detect mosquitoes (with accuracy above 90 %) than human labellers, where only 70 % of labels are in full agreement amongst four labellers.

Furthermore, without additional hyperparameter tuning we demonstrate that our approach scales well to different data domains, a transfer that traditional hand-crafted features or classifiers struggle to make. Highlighting the generic nature of the solution we propose, we show that the CNN is also able to extract feature representations that allow it to distinguish between nine species of birds with reliably high accuracy (over 90 %), similarly from very little data.

The remainder of this chapter is structured as follows. Section 4.3 addresses related work, explaining the motivation and benefits of our approach. Section 4.4 details the method we adopt, providing insight into the relative strengths of wavelet transforms. Section 4.5 describes the experimental setup. Section 4.6 highlights the value of the method. We visualise and interpret the predictions made by our algorithm on unseen data in Section 4.7 to help reveal informative features learned from the representations and verify the method. Finally, we suggest further work and conclude in Section 4.8.

4.3 Related work

The use of artificial neural networks in acoustic detection and classification of species dates back to at least the beginning of the century, with the first approaches addressing the identification of bat echolocation calls (Parsons and Jones, 2000). Both manual and algorithmic techniques have subsequently been used to identify insects (Chesmore and Ohya, 2004; Zilli et al., 2014), elephants (Clemins et al., 2005),

delphinids (Oswald et al., 2003), and other animals. The benefits of leveraging the sound animals produce—both actively as communication mechanisms and passively as a results of their movement—is clear: animals themselves use sound to identify prey, predators, and mates. Sound can therefore be used to locate individuals for biodiversity monitoring, pest control, identification of endangered species and more.

This section will therefore briefly review the use of machine learning approaches in bioacoustics. We describe the traditional feature and classification approaches to acoustic signal detection. In contrast, we also present the benefit of feature extraction methods inherent to current deep learning approaches. Finally, we narrow our focus down to the often overlooked wavelet transform, which offers significant performance gains in our pipeline.

4.3.1 Applications

The employment of artificial neural networks has proven successful for over a decade. Chesmore and Ohya (2004) used a neural network classifier to discriminate four species of grasshopper recorded in northern England, with accuracy surpassing 70 %. Other classification methods include Gaussian mixture models (Potamitis et al., 2007; Pinhas et al., 2008) and HMMs (Leqing and Zhen, 2010; Zilli et al., 2014), applied to a variety of different features extracted from recordings of singing insects. The work of Chen et al. (2014) attributes the difficulty of automated insect detection to the sole use of acoustic devices, which the authors claim are often not capable of producing a signal sufficiently clean to be classified correctly. In their work they replace microphones with optical sensors, recording mosquito wingbeat through a laser beam hitting a phototransistor array – an extension of the method proposed by Moore et al. (1986). In a real-world setting, the resultant signals have a higher signal-to-noise ratio than those recorded acoustically. We regard these approaches and acoustic sensors as complementary, rather than competitors, and note that approaches which work well for acoustic detection can also be used to

perform detection in other datasets, including optically-sensed data, as well as in other bioacoustic problems.

Whichever technique is used to record a mosquito wingbeat frequency, the need arises to be able to identify the insect's flight in a (more or less) noisy recording. The following section therefore reviews recent achievements in feature representation and learning, in the broad context of practical acoustic signal classification.

4.3.2 Feature representation and learning

The process of automatically detecting an acoustic signal in noise typically consists of an initial preprocessing stage, which involves cleaning and denoising the signal itself, followed by a feature extraction process, in which the signal is transformed into a format suitable for a classifier, followed by the final classification stage. Historically, audio feature extraction in signal processing employed domain knowledge and intricate understanding of digital signal theory (Humphrey et al., 2013), leading to hand-crafted feature representations.

Many of these representations often recur in the literature. A powerful, though often overlooked, technique is the wavelet transform, which has the ability to represent multiple time-frequency resolutions (Akay, 1998, Ch. 9). An instantiation with a fixed time-frequency resolution thereof is the STFT. MFCCs create lower-dimensional representations by taking the STFT, applying a non-linear transform (the logarithm), pooling, and a final affine transform. A further example is presented by Linear Prediction Cepstral Coefficientss (LPCCs), which pre-emphasise low-frequency resolution, and thereafter undergo linear predictive and cepstral analysis (Ai et al., 2012).

Detection methods have fed simple STFT representations to standard classifiers (Potamitis, 2014), but more frequently higher level features and feature combinations are used, applying dimensionality reduction to combat the curse of dimensionality (Lee et al., 2009). High-level features (e.g. MFCCs and LPCCs) were originally

developed for specific applications, such as speech recognition, but have since been used in several audio domains (Li et al., 2001). Humphrey et al. (2013) argue that using features specifically developed for a prior application is unsustainable and has contributed to a stagnation in the field of audio event recognition.

On the contrary, the deep learning approach usually consists of applying a simple, general transform to the input data, and allowing the network to both learn a feature representation and perform classification. This enables the models to learn salient, hierarchical features from raw data. The automated deep learning approach has recently featured prominently in the machine learning literature, showing impressive results in a variety of application domains, such as computer vision (Krizhevsky et al., 2012) and speech recognition (Lee et al., 2009). However, deep learning models such as convolutional and recurrent neural networks are known to have a large number of parameters and hence typically require large data and hardware resources. Despite their success, these techniques have only recently received more attention in the time-series signal processing literature.

A prominent example of this shift in methodology is the BirdCLEF bird recognition challenge. The challenge consists of the classification of bird songs and calls into up to 1500 bird species from tens of thousands of crowdsourced recordings. The introduction of deep learning has brought drastic improvements in mean average precision scores. The best mean average precision score of 2014 was 0.45 (Goëau et al., 2015), which was improved to 0.69 the following year when deep learning was applied, outperforming the closest scoring hand-crafted method by 19 % (Joly et al., 2016). The impressive performance gain came from the utilisation of well-established convolutional neural network practice from image recognition. By transforming the signals into STFT spectrogram format, the input is represented by 2D matrices, which are used as training data. The following year saw a further jump to 0.71 (Sevilla et al., 2017) by utilising transfer learning of the Inception-v4 deep convolutional neural network which was highly successful in ImageNet. Alongside this example, the most widely used means to transform the input signals is the

STFT (Sainath et al., 2015; Gwardys and Grzywczak, 2014; Potamitis, 2016).

An alternative feature transformation can be obtained with wavelets. Gaining popularity in the late '90s, wavelets have been applied successfully to efficient image compression (Christopoulos et al., 2000, JPEG 2000), de-noising (Donoho, 1995), and have shown an ability to form efficient multi-resolution representations (Daubechies, 1990). These properties have led to the use of wavelets in deep learning in two general ways. In one, wavelets are used as a pre-processing step to form noise-robust representations of time series, while in the second wavelets are employed to replace neurons to form wavelet neural networks. An example application of the former used Haar wavelets for stock price time-series forecasting with recurrent neural networks (Hsieh et al., 2011). In the latter scenario, wavelet neural networks have seen some success in time-series prediction (Chen et al., 2006), signal classification and compression (Kadambe and Srinivasan, 2006), but a lack of standard representations and general frameworks has prevented wider adoption (Alexandridis and Zapranis, 2013).

As a result, to the best of our knowledge, the wavelet transform has rarely been used as the representation domain for a *convolutional* neural network. In the following section we present our method, which leverages the benefits of the wavelet transform demonstrated in the signal processing literature, as well as the ability to form hierarchical feature representations for deep learning.

4.4 Method

We present a novel wavelet transform-based convolutional neural network architecture for the detection of events in noisy audio recordings. As our results of Section 4.6 indicate superior performance when training on wavelet representations of the data, we describe in depth the wavelet transform to provide insight into its benefits over the conventional STFT. We explain the wavelet transform in the context of the algorithm, thereafter describing the neural network configurations and a range

of traditional classifiers against which we assess performance. The key steps of the feature extraction and classification pipeline are given in Algorithm 1.

Algorithm 1 Detection Pipeline

1: Load N labelled microphone recordings $x_1(t), x_2(t), \ldots, x_N(t)$.
2: Take transform with h_1 features such that we form a feature tensor $\mathbf{X}_{\text{train}}$ and corresponding label vector $\mathbf{y}_{\text{train}}$:

$$\mathbf{X}_{\text{train}} \in \mathbb{R}^{N_S \times h_1 \times w_1}, \mathbf{y}_{\text{train}} \in \mathbb{R}^{N_S \times n},$$

where N_s is the number of training samples formed by splitting the transformed recordings into 2D 'images' with dimensions $h_1 \times w_1$, and n is the number of classes.
3: Train classifier on $\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}$.
4: For test data, $\mathbf{X}_{\text{test}}$, neural network outputs a prediction $\hat{y}_i$ for each class C_i:

$$0 \leq \hat{y}_i(\mathbf{x}) \leq 1, \quad \text{such that} \quad \sum_{i=1}^{n} \hat{y}_i(\mathbf{x}) = 1.$$

4.4.1 The wavelet transform

We begin by discussing the details of the transform used in Step 2 of Algorithm 1 as a base to further extract features. A well-established approach in signal processing is the Fourier transform, which can be used to express any signal with an infinite series of sinusoids and cosines. Its main disadvantage is the provision only of frequency resolution, meaning one can identify all the frequencies present in a signal, but not their occurrence in time. To overcome this, common approaches include cutting the signal into sections of time and treating each segment separately. This action however smears out frequencies, especially in the case of short windows. A wide window is able to provide better frequency resolution, however at the sacrifice of time resolution. Choosing a window function therefore limits one to a fixed time-frequency resolution. The uncertainty in time-frequency is referred to as the Heisenberg-Gabor limit (Bansal and Kumar, 2015) which is derived from the notion that the product of the precision in time and frequency is limited.

The wavelet transform employs a fully scalable modulated window which provides a

principled solution to the windowing function selection problem (Valens, 1999). The window is slid across the signal, and for every position a spectrum is calculated. The procedure is then repeated at a multitude of scales, providing a signal representation with multiple time-frequency resolutions. This allows the provision of good time resolution for high-frequency events, as well as good frequency resolution for low-frequency events, which in practice is a combination best suited to real signals.

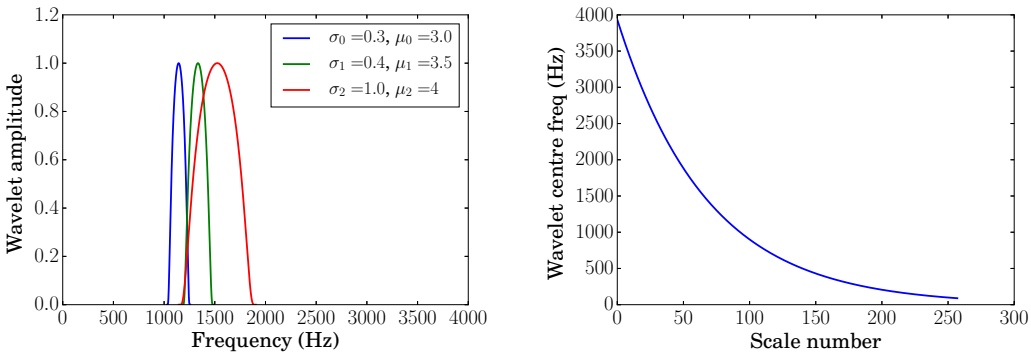
We choose to use the Continuous Wavelet Transform (CWT) due to its successful application in time-frequency analysis (Daubechies et al., 2011). The CWT is particularly well-suited over the discrete wavelet transform to time-frequency analysis as redundancy makes information available in peak shape and peak composition more visible and easier to interpret (Du et al., 2006). The CWT can be written in the time domain as:

$$a(s,\tau) = |s|^{-1/2} \int_{-\infty}^{\infty} f(t)\psi^* \left(\frac{t-\tau}{s}\right) \mathrm{d}t, \tag{4.1}$$

or equivalently in the frequency domain as:

$$a(s,\tau) = |s|^{1/2} \int_{-\infty}^{\infty} F(\omega)\Psi^*(s\omega)e^{i\omega\tau} \,\mathrm{d}\omega, \tag{4.2}$$

where s is the scale factor, τ is the translation factor, $|s|^{-1/2}$ is the energy normalisation factor, and * denotes complex conjugation. The wavelets are generated by scaling and translating a single mother wavelet $\psi(t)$. Through continuous dilation in τ, the resulting CWT coefficients $a(s,\tau)$ can be assembled for a multitude of scales to either reconstruct the signal with an inverse transform, or to create a spatial representation, called the scaleogram. An equivalent representation in Fourier space requires the continuous application of 1D Fourier transforms with windows that are translated in time. We can illustrate this by substitution of $\psi^*_{s,\tau}(\frac{t-\tau}{s}) = e^{-i\omega t}$ in Equation (4.1). Essentially, this is equivalent to using a fixed basis with $s = 1$ and ignoring the dilation in τ. We thereby emphasise the more principled solution employed with the CWT, eliminating the need to choose and parameterise the window function necessary for STFT representations. Furthermore, working with the CWT one is free to choose a wavelet function with properties and characteristics



(a) Bump mother wavelets of fixed scale, s_{10}, with varying values of μ, σ, constructed from Equation (4.3).

(b) By converting wavelet scale to frequency, $f = (\frac{1}{s}\frac{\mu}{2\pi})$, we can illustrate the tiling of the frequency plane with the bump wavelet

Figure 4.1: Illustration of key properties of the bump wavelet, constructed from Equations (4.2) and (4.3).

that best suit the data, given knowledge of the signal being analysed. A popular choice of wavelet function for time-frequency analysis is given by the bump wavelet (Vialatte et al., 2009), expressed in the Fourier domain as:

$$\Psi(s\omega) = \exp\left(1 - \frac{1}{1-(s\omega-\mu)^2/\sigma^2}\right)\mathbb{I}[(\mu-\sigma)/s, (\mu+\sigma)/s], \tag{4.3}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Valid values for μ, σ are $[3, 6], [0.1, 1.2]$ respectively. Smaller values of σ result in a wavelet function spanning a narrower frequency bandwidth (Figure 4.1a), which results in superior frequency localisation but poorer time localisation. The bump wavelet is symmetric in frequency and has a direct relationship between wavelet scale and centre frequency, which we illustrate in Figure 4.1b. As a result we can create spectrograms in frequency which retain clear interpretability (which becomes important for Sections 4.5 and 4.7).

The spatial features thus created are then passed to the classifiers in the next step of the algorithm. We discuss neural network and more traditional implementations separately in the upcoming sections.

4.4.2 Neural network configurations

We show our CNN (See Section 2.4.4.1) architecture in Figure 4.2. The data size constraint results in an architecture choice of few layers and free parameters. We partition a spectrogram (either wavelet or STFT, as discussed in Section 4.5) of the mosquito recording into images of dimensions $h_1 \times w_1$. This serves as input to a single convolutional layer with N_k filters with filter weights $\mathbf{W} \in \mathbb{R}^{k \times k}$. Following convolution, feature maps are formed with dimensions reduced to $h_2 \times w_2$. These maps are fully connected to N_d units in the dense layer, which is connected to the two output classes with dropout (Srivastava et al., 2014) with probability p. ReLU activations are employed based on their desirable training convergence properties (Krizhevsky et al., 2012).

Finally, we perform grid search over potential candidate hyperparameters using 10-fold cross-validation on a subset of the mosquito training data. We detail the hyperparameter choice and show the results in Section 4.5.2. The combination of cross-validation and dropout helps avoid overfitting to our scarce data environment. This is shown by the excellent performance transfer with no hyperparameter retuning in Section 4.6.

4.4.3 Traditional classifier baseline

As a baseline, we compare the neural network models with more traditional classifiers that typically require explicit feature design. We choose three candidate classifiers widely used in machine learning with audio: RFs, NBs, and SVMs using a Radial Basis Function (RBF) kernel (Radial Basis Function-SVMs). Their popularity stems from ease of implementation, reasonably quick training, and competitive performance (Silva et al., 2013), especially in data-scarce problems. For brevity we present results with the best-performing of these only, namely the SVM.

We selected ten features to encode the observed raw data: STFT spectrogram slices with 256 coefficients (created with a Hanning window and 256 samples of overlap),

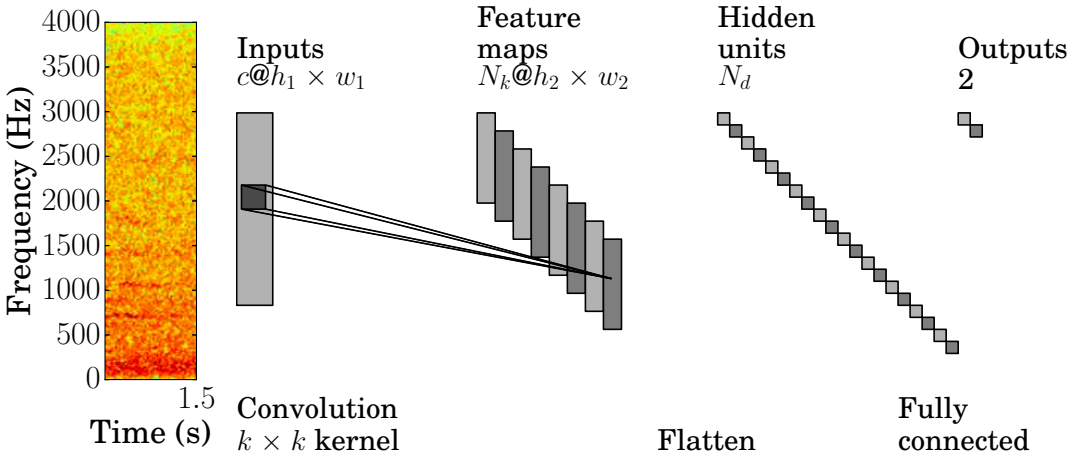


Figure 4.2: The CNN pipeline. 1.5 s spectrogram of mosquito recording is partitioned into single-channel images ($c = 1$) of dimensions $h_1 \times w_1$. The images serve as input to a convolutional layer with N_k filters with kernel size $k \times k$. Feature maps are formed with dimensions reduced to $h_2 \times w_2$ following convolution. These maps are fully connected to N_d units in the dense layer, fully connected to 2 units in the output layer.

13 MFCCs, entropy, energy entropy, spectral entropy, flux, roll-off, spread, centroid, and the zero crossing rate (for a detailed explanation of these features, see for example the open-source audio signal analysis toolkit by Giannakopoulos (2015)). We note that our choice of feature parameters is based on past literature (Dieleman and Schrauwen, 2014, STFT), (O'Shaughnessy, 2008, MFCCs), as well as empirical evidence. Prior parameterisation of the feature space is necessary to some extent, as the number of feature and classifier parameters grows combinatorially to the point where joint optimisation of all possible variables is infeasible. We select certain aspects of classifier-feature pipelines by cross-validation as detailed in Section 4.5.2.

4.5 Experimental details

4.5.1 Datasets

The mosquito data used here were recorded in January 2016 within culture cages containing both male and female *Culex quinquefasciatus*. Mosquito wingbeat sounds

commonly have a fundamental frequency in the range of 150 to 750 Hz (Clements, 1999). In noisy recording conditions, higher harmonics are less audible due to the faster decay of shorter wavelength waves. Furthermore, the signals are sampled with inexpensive smartphone microphones to allow widespread deployment at low cost. Given the quality of these microphones, we observe empirically that sound emitted by mosquitoes mostly disappears in noise for frequencies higher than the third harmonic. We therefore choose to sample at $F_s = 8\,\text{kHz}$. Figure 4.3 shows a frequency domain excerpt of a particularly faint recording in the windowed frequency domains. For comparison we also illustrate the wavelet scalogram taken with the same number of scales as frequency bins, h_1, in the STFT. We plot the logarithm of the absolute value of the derived coefficients against the spectral frequency of each feature representation. Figure 4.3 (lower) shows the classifications within $y_i = \{0, 1\}$: absence, presence of mosquito, as labelled by four individual human researchers. These labels are created in version 2.2.2 of Audacity (2018) with access to the recording audio and a matching spectrogram visualisation. Of these, one particularly accurate label set created with great care under ideal conditions is taken as a gold-standard reference to both train the algorithms and benchmark with the remaining experts. The classifications are restricted to only 2 classes due to the absence of labelled data in a multi-species scenario.

However, to stress the transferability of the algorithm, but maintain focus on data-restricted audio scenarios, we choose to use a 10 class subset from the BirdCLEF dataset (Goëau et al., 2015), with a range of 2 to 4 recordings per class of maximum length of 2 minutes per class. The mixed sample rate recordings are re-sampled to 44.1 kHz and assigned either a bird class label or noise label at an interval of 0.1 seconds per recording. The recording-specific noise class labels are all grouped into a single noise class. We note that beyond the simple re-scaling of features to the appropriate sample rate, we choose not to hand-tailor any features or classifiers to demonstrate the relative effectiveness of the application of the surveyed methods to data that is novel to the classification pipeline.

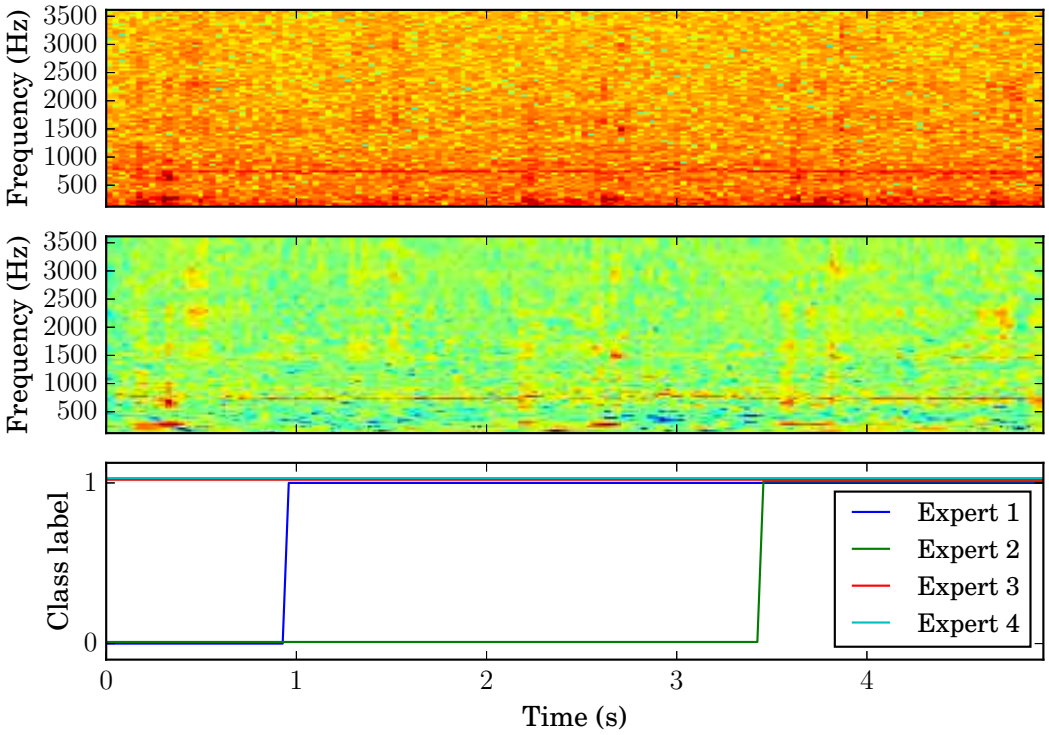


Figure 4.3: STFT (top) and wavelet (middle) representations of signal with $h_1 = 256$ frequency bins and wavelet scales respectively. Corresponding varying class labels (bottom) as supplied by human labellers. The wavelet representation shows greater contrast in horizontal constant frequency bands that correspond to the mosquito tone.

4.5.2 Cross-validated parameter search

In this section we describe the experiment design and choice of hyperparameters, optimised to maximise F_1 score over a cross-validation sample. The available 57 mosquito recordings were split into 50 % training and 50 % held out data. The training data was then further split tenfold to perform cross-validation, creating approximately 3,000 to 30,000 training samples, for window widths $w_1 = 10$ and $w_1 = 1$ samples, respectively. The neural networks were trained with a batch size of 256 for 20 epochs, according to validation accuracy in conjunction with early stopping criteria.

For fair comparison we partition the data (choose window length w) to the strengths of each individual classifier. When evaluating cross-validation performance over the

label interval, our hyperparameter optima ($w_1 = 10$ for the CNN, and $w_1 = 1$ for the SVM) given in bold in Table 4.1 suggest that stacking the windows together creates feature vectors that lead to performance degradation for the SVM. Therefore, for each CNN input image with height $h_1 = 256$ and width $w_1 = 10$, our SVM will use 10 training samples with $w = 1, h = 256$ instead. Optimum window widths may vary with the dynamics of the signal, so it is important to consider this parameter systematically. In particular, should significantly more data be available, the drawbacks due to the use of larger windows (decrease in training samples the classifier sees) would be mitigated by the higher number of training samples at disposal. Conversely, the advantage of longer windows lies with supplying a longer temporal context.

The traditional classifiers are cross-validated with Principal Component Analysis (PCA), and Recursive Feature Elimination (RFE) (Guyon et al., 2002), with the number of components controlled by n and m respectively. The best performing feature set for all traditional classifiers is the set extracted by cross-validated RFE, outperforming all PCA reductions for every classifier-feature pair. The highest scoring hyperparameter, $m = 27$, defines a feature set, that we denote as RFE_{88}, which retains 88 dimensions from the ten original features spanning 304 dimensions ($F_{10} \in \mathbb{R}^{304}$).

4.5.3 Computational details

We consider computational complexity by splitting the pipelines into three processing stages: feature transformation, classifier training, and classifier prediction. The overall compute time is thus the sum of all three. We further break this down for individual pipelines, noting that various software libraries can differ significantly in processing time for the same transformation. We offer some insights from the figures given in Table 4.2 in this subsection.

The native training complexity of the SVM is stated as $\mathcal{O}(n_{\text{SV}}d)$, where d is the input

Table 4.1: Results for grid search over hyperparameter values. The cross-validated F_1 score is reported for optimal hyperparameters found (in bold).

Classifier	Features	Parameter grid	F_1 score
CNN	STFT	$w_1 \in \{1, \mathbf{10}, 100\}$, $k \in \{2, \mathbf{3}, 4, 5\}$, $N_k \in \{8, 16, 32, \mathbf{64}, 128, 256, 1028, 2056\}$, $N_d \in \{64, \mathbf{128}, 256, 512, 1024\}$.	0.861 ± 0.066
CNN	Wavelet	$w_1 \in \{1, \mathbf{10}, 100\}$, $k \in \{2, 3, 4, \mathbf{5}\}$, $N_k \in \{8, 16, 32, \mathbf{64}, 128, 256, 1028, 2056\}$, $N_d \in \{64, \mathbf{128}, 256, 512, 1024\}$.	0.915 ± 0.023
NB, RF, SVM	$F_{10} \in \mathbb{R}^{304}$	$w_1 \in \{\mathbf{1}, 10, 100\}$, PCA $\in \mathbb{R}^N$, $N \in 0.8^n \times 304$, $n \in \{0, 1, \ldots, 12\}$, RFE $\in \mathbb{R}^M$, $M \in 304 - 8m$, $m \in \{0, 1, \ldots, \mathbf{27}, \ldots, 35\}$.	0.880 ± 0.055

dimension, and n_{SV} is the number of support vectors (Claesen et al., 2014). Table 4.2 shows that an increase in d coupled with the already large number of training samples (leading to a large n_{SV}) causes a significant slowdown in both training and prediction with the SVM. A feature dimension reduction, as encountered with the MFCC or RFE approaches, while slightly more costly as a pre-processing step, speeds up the training and prediction significantly.

The CNN was trained in Keras (Chollet et al., 2015), with an NVIDIA 970 GTX GPU. This allows quick training and prediction, resulting in much shorter computation times than those of the `scikit-learn` SVM (running on a CPU) when working with a large feature space.

Furthermore, the CWT is highly redundant and so incurs a greater computational cost. Its computational complexity increases linearly with the number of wavelet scales (provided sufficient RAM). Despite this, the sum of feature transformation and training time is well under the length of the audio recordings, suggesting real-time detection to be perfectly feasible given appropriate hardware. A significant reduction in the CWT processing time can be achieved by calculating each wavelet scale in parallel, due to the independence of each computation per scale. Further considerable speed-up can be achieved by utilising a discrete wavelet transform or a

Table 4.2: Execution time given in seconds for feature-classifier pipelines trained on 15 minutes (900 s) and evaluated over 15 minutes of audio data sampled at 8000 Hz. Run times are an average of three passes on a mid-range desktop with an Intel i7-4790k CPU with 16 GB DDR4 RAM and an NVIDIA GTX 970 GPU.

Pipeline	Dimensions	Feature transform	Training	Prediction
SVM MFCC	13	1.5 s	17.0 s	3.6 s
SVM RFE_{88}	88	5.2 s	42.9 s	8.4 s
SVM STFT	256	1.4 s	121.1 s	32.7 s
SVM Wavelet	256	170.6 s	131.4 s	28.9 s
CNN MFCC	13	1.5 s	16.0 s	1.0 s
CNN RFE_{88}	88	5.2 s	24.0	1.0 s
CNN STFT	256	1.4 s	37.5 s	4.0 s
CNN Wavelet	256	170.6 s	44.8 s	7.5 s

fast wavelet transform (Cody, 1992). While not the focus of this specific chapter, this may be worth considering when transferring algorithms to embedded devices and specialised hardware in future work.

4.6 Classification performance

The performance metrics are defined at the resolution of the supplied label interval (0.1 second granularity) and presented in Table 4.3 for the mosquito dataset, and in Table 4.4 and Figure 4.4 for the BirdCLEF subset. We highlight three key results in both applications.

4.6.1 Mosquito detection

Firstly, both the traditional and deep learning algorithms accurately and reliably detect mosquitoes, far surpassing human labellers in both F_1 score and precision–recall area. Since human labels were supplied as absolute (either $\hat{y}_i = 1, \hat{y}_i = 0$), an incorrect label incurs a large penalty on precision–recall curve areas, explaining the large precision–recall area deficit attributed to human labelling.

Secondly, the CNN provides a consistent performance boost with every feature combination, even for the features hand-crafted for use with SVMs (RFE_{88}).

Table 4.3: **Mosquito detection**: summary classification metrics reported as means ± the standard deviation from $n = 30$ random hold out dataset splits with 50 % training data, and 50 % test data.

Classifier	Features	F_1 score	TPR	TNR	PR area
CNN	MFCC	0.895 ± 0.022	0.89 ± 0.04	0.90 ± 0.03	0.963 ± 0.012
SVM	MFCC	0.880 ± 0.020	0.88 ± 0.02	0.88 ± 0.02	0.951 ± 0.013
CNN	**$\mathbf{RFE_{88}}$**	$\mathbf{0.922 \pm 0.019}$	$\mathbf{0.93 \pm 0.02}$	$\mathbf{0.91 \pm 0.04}$	$\mathbf{0.980 \pm 0.007}$
SVM	RFE_{88}	0.904 ± 0.020	0.91 ± 0.02	0.90 ± 0.02	0.963 ± 0.013
CNN	STFT	0.883 ± 0.031	0.86 ± 0.05	0.91 ± 0.02	0.939 ± 0.017
SVM	STFT	0.858 ± 0.031	0.80 ± 0.05	0.91 ± 0.02	0.889 ± 0.036
CNN	Wavelet	0.913 ± 0.020	0.92 ± 0.02	0.91 ± 0.02	0.962 ± 0.012
SVM	Wavelet	0.897 ± 0.020	0.90 ± 0.02	0.90 ± 0.02	0.944 ± 0.012
Labeller 1	Audacity	0.819 ± 0.018	0.89 ± 0.02	0.85 ± 0.02	0.843 ± 0.006
Labeller 2	Audacity	0.856 ± 0.019	0.92 ± 0.03	0.88 ± 0.02	0.873 ± 0.008
Labeller 3	Audacity	0.852 ± 0.018	0.77 ± 0.02	0.98 ± 0.02	0.901 ± 0.007

Table 4.4: **BirdCLEF subset**: summary classification metrics reported as means ± the standard deviation from $n = 30$ random dataset splits with 50 % training data, and 50 % test data.

Classifier	Features	F_1 score	PR area
CNN	MFCC	0.860 ± 0.028	0.915 ± 0.031
SVM	MFCC	0.891 ± 0.029	0.931 ± 0.029
CNN	RFE_{88}	0.853 ± 0.036	0.853 ± 0.036
SVM	RFE_{88}	0.857 ± 0.024	0.905 ± 0.030
CNN	STFT	0.909 ± 0.031	0.927 ± 0.031
SVM	STFT	0.757 ± 0.021	0.821 ± 0.027
CNN	**Wavelet**	$\mathbf{0.925 \pm 0.021}$	$\mathbf{0.947 \pm 0.023}$
SVM	Wavelet	0.896 ± 0.023	0.939 ± 0.022

Finally, we note that the wavelet pipeline strongly outperforms the STFT, with both the CNN and SVM.

4.6.2 Bird classification

We now make three observations from Figure 4.4 and Table 4.4, representing a scenario that is novel to the classifier pipelines.

Firstly, the wavelet features provide the best performance with both the CNN and the SVM, with the top result achieved by the CNN-wavelet pipeline. As with the prior application, the wavelet significantly outperforms the STFT with all classifiers, with the difference magnified in this application.

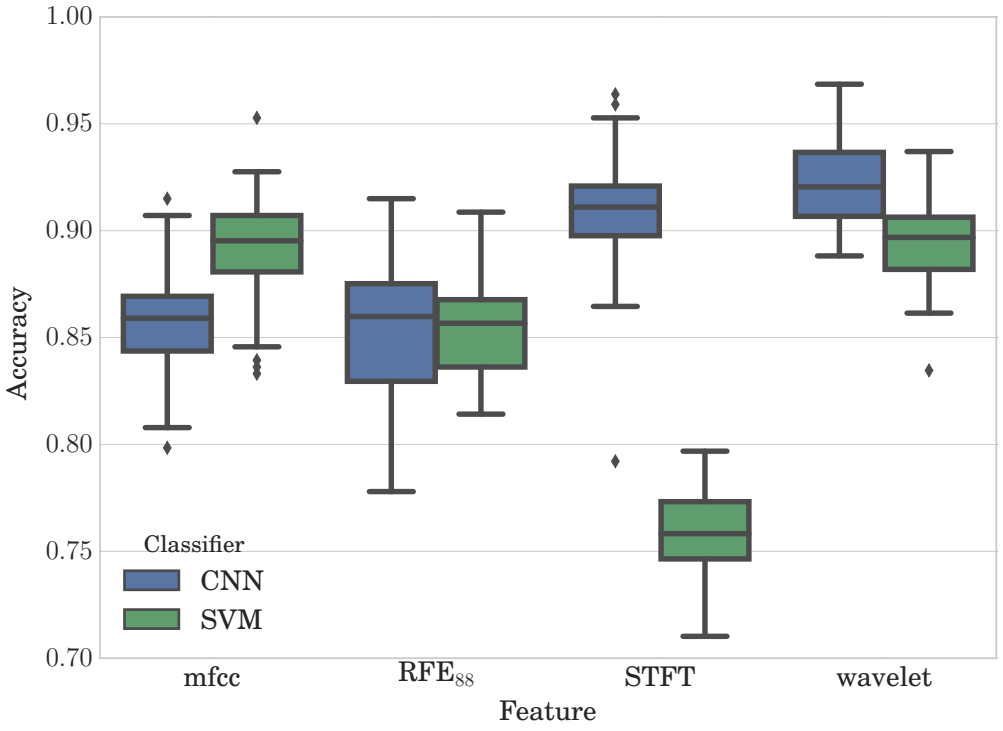


Figure 4.4: **BirdCLEF** subset: boxplots of mean accuracy per class (F_1 score) for $n = 30$ trials of the CNN and SVM methods, grouped by feature combination.

Secondly, the downside to the elaborate hand-tuned feature selection scheme (RFE_{88}) quickly becomes evident when comparing performance conditioned on these features (with F_1 scores of approximately 0.85) to the results of either general deep learning configuration (with F_1 scores of 0.91 and 0.93 for the STFT and wavelet, respectively). We find our results consistent with claims made about the unsustainable nature of hand-crafted feature and classifier design (Humphrey et al., 2013).

Finally, the CNN performs significantly better with high-dimensional, generalisable, features (STFT and wavelet) in this more difficult problem.

4.7 Visualising discriminative power

In the absence of data labels, visualisations can be key to understanding how neural networks obtain their discriminative power. To ensure that the characteristics of

Algorithm 2 Spectra calculation

for feature in {STFT, wavelet} **do**
  for class in C_i **do**
    Collect highest N predictions, $\hat{y}_i$
    Collect corresponding inputs, $\mathbf{X}_{i,\text{test}} \in \mathbb{R}^{N \times h_1 \times w_1}$,
      ▷ forming a concatenation of 2D images with dimensions $h_1 \times w_1$.
    Collect N_s training samples to form $\mathbf{X}_{i,\text{train}}$
    Take ensemble average across patches and individual columns:

$$\mathbf{x}_{i,\text{test}}(f) = \frac{1}{w_1}\frac{1}{N}\sum_{j=1}^{w_1}\sum_{k=1}^{N} X_{ijk,\text{test}}, \text{ where } X_{ijk} \in \mathbb{R}^{h_1}, \tag{4.4}$$

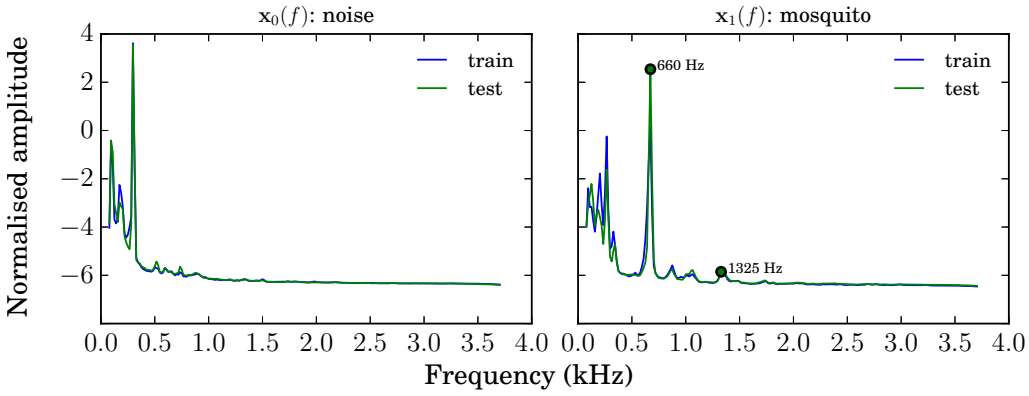
$$\mathbf{x}_{i,\text{train}}(f) = \frac{1}{w_1}\frac{1}{N_s}\sum_{j=1}^{w_1}\sum_{k=1}^{N_s} X_{ijk,\text{train}}. \tag{4.5}$$

    Normalise by mean and standard deviation
  end for
end for

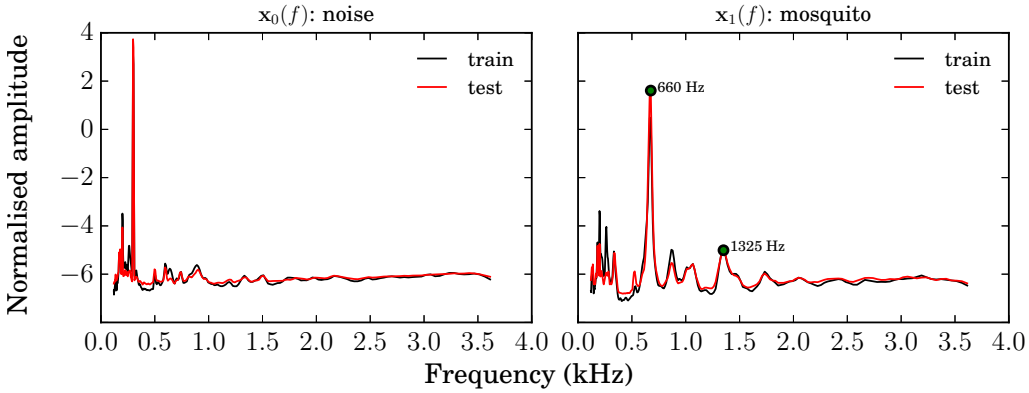
the signal have been learnt successfully, we compute the frequency spectra $\mathbf{x}_{i,\text{test}}(f)$ of samples that maximally activate the network's units. We compare this to the training spectra $\mathbf{x}_{i,\text{train}}(f)$ using Algorithm 2.

Figures 4.5 and 4.6 show that the test samples closely resembling the training set cause the highest activations – a property we expect from our algorithms to verify they have successfully been trained. Furthermore, Figure 4.5 shows that our prior expectation for the mosquito class matches the spectral content that triggers the most confident predictions. This is in the form of a distinct frequency peak around 660 Hz and its harmonic at 1325 Hz, which differs significantly from the noise class. Similarly, Figure 4.6 shows unique spectral regions dedicated to each species, also with significant deviation from the noise class.

As we chose a wavelet basis with a scale directly proportional to a centre frequency, we can directly compare spectral representations with the STFT. The wavelet representation results in the more easily distinguishable peaks in the mosquito class (Figure 4.5), and overall smoother spectral representations of the bird calls (Figure 4.6). We note that a mismatch between high-scoring test and labelled spectra (or matches in non-information-bearing regions of the spectrum) suggest that the



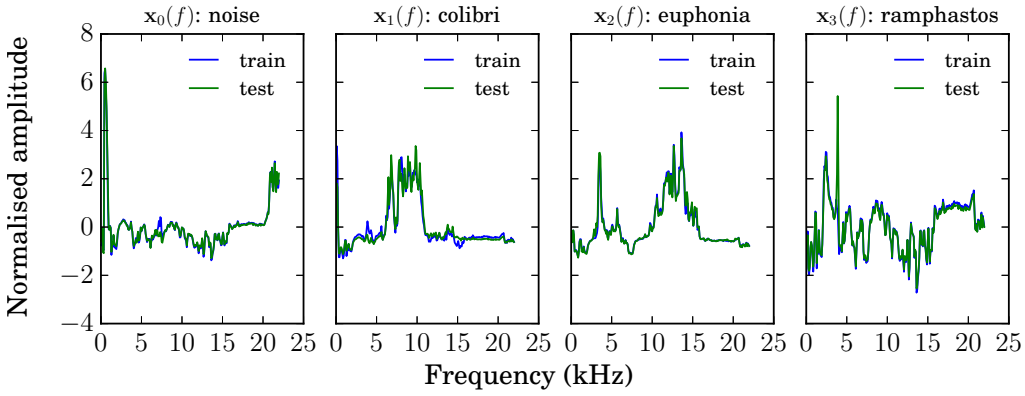
(a) STFT



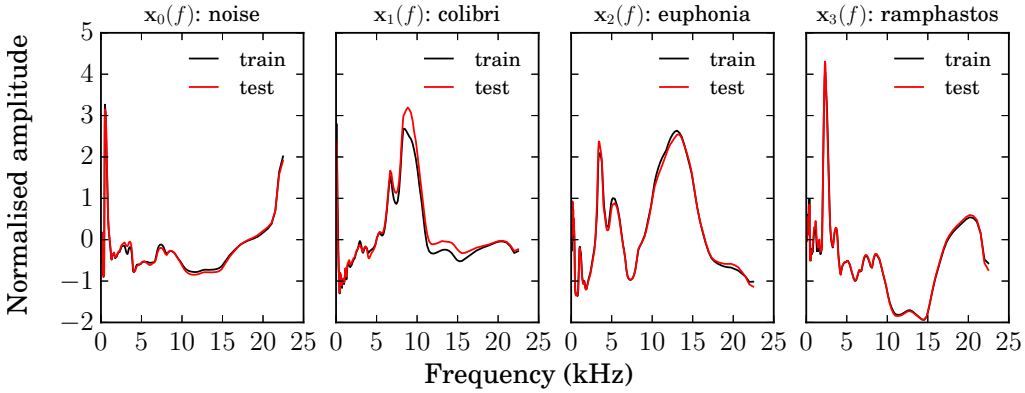
(b) Wavelet

Figure 4.5: **Culex mosquito** dataset: plot of normalised feature coefficient against STFT frequency bin, (a), and wavelet centre frequency, (b), for the 10 % most confident predicted outputs over a test dataset. The learned spectra $\mathbf{x}_{i,\text{test}}(f)$ for the highest N scores closely match the labelled class spectra $\mathbf{x}_{i,\text{train}}(f)$.

network could be learning to detect the noise profile of the microphones used for data collection rather than the sound emitted by the object of interest.



(a) STFT



(b) Wavelet

Figure 4.6: **BirdCLEF** subset: plot of normalised feature coefficient against STFT frequency bin, (a), and wavelet centre frequency, (b), for the 10 % most confident predicted outputs over a test dataset. The learned spectra $\mathbf{x}_{i,\text{test}}(f)$ closely match the labelled class spectra $\mathbf{x}_{i,\text{train}}(f)$.

4.8 Chapter summary

This chapter has presented a novel approach for acoustic classification in a real-world, data-scarce scenario. We have been able to more accurately and reliably differentiate between the presence and absence of a mosquito than human labellers. Furthermore, we have shown that a CNN outperforms generic classifiers such as support vector machines commonly used in the field.

Moreover, we have highlighted the importance of the generality of deep learning approaches by evaluating classification performance over a 10 class subset of bird species recordings, where the wavelet-trained CNN outperforms traditional clas-

sification algorithms with no hyperparameter re-tuning of either approach. The consistent improvement observed with wavelet features over the STFT warrants further research on whether the STFT is the correct choice of a base transform, as has overwhelmingly been used in the literature.

Finally, our generic feature transform has allowed us to visualise the learned class representation by back-propagating predictions made by the network. We thus verify that the network correctly infers the frequency characteristics of the signal, rather than a peculiarity of the recording such as the microphone noise profile. In the following chapter we deploy a computationally efficient algorithm in a physical device for the purpose of further data collection. Based on the balance of good overall F_1 score and quick prediction times, we have chosen the SVM MFCC pipeline (Section 4.5). In turn, the pipeline has been used to collect more data in two field studies, as well as fuel further research. We discuss the applications that were taken forward, and their overarching contributions, in the introduction of the upcoming chapter.

Chapter 5

HumBug: data acquisition and smartphone applications

5.1 Overview

This chapter continues the timeline given in the Gantt chart of Figure 3.2 (Chapter 3), at the stage of completion of expertly-labelled data experiments. In this section we give the context of the contributions of three publications, which all revolve around the data collection process, fieldwork, and considerations for the real-time smartphone app. In each section we elaborate on these contributions in greater detail.

Following research output presented in Chapter 4, which strongly suggests that there is value in algorithmic detection of mosquito sounds when compared to human interpretation, we seek to advance this project through further data collection. To aid in data collection, we utilise an updated version of a smartphone app which gives the ability to record on detection of a mosquito. We discuss this in Section 5.2, which builds upon work published and presented at the *NeurIPS 2017 Workshop on Machine Learning for the Developing World* (Li et al., 2017b).

Further fieldwork conducted with this algorithm is then discussed in Section 5.3 which is based on work accepted at the *DCASE 2018 Workshop* proceedings (Li

et al., 2018).

Following the initial investigations of both offline recordings and field trials, we show how a detector can be built which is both computationally efficient and takes into account a method to penalise different types of error in Section 5.4. The work on which this section was based on was presented at the *NeurIPS 2017 Workshop on Machine Learning for Audio Signal Processing*.

Following the acquisition of further data, Section 5.5 then details the process of collecting labels from the Zooniverse crowdsourcing platform. The increase in the size of the dataset allows us to scale to larger networks with a greater number of layers. We publicly release our model and labelled data to encourage further research. Our contributions are published at the *Machine Learning for the Developing World* workshop at *NeurIPS 2019*, and the *International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2020*.

5.2 Mosquito detection with low-cost smartphones: data acquisition for malaria research

In this section, we present a prototype mobile sensing system that acts as both a portable early warning device and an automatic acoustic data acquisition pipeline to help fuel scientific inquiry and policy. The machine learning algorithm that powers the mobile system achieves excellent off-line multi-species detection performance while remaining computationally efficient. Furthermore, we have conducted preliminary live mosquito detection tests using low-cost mobile phones and achieved promising results. Following an initial proof-of-concept phase, the system underwent field tests in July 2018 to improve and evaluate the detection model with more field data. We show the results of the field test in Section 5.3. As of May 2020, further deployment of our detection ecosystem is planned in the near future in Africa.

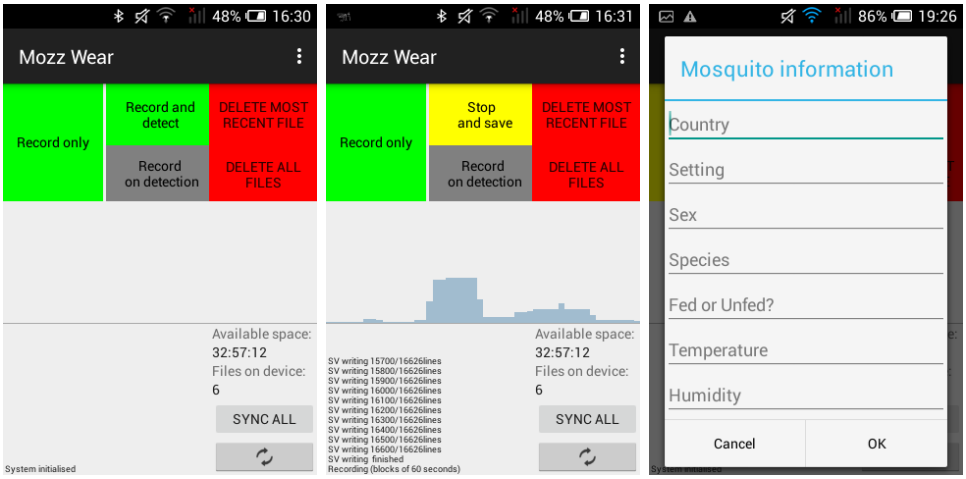
5.2.1 Mobile devices

As sensor-rich embedded devices, smartphones provide a perfect platform for environmental sensing (Lane et al., 2010). They are programmable and equipped with cheap yet powerful sensors, such as microphones, GPS, digital compasses and cameras. The touch screen is ideal for displaying real-time feedback to the user and inputting peripheral information in data acquisition. The built-in WiFi and cellular access make smartphones extremely useful for data streaming or synchronisation. These desirable properties have enabled a wide range of applications of smartphones in environmental monitoring (Yang et al., 2014; Lane et al., 2015; Guo et al., 2015).

The identification of disease-carrying mosquitoes by their flight tones has been researched for more than half a century (Jr. and Kahn, 1949; Raman et al., 2007). However, to the best of our knowledge, there is no acoustic mosquito sensing pipeline that is cost-effective and deployable in large field studies. The HumBug project aims to accomplish real-time acoustic mosquito detection using low-cost smartphones to guide control programmes, through relating such detections to further information such as geographic location, vegetation type, and climate.

5.2.2 Data acquisition

To enable acoustic detection of mosquitoes, an easy-to-use data acquisition system needs to be set up for retrieving and transmitting data for training of detection models and live prediction of mosquito presence. The *MozzWear* Android app we developed provides a simple graphical user interface for sound recording and data synchronisation (Figure 5.1a). At this stage of the project, the app supported the "Record only", and the "Record and detect" functions. Future work aims to support the "Record on detection" function. The "Record and detect" and the "Record only" both record sound. In addition, the "Record and detect" function launches the detection module and displays real-time predictions of the mosquito presence (Figure 5.1b). Users of the app may have certain prior information to enter at the



(a) MozzWear Interface (b) MozzWear interface ("Record and detection"). (c) Peripheral Information Input.

Figure 5.1: The MozzWear android app. The latest version is available on https://github.com/HumBug-Mosquito

data collection stage, e.g. peripheral information of the environment, or species categories for cage mosquitoes. The app provides a pull-down menu (Figure 5.1c) for users to enter such information (if available). All recordings and corresponding peripheral information are synchronised to the online HumBug project database through WiFi or cellular connections.

5.2.3 Classification pipeline

The key objective of the classification algorithms for our specific task is high detection accuracy with low computational load and memory usage. We describe our adopted feature extraction methods, two-stage multi-species classification algorithm, as well as the model training strategy.

Feature extraction: As introduced in Chapter 2, where we surveyed a variety of audio representations, MFCCs are one of the most widely used audio features in speech recognition and acoustic scene classification (Barchiesi et al., 2015). MFCCs provide compact representations of spectral envelopes, by performing the DCT on the log power spectrum grouped according to the mel scale of frequency bands. It

usually compresses the high-dimensional spectrum into a much lower space, e.g. 13-dimensional coefficients. Thus, it is well suited for usage in low-power devices.

Two-stage multi-species classification: A two-stage classification paradigm is employed for detection. In the first stage, a binary class SVM is used to detect the presence of mosquitoes. Once a mosquito is identified, a multi-species classification using the one-versus-one multi-class strategy with the SVM is used in the second stage to identify the exact mosquito species. In future work, Section 5.4 furthermore introduces the cost-sensitive SVMs in the first stage to provide a more direct control on the false positive rate.

Training strategy: At this stage of the project, only small amounts of training data were available, as labelling recordings is a manual and time-consuming process. We divided recordings into audio clips with a duration of 0.1 seconds (which we call samples), and assigned labels to each clip according to data tagging results from humans. A balanced training dataset was first created so that each class had the same number of training samples. The random sampling strategy, which randomly samples short audio clips without replacement in the balanced training set, was found to produce the best detection performance.

5.2.4 Experiments and results

We report offline detection results produced with a dataset containing audio recordings collected by the CDC and USAMRU-K. The CDC dataset contains sound recordings from 20 mosquito species. The number of recordings for a majority of the species is small – for only 6 species we have no fewer than 62 0.1 second samples. Therefore, we collate all acoustic data from these 6 species into the examined dataset. The USAMRU-K dataset contains sound recordings of the *An. Gambiae* species. We create a balanced dataset by downsampling data from each of these 7 mosquito species and background recordings, so that each class contains 62 samples. The

Table 5.1: Statistics of detection accuracies with 7 mosquito species. Results are generated from 100 random splits of balanced data.

Species	Mean ± std.	Min	Max
No mosquito	0.82 ± 0.11	0.50	1.00
Aedes aegypti	0.82 ± 0.07	0.58	0.97
Anopheles quadrimaculatus	0.72 ± 0.09	0.41	0.94
Culex tarsalis	0.92 ± 0.06	0.74	1.00
Anopheles albimanus	0.76 ± 0.07	0.57	0.93
Culex quinquefasciatus	0.87 ± 0.06	0.69	1.00
Aedes albopictus	0.83 ± 0.08	0.60	1.00
Anopheles gambiae	0.68 ± 0.17	0.29	1.00

balanced dataset simplifies training and leads to easier interpretation of prediction results on the test set.* 100 trials were conducted by sampling 100 different copies of the balanced datasets through random initialisations. The "random sampling" training strategy described previously is adopted to train the detection algorithm with 50 % of samples in the balanced dataset. All the recordings used in this section were obtained with MozzWear installed on Alcatel One Touch 4009X mobile phones, available at a network operator's shop in the UK for £20. The app is therefore suitable for low-end devices, aiding the ease of global deployment.

We report statistics of detection performance on test data among 100 trials in Table 5.1. Average detection accuracies for different mosquito species vary from 0.68 to 0.92. The standard deviations calculated based on detection accuracies in 100 trials are low for most classes.

These results demonstrate that the current detection model, trained with a limited amount of data, has achieved effective multi-species detection with a dataset involving two experiments conducted in two different locations. Field trips are conducted and addressed in Section 5.3 to train more robust models and better assess detection performance using field data.

*We explore approaches to imbalanced classification, once a larger quantity of data has been accumulated, in Section 5.5.

5.3 Fast mosquito acoustic detection with field cup recordings: an initial investigation

Our previous assessment of mosquito acoustic detection techniques relied upon lab recordings of mosquitoes provided by the CDC and USAMRU-K. We illustrate in this section our proposed algorithm's performance over an extensive dataset, consisting of recordings made in plastic cups of more than 1000 mosquito individuals from 6 species captured in field studies in Thailand.

We propose an effective classification pipeline based on the mel-frequency spectrum allied with convolutional neural networks. This detection pipeline is computationally efficient in not only detecting mosquitoes, but also in classifying their species. Similarly to the wavelet, we use a transformation that reduces the amount of feature engineering, while still making use of a compressed feature space.

A total of 1256 individual mosquitoes of 9 different mosquito species were captured and individually recorded during a two-month mosquito survey in rural Thailand. We here present the development of a machine learning algorithm that is computationally efficient and report in this section on its performance over this field-recorded dataset.

5.3.1 Field experiments and data summary

A two-month comprehensive survey of mosquito fauna was conducted at Pu Teuy Village, Sai Yok District, Kanchanaburi Province, Thailand. The survey was conducted within the peak mosquito season (May to October), and ran from 12 June until the end of July in 2018. Three methods of capture were used: human-baited nets, cow-baited nets, and larval collections. The baited nets were run for 12 hours overnight with collections made each hour throughout. All adult mosquitoes were placed into sample cups large enough for them to fly freely and their flight was recorded the morning following capture.

Recording was conducted using two microphone set-ups: “budget” and “high spec”. The “budget” set-up used an Alcatel OneTouch Pixi smartphone with a TIE 19-90003 condensor microphone. The budget setup also used our MozzWear app on Alcatel smartphones to perform data capture and digitisation. The ‘high-spec’ set-up used a high specification field microphone (Telinga EM-23) plugged into a digital sound recorder (Olympus LS-14). Monophonic recordings were collected in both set-ups. Larval collections were made along a small river with known *anopheline* larval sites and the larvae/pupae were placed into rearing trays. The emerging adults were placed into individual sample cups and provided with 10 % w/v* sucrose solution and recorded as above. As of July 2018, a total of 1256 individual mosquitoes had been captured. The detailed number of individuals for each species is reported in Table 5.2. A total of 127 mosquito individuals died before recording, 92 were lost, 46 did not fly and 21 individuals are as yet unidentified.

After recording the mosquito flight tones of these captured mosquito individuals, data tagging was required to mark segments of recordings with mosquito flight tones. Our project research team labelled a subset of the recordings and obtained more than 1 hour of mosquito flight tones from the field-captured mosquitoes, in addition to background recordings. The number of mosquito individuals and durations of flight tones of each species are shown in Table 5.2.

5.3.2 Model

We have shown in Chapter 4 that the wavelet transformation provides informative time-frequency features that are well-suited to subsequent use, particularly when paired with convolutional neural networks. However, the continuous wavelet transformation used in this work is computationally demanding (as shown in Table 4.2) and unsuited for the task of real-time mosquito species classification on low-cost phones.

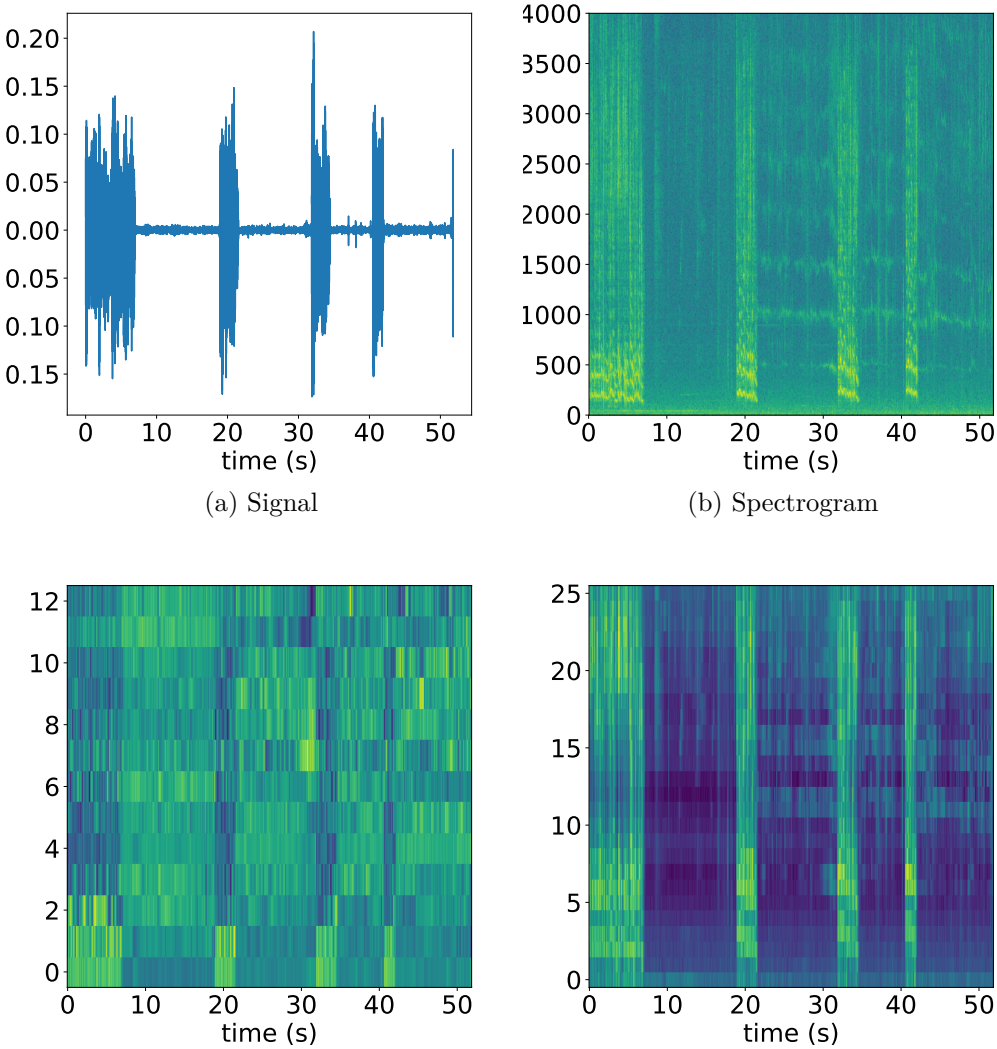
*weight by volume

Table 5.2: Number of captured mosquito individuals and durations of recorded mosquito flight tones for different species.

Mosquito species	# individuals	Recorded time
Aedes sp.	256	954 seconds
An. maculatus	105	486 seconds
An. dirus	110	474 seconds
An. harrisoni	150	612 seconds
Armigeres sp.	261	1084 seconds
Culex sp.	67	386 seconds
Mansonia sp.	4	14 seconds
An. minimus	8	61 seconds
An. barbirostris	9	13 seconds

In Chapter 4 we also identified MFCCs as effective features for multiple machine learning algorithms. MFCCs are computationally efficient and have been a popular choice in mosquito detection and other acoustic scene classification tasks (Li et al., 2017b; Barchiesi et al., 2015). However, the DCT step leads to less human-interpretable features than MFCCs, as shown in Figure 5.2, where the mel-frequency spectrum (Figure 5.2d) better preserves the harmonic structure in the spectrogram (Figure 5.2b) in comparison to the MFCC spectrum (Figure 5.2c). Furthermore, the mel-frequency spectrum is also fast to compute and forms a compact representation of the spectrum. Our experiments, including that presented in Section 5.3.4, have shown that the mel-frequency spectrum leads to detection performance with no statistically significant difference to results obtained with the MFCC.

We therefore use the mel-frequency spectrum to construct time-frequency representations. For an audio clip we can compute the mel-frequency spectrum for smaller segments within the clip, and combine them to form a time-frequency matrix. This resultant matrix forms the input space of the subsequent machine learning algorithm (Figure 5.3). As with previous work, we use a CNN with a similar configuration to that in Chapter 4. A schematic representation for the log-mel spectrum is given in Figure 5.3.



(a) Signal

(b) Spectrogram

(c) MFCC

(d) Mel-frequency spectrum

Figure 5.2: An example recording and associated spectral features. Mosquito flight tones are present from 21 s to 31 s for cow capture #242, 35 s to 40 s from cow capture #243, 42 s to 52 s from cow capture #251. Other segments of the recording (including the high amplitude sections) are either background noise or human voices.

5.3.3 Training strategy

In training and evaluating mosquito acoustic detection algorithms, we first randomly remove 50% of the recordings to create the hold-out test dataset. For the remaining recordings, we divide recordings into audio clips with a length of 0.1 seconds, thus creating a relatively large number of samples with which we train the CNN and

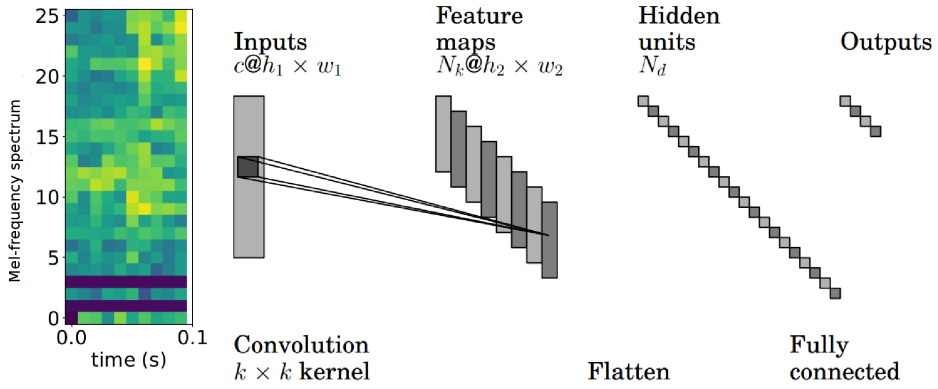


Figure 5.3: Example CNN architecture. Input to the CNN is a mel-frequency spectrum computed from a 0.1 second audio clip with $c = 1$ channel and dimension $h_1 \times w_1$. The CNN has N_k filters with kernel $K \in \mathbb{R}^{k \times k}$ thus it reduces the input dimension to $h_2 \times w_2$ following convolution with each filter. These feature maps are flattened (i.e. vectorised) before being fully connected to N_d hidden units in a dense layer. The last fully-connected layer produces the classification output.

other benchmark algorithms.

As shown in Table 5.2, the dataset is highly imbalanced. To avoid issues with very small data sample sizes, we only use samples from the *Aedes* sp., *An. Maculatus*, *An. dirus*, *An. harrisoni*, *Armigeres* and *Culex* sp.* to evaluate the detection performance of the algorithms, as there is less than 2 minutes of recordings for the other species. Three of these species are known malaria vectors, including *An. maculatus*, *An. dirus* and *An. harrisoni*. Following Li et al. (2017b), we randomly sample the training samples, without replacement, to produce a balanced training set.† In our application, this creates a dataset of close to 2000 samples for each mosquito species. A total of 100 randomised trials were performed so that different training and test datasets were produced among different simulation trials.

5.3.4 Experiment configuration

The input image to the CNN is of dimension 26×10, where 26 is the dimension of the mel-spectrum and 10 is the number of 0.01 second windows within a 0.1

*sp. is an abbreviation to refer to multiple species, *species pluralis*.
†Approaches to imbalanced classification are considered in Sections 5.5 and 6.3.3.

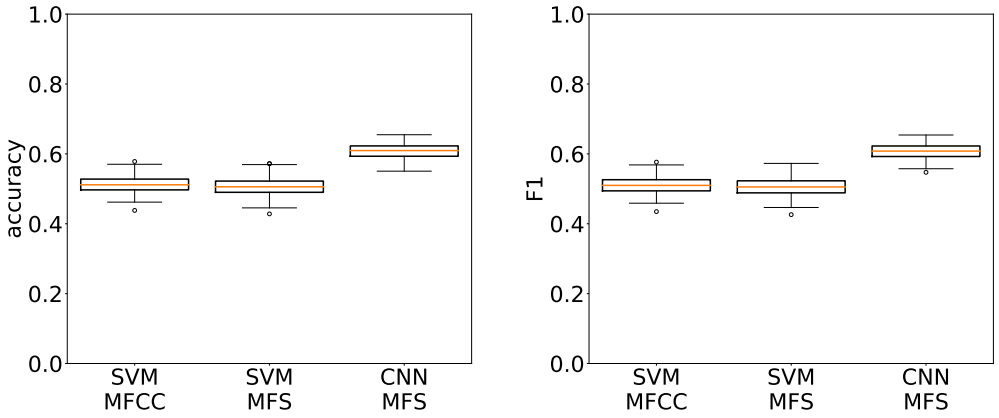
second audio clip. In this initial investigation we adopt a network structure inspired by Simard et al. (2003) which was used for MNIST handwriting digit recognition. There are three convolutional layers: the first layer consists of 8 filters, the second one has 32 filters, and the last one is made up of 64 filters. All filters have 3×3 kernel size. They are followed with one hidden layer of 128 nodes. The ReLU and a dropout rate of 0.3 are used. The neural network is trained using the Adam algorithm (Kingma and Ba, 2014) with a batch size of 256 for 100 epochs.

As the benchmark classifier we choose an SVM using a one-versus-one multi-class classification strategy (Cortes and Vapnik, 1995). The one-versus-one multi-class classification strategy has a simpler data balancing requirement compared to one-versus-rest. As discussed in Li et al. (2017b), MFCC features combined with an SVM obtains the best multi-species classification accuracy among several common acoustic features and off-the-shelf detection algorithms. However, subsequent studies showed that the mel-frequency spectrum leads to similar detection performance and more human-interpretable features. As the SVM requires a feature vector as an input instead of a two-dimensional array, we calculate the mel-frequency spectrum and MFCCs for the SVM from the entire 0.1 second audio clip, to form feature vectors in $\mathbb{R}^{26\times1}$. *

5.3.5 Results

Figure 5.4 shows the out-of-sample classification accuracy and F_1 score of the compared algorithms, respectively. The CNN exhibits significantly better classification performance, in terms of both classification accuracy and F_1 score over the SVM algorithms. We observe no significant difference between results obtained with MFCCs and the mel-frequency spectrum. As shown in Figure 5.2, the mel-frequency spectrum is able to better preserve the harmonic structure of the mosquito flight tone than MFCCs. Figures 5.5a and 5.5b show confusion matrices for the mel-

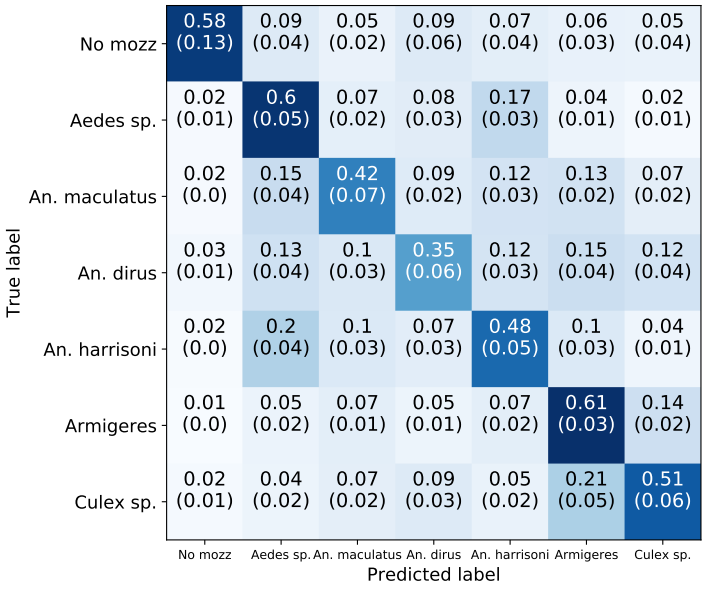
*Stacking shorter frames to form $\mathbb{R}^{260\times1}$ instead led to a similar performance loss as previously encountered and demonstrated in Table 4.1.



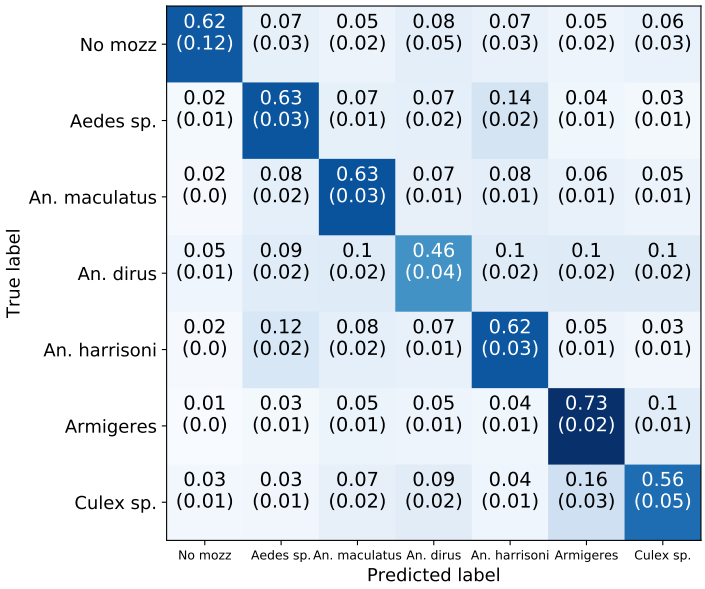
(a) Classification accuracy (b) F_1 score

Figure 5.4: Boxplots showing out-of-sample classification accuracy and F_1 scores across 100 randomised trials.

frequency SVM and CNN, respectively. The CNN exhibits a significant increase in mean accuracies over the SVM for every mosquito species in this experiment.



(a) SVM



(b) CNN

Figure 5.5: Confusion matrices of out-of-sample classification performance for models operating on the mel-frequency spectrum. The first value in each entry is the corresponding mean value among 100 simulation trials, while the value in the parenthesis reports the standard deviation.

5.3.6 Initial investigation discussion

This initial investigation has reported classification results with labelled data from a subset of recordings in which labels were obtained by data tagging from project team members. Further work includes data which has been labelled via citizen scientists on the Zooniverse citizen science platform. We discuss this in more detail in Section 5.5. Working with such data necessitates algorithms capable of handling crowdsourced labels which are given over different time scales to existing data labels – and this is a topic of active current research. Our current results and research directions in this field are reported in Chapter 7.

Finally, in this section we have shown the classification results for each species sampled. If we know a priori which species are more likely to vector malaria, this changes our cost function associated with detection per class. This leads to the next section, where we work with cost-sensitive classification. Later in the thesis we return to the concept of cost sensitivity by introducing an explicit utility function in Chapter 6.

5.4 Cost-sensitive detection with variational autoencoders for acoustic sensing

As outlined in Section 2.3.4, the ability to control different types of errors can be important for environment sensing tasks. For example, a small false negative rate is critical in hazard event detection or rare bird species detection. Furthermore, due to storage limitations it would be desirable to achieve a small false positive rate if the acoustic sensor stores recordings only when it detects events of interest.

In this section we adopt a simple, yet, principled Neyman–Pearson approach (Scott, 2007) as a precursor to Chapter 6 which combines utility theory with Bayesian neural networks. The Neyman–Pearson approach is able to select classifier parameters that minimise the false negative rate while keeping the false positive rate below a

pre-specified threshold. A Variational Autoencoder (VAE) (Kingma and Welling, 2013) is used to harness the structure of the features of a large quantity of unlabelled acoustic mosquito data. As the transform is performed only once offline, the reduced feature dimensionality allows for faster execution on low-power embedded devices.

This work was developed in parallel with that of Section 5.2, and hence makes use of the expert-labelled data used throughout Chapter 4.

5.4.1 Method

The VAE is a variational inference technique using a neural network for function approximations (Kingma and Welling, 2013). It has become one of the most popular choices for unsupervised learning of complex distributions. As a generative model, it assumes that there is a latent variable $\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z})$ that influences the observation $\mathbf{x}$ through a conditional distribution (the probabilistic decoder) $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ parameterised by $\boldsymbol{\theta}$. The variational lower bound on the marginal likelihood of a data point x_i is

$$\log p_{\boldsymbol{\theta}}(x_i) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; x_i) = -D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|x_i)||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|x_i)}[\log p_{\boldsymbol{\theta}}(x_i|\mathbf{z})], \quad (5.1)$$

where D_{KL} is the Kullback–Leibler (KL) divergence term and the variational parameter $\boldsymbol{\phi}$ specifies the recognition model (the probabilistic encoder) $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. The variational autoencoder jointly optimises $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ with respect to the variational lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; x_i)$.

The VAE assumes that the latent variable $\mathbf{z}$ can be drawn from an isotropic multivariate Gaussian distribution $p_{\boldsymbol{\theta}}(\mathbf{z}) = N(\mathbf{z}; \mathbf{0}, \mathbf{I})$ where $\mathbf{I}$ is the identity matrix. It is then mapped through a function to approximate the data-generating distribution using neural networks. More specifically, both the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and the decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ are modelled using multivariate Gaussian distributions with diagonal covariance matrices, where the means and variances of the Gaussian distributions are computed using neural networks. A re-parameterisation trick is needed to optimise the KL divergence, by making the network differentiable so that back-propagation

can be performed. We refer the readers to Kingma and Welling (2013) for more details.

5.4.2 2ν-SVM

To utilise the cost-sensitive framework, we make use of the ν-SVM (Schölkopf et al., 2000). The SVM takes as input the latent variable $\mathbf{z}$, which allows us to write the maximum margin solution as the following quadratic programming problem:

$$\min_{\mathbf{w},b,\epsilon,\rho} \frac{1}{2}||\mathbf{w}||^2 - \nu\rho + \frac{1}{N}\sum_{n=1}^{N}\epsilon_n, \tag{5.2}$$

$$\text{subject to } \epsilon_n \geq 0, \rho \geq 0, t_n y(\mathbf{z}_n) \geq \rho - \epsilon_n, \forall n.$$

This formulation allows for a straightforward interpretation of the parameters in the minimisation. The parameter $\nu \in [0, 1]$ serves as an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors (Schölkopf et al., 2000). The parameter ρ influences the width of the margin.

To allow for cost-sensitive classification, Chew et al. (2001) proposed the 2ν-SVM by introducing an additional parameter to produce an asymmetric error (Davenport et al., 2010):

$$\min_{\mathbf{w},b,\epsilon,\rho} \frac{1}{2}||\mathbf{w}||^2 - \nu\rho + \frac{\gamma}{N}\sum_{n\in I_+}\epsilon_n + \frac{1-\gamma}{N}\sum_{n\in I_-}\epsilon_n, \tag{5.3}$$

$$\text{subject to } \epsilon_i \geq 0, \rho \geq 0, t_n y(\mathbf{z}_n) \geq \rho - \epsilon_n, \forall n.$$

I_+ denotes the set of data elements with the label $t_n = +1$, and I_- denotes the set of data elements with the label $t_n = -1$. We can express the problem in a different way by introducing parameters $\nu_+ \in [0, 1]$ and $\nu_- \in [0, 1]$ to replace ν and $\gamma \in [0, 1]$ (hence the name 2ν-SVM). ν_+ and ν_- bound the fractions of margin errors and support vectors from each class (Davenport et al., 2010).

5.4.3 Cost-sensitive ensemble selection

In order to perform cost-sensitive classification, we need an objective function to gauge the performance of a classifier with different cost constraints. Scott (2007) proposed a scalar performance measure, the *Neyman–Pearson measure* $\hat{e}$, that soft-constrained the false positive rate of the classifier to be below a target value α, while minimising the false negative rate:

$$\hat{e} = \frac{1}{\alpha} \max\{\hat{P}_F - \alpha, 0\} + \hat{P}_M, \tag{5.4}$$

where $\widehat{P}_F$ and $\widehat{P}_M$ are the empirical false positive rate and the empirical false negative rate, respectively.

We may expect that certain variational autoencoder configurations are more effective in representation learning for specific audio signals. Different classifier hyper-parameters, e.g. ν_+, ν_- and the detection threshold of detector outputs, may suit different cost objectives to varying degrees. We can apply the ensemble selection framework proposed in Caruana et al. (2006) to form a model ensemble that constitutes the most informative base models which minimise $\hat{e}$ in Equation (5.4). In our context, the chosen base models are autoencoders with different network structures, and the cost-sensitive SVM with varying hyper-parameters. Note that the base classifiers are not restricted to the 2ν-SVM in the cost-sensitive ensemble selection framework. Any classifier with probabilistic outputs can be adopted, as different types of errors can be controlled by varying the detection threshold*. After the model selection, the classification decision in the test stage will be a majority vote among the committee of the selected Q best models. The ensemble selection architecture is shown in Figure 5.6.

*or, equivalently, choosing an operating point on the Receiver Operating Characteristic (ROC) curve (Streiner and Cairney, 2007)

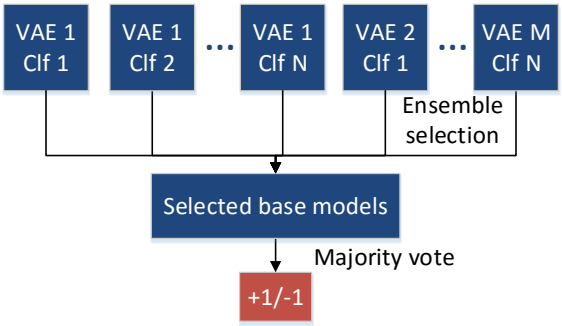


Figure 5.6: The ensemble selection approach. There are M different VAE network candidate structures and N different combinations of classifier hyper-parameter values.

5.4.4 Data

We conducted experiments using the *Culex quinquefasciatus* dataset (Section 4.5) collected in the initial phase of the HumBug project. The dataset used here includes 57 audio recordings with a total length of around 50 minutes.* 736 seconds of these recordings contain sound of the *Culex quinquefasciatus* mosquitoes. We split these recordings into short audio clips, or what we call samples, with a duration of 0.1 seconds. Labels are given to each of these short audio clips. We resample to obtain a balanced dataset. Hence, the dataset contains 7360 positive samples (with mosquito sound) and 7360 negative samples (no mosquito sound). A random sampling approach (Li et al., 2017b), which randomly samples audio clips without replacement in the dataset, was used to form the training set with 10 % of total samples. The remaining 90 % samples are used for testing. We note that this renders the use of supervised deep neural networks impractical due to the low number of samples available for training, differentiating this work from Chapter 4.

5.4.5 Parameter values

For the purpose of this section, we set our target false positive rate threshold to 0.1, i.e. we would like to minimise the missed detections while maintaining the

*This is the same dataset used throughout Chapter 4.

false alarm rate to be below 10 %. How to set the exact value of this threshold is a recurring topic which we return to in Section 6.4.4.1 by interpreting the problem as a utility. Our preference on the error distribution will depend on the cost of making errors, which is a multi-faceted problem. For example, we can consider the cost of transmitting data from the smartphone, once a recording has been made. With sufficient network coverage and bandwidth, we will be inclined to accept a greater number of false positives, as the cost of uploading recordings with false positives (and hence no mosquito) is acceptable to us. On the other hand, if data transmission is very expensive, we would prefer to only upload data we are more certain about containing mosquitoes. The final parameter values will depend on the intended target application. For this reason, any values chosen for thresholds or utilities are preliminary, and are due to be updated in light of new pilot studies which have been scheduled for 2020 and 2021. Our goal in the thesis is to provide frameworks which accommodate for any arbitrary user preference.

The SVM with the RBF kernel and the MFCC features serves as the benchmark algorithm, which was chosen on the basis of its surveyed performance in low-dimensional data, as well as computational efficiency when benchmarked against many similar classifiers such as RFs, NB, etc. (Kiskin et al., 2017).

Candidate parameter values used to form the model library include the 2ν-SVM parameters:

$$\gamma : 2^{-5}, 2^{-3}, \ldots, 2^{5},$$

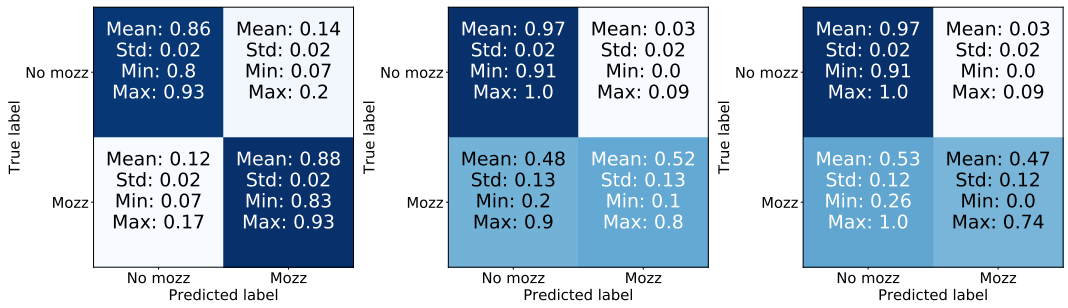
$$\nu_+/\nu_- : 10^{-5}, 3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 0.001, 0.003, 0.01, 0.03, 0.1, 0.2, 0.3, \ldots, 1$$

The MFCC feature is 13-dimensional. To reduce feature dimension while maintaining discriminative power, we use the VAE for feature re-representation. The candidate dimensions of the latent variable of the VAE include 3 and 5, while the number of nodes in the hidden layer we trial are either 10 or 50. Note that these choices are representative only and can vary for different datasets.

We initialise parameters of the VAE using a normal distribution with a standard deviation 0.01. Adam (Kingma and Ba, 2014) is used in the optimisation of the VAE. The cost-sensitive detector forms an ensemble of 100 base models to produce the final prediction.

5.4.6 Results

We performed 100 simulation trials, in which each trial differs due to the random seed initialisation, hence producing different data partitions and initial parameter values. We see from Figure 5.7b and Figure 5.7c that the cost-sensitive SVM (CSSVM) framework is able to maintain the false positive rate below 0.1 in all simulation trials. The SVM with the MFCC features has impressive performance (Figure 5.7a), but it fails to control the false positive rate to be below our target value.



(a) MFCC, SVM (b) MFCC, CSSVM (c) MFCC+VAE, CSSVM

Figure 5.7: Confusion matrices of the test errors with different features and classifiers. "Mozz" is the positive class.

Although the VAE leads to a slightly lower true positive detection rate (comparing Figure 5.7b and 5.7c) for this dataset, the VAE provides a flexible mechanism to reduce feature dimension in the ensemble model. The reduction in the model size can be attractive in porting the model into embedded devices, though if we do not require further feature compression of the MFCCs, it is not worth the detection performance degradation.

5.4.7 Scaling to a larger dataset

In this section we have considered VAE feature re-representations, which are helpful for low-power embedded systems. In the upcoming section, we shift our focus to a more data-rich scenario, in which we utilise deep neural networks for the training phase. As the main focus is for offline training, this relaxes the need for reducing the feature dimensionality. We are presented with new issues of class imbalance which we begin to address by re-weighting the cross-entropy function, and later re-visit in Chapter 6. We also consider the effect of weighting the cross-entropy on the balance of error type.

5.5 HumBug Zooniverse: a crowdsourced acoustic mosquito dataset

For the remainder of the thesis we shift our focus to dealing with larger volumes of data, which was collected as described in Section 5.2 and 5.3. This section unifies work that has been presented at the *Machine Learning for the Developing World* workshop at *NeurIPS 2019*, and has also been accepted at the *International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2020* (Kiskin et al., 2019, 2020).

In recent years, deep learning has become widely used for bioacoustic classification tasks. To aid the potency of existing algorithms and encourage the development of more data-driven approaches, we release a new dataset of mosquito audio recordings. With over a thousand contributors, we obtained 195,434 labels of two second duration, of which approximately 10 percent signify mosquito events. We showcase the metadata, and present an application of a convolutional neural network trained on log-mel (Hershey et al., 2017) spectrogram features. This represents the basis of many neural networks currently used in this problem domain (Deng et al., 2014; Längkvist et al., 2014). Furthermore, we suggest methods for dealing with the

difficulty of this real-world data, namely ways of dealing with data imbalance and label coverage. We hope this will become a vital resource for those researching all aspects of malaria, and add to the existing audio datasets for bioacoustic detection and signal processing.

5.5.1 Related work

The closest existing example data, to the best of our knowledge, is the *Wingbeats* pseudo-acoustic data, which consists of 279,566 short recordings (of approximately one second duration) (Fanioudakis et al., 2018). One potential issue with this data is that it is collected via optical sensors, which may not translate well to detection with low-cost acoustic sensors. In order to provide a new dataset that may be more suited to helping with the detection of mosquitoes in the developing world, our data is collected via conventional microphones that are found in low-cost mobile phones. Collecting via these microphones allows both greater participation from those who already live in malaria-ridden areas, and the possibility of providing a method for detection in people's homes. Specifically, these smartphones can be placed in bed net corners, where mosquitoes tend to congregate, to allow the autonomous collection of data. This is a suggested strategy which as of May 2020 is explored as part of the HumBug project. There has already been research that uses similar technology (Li et al., 2017b; Mukundarajan et al., 2017) and we believe that the dataset we release will only help further to solve these kinds of issues.

Furthermore, we highlight the utility of crowdsourcing labels, where we have gathered labels for our data via the Zooniverse platform. The Zooniverse platform is a useful tool for labelling data (see Figure 5.8) and has seen success in multiple other projects that help with machine learning for the developing world, such as disaster response and earthquake detection (Simpson et al., 2014).

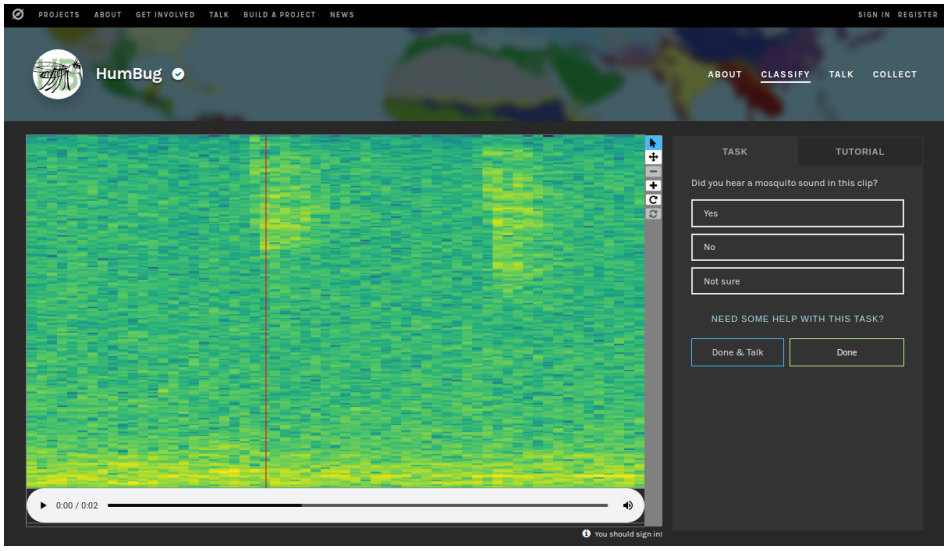


Figure 5.8: User interface for the classification page of the Zooniverse HumBug project, found at https://www.zooniverse.org/projects/yli/humbug/. A short-time Fourier transform spectrogram representation is offered above the audio file.

5.5.2 Data acquisition

The recordings* presented are from four sources, which we break down as follows:

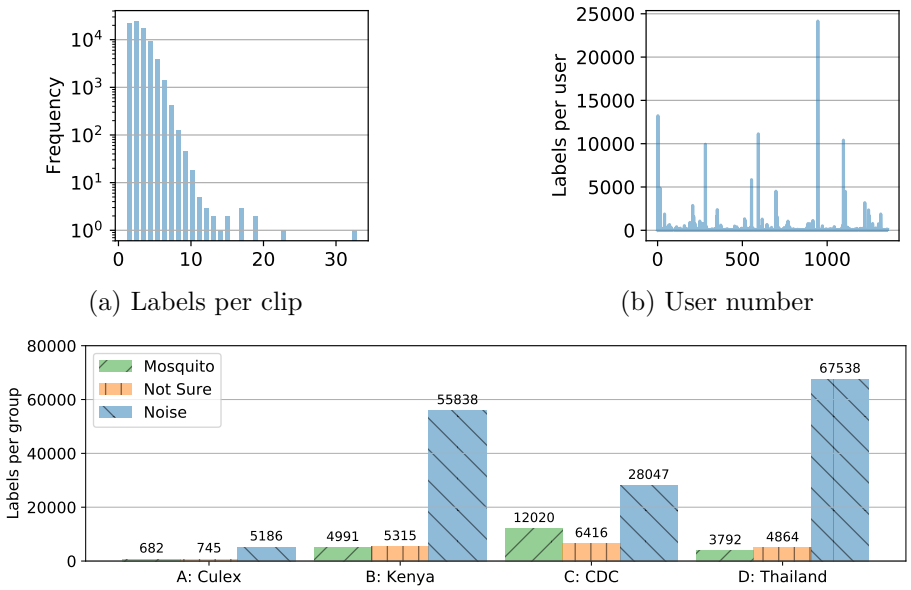
Group A consists of laboratory-based mosquito colonies held in the UK (Oxford), providing acoustic data for vector species *Culex quinquefasciatus*, a vector of the West Nile virus. These culture cages contained both male and female insects.

Group B was acquired from laboratory-based mosquito colonies in the USAMRU-K, providing acoustic data for *Anopheles gambiae*, the primary vector of malaria in Africa.

Group C was made with the recording of multiple species at the insectary at the CDC, Atlanta, USA, including *Aedes aegypti* and *Aedes albopictus*, vectors of yellow fever and dengue fever respectively.

Group D is formed from recordings of wild captured mosquitoes (including *Anopheles barbirostris* and *Anopheles maculatus*, Asian vectors of malaria), sampled

*See the GitHub repository at https://github.com/HumBug-Mosquito/ZooniverseData for the `csv` file and corresponding `wav` audio data.



(a) Labels per clip

(b) User number

(c) Responses of raw votes available in the database, split by recording group

Figure 5.9: Statistics of the crowdsourced Zooniverse dataset. The dataset is comprised of four sources of data, labelled "**A**", "**B**", "**C**", and "**D**", described in Section 5.5.2. The total number of classifications made is 195,434. This is made on 80,101 overlapping 2 second chunks, each with a unique `audio_id`, which create a 22 hour dataset of unique audio. 57,710 (72 %) of the audio samples contain more than one label, of which 10,487 (18 %) contain disagreement. In total, approximately 1 in 10 recordings contains an audible mosquito, with the distribution given in (c).

form the Pu Teuy Village, Sai Yok District, Kanchanaburi Province, Thailand. This site is a field site of Kaetsart University, Bangkok.

Figure 5.9c shows the proportion of data labelled as each class within the four sources of recordings which constitute the whole dataset. We note groups **A, B** and **D** show a similar fraction (approx 5 to 10 %) of mosquito events, with group **C** containing the highest relative frequency of clearly audible mosquitoes (approx 25 %).

5.5.3 Data labelling

With large-scale field deployment, the number of recordings requiring data tagging was beyond the capacity of experts and researchers in the HumBug project. We

hence resorted to the power of crowdsourcing, creating a project on Zooniverse, the world's largest citizen science platform (Simpson et al., 2014), to solicit labels from over a million volunteers. Volunteers listen to 2 second sound clips and can see the corresponding spectrograms, then give their decisions in a set {`Yes`, `No`, `Not Sure`} (Figure 5.8). The total number of participants for this release of the dataset is 1,316. The audio clips each overlap by 1 second (50 %) in order of the `audio_id` in the `csv` file within each group. This ensures that despite the unique `audio_id` with its associated votes, each section of audio is covered by at least two labels.

Figure 5.9b shows the number of classifications made by each individual volunteer. While the majority of participants only clicked through a few examples, the remaining users supplied votes to cover 80,101 audio segments. We represent the resulting number of labels per audio clip with a logarithmic scale histogram in Figure 5.9a. The first few bins are healthily populated, with a logarithmic decline with the increase in number of labels. Of the 80,101 overlapping audio clips, 57,710 (72 %) contain more than one label, of which 10,487 (18%) contain disagreement.

To showcase an example use of this data, we use a simple majority voting scheme in Section 5.5.4 as a baseline. We convert the "`Not Sure`" labels into a numerical value of 0.5, take the mean of all the votes given to each audio clip and round to the nearest integer (0 or 1). We resolve tiebreaks in favour of the mosquito class, though note that we could in principle set an acceptable threshold for what data to use for training, or use classifiers that accept probabilistic input. To utilise the multiple votes that the label overlap provides, we aggregate votes from overlapping audio segments to form a new dataset of one second clips. Classification tasks using this dataset benefit from the aggregated labels per clip, resulting in higher rates of both precision and recall. We supply this processed, simplified, dataset in a separate `csv` file with the fields {`path`, `subject_set`, `Yes`, `No`, `Not Sure`}, where `path` is the path to these new higher resolution (shorter label duration) label audio clips, and {`Yes`, `No`, `Not Sure`} are columns that count the number of these occurrences per each unique one second clip.

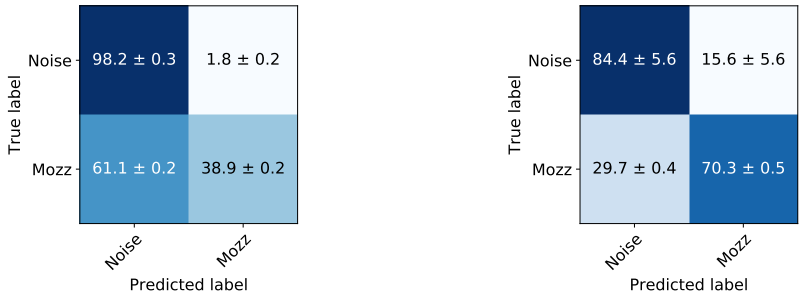
5.5.4 Baseline

We show an example classification use of the data in the `iPython` notebook `baseline.ipynb` found in the repository*. We employ a convolutional neural network, following similar architectures used previously in Chapter 4, with two layers, with filter sizes 3×3 and unit stride length.† To deal with the data imbalance, we use a weighted cross-entropy with the class-weights, $\mathbf{w} = [1, 10]$ equal to the inverse of the relative frequency of each class, calculated from the aggregated majority labels (an approximation can be viewed in Figure 5.9c). Each audio recording of one second duration is transformed into the log-mel spectrogram domain, as this feature space is commonly found to perform well in the literature (Gemmeke et al., 2017; Hershey et al., 2017). Each transformed data sample is treated as a 2D image with dimensions 128×11. The number of rows corresponds to the number of log-mel features, and the width contains 11 samples (a result of a 0.1 second window length for the feature transform with padding). The feature-transformed data, with its corresponding majority label, is given in $\{\mathbf{X}, y\} = \{\texttt{data\_mel}, \texttt{label}\}$. We resolve tiebreaks in favour of the positive (mosquito) class.

We also include a copy of the data transformed to MFCCs, as the more salient lower-dimensional representation can help algorithms such as SVMs learn models (Kamruzzaman et al., 2007). The code provides the core functionality such that the user may choose whichever representation is most convenient.

The data $\{\mathbf{X}, y\}$ is then tenfold split with `sklearn`'s `train_test_split` function. For reproducibility, we include the vector of random states used to partition the data in the code. The combination of majority voting and the log-mel features with the CNN results in the mean confusion matrices with the standard deviations given in Figure 5.10. We choose confusion matrices as metrics, as these are informative in the presence of a class imbalance as the matrices allow us to visualise both false positive and false negative errors. Figure 5.10a shows the results of training

*https://github.com/HumBug-Mosquito/ZooniverseData
†Full details and hyper-parameters are given in the supplemented code.



(a) Cross-entropy class weights $w_0 = 1, w_1 = 1$

(b) Cross-entropy class weights $w_0 = 1, w_1 = 10$

Figure 5.10: Mean confusion matrices of classifications made with the CNN baseline on ten folds of the dataset, given in the form of $\mu \pm \sigma$, where μ is the mean, and σ is the standard deviation. The two subfigures show two weighted cross-entropies which encode our utility in detecting either class. The trade-off of false negative and true positive rate is evidenced when we change the weighting of each class in the loss function.

the CNN with equal class weights, $w_i = 1$. This allows us to detect 39 % of the mosquito events while incurring a 2 % false positive rate. The effect of more heavily weighting the under-represented mosquito class, with $w_1 = 10$, is to increase the rate of true positives at the expense of a greater number of false positives (Figure 5.10b). We take a more sophisticated approach to data imbalance using a Bayesian utility framework in the following chapter.

5.6 Chapter summary

This chapter consists of research output most specific to the HumBug smartphone app, field deployment, and data collection. In Section 5.2 we have highlighted the technology we used for data collection, namely the MozzWear android app. We have also shown that it is possible to distinguish between 7 species of mosquito with high accuracy from recordings made on low-cost smartphones with MozzWear.

We have described further data collection in Thailand, and have shown the possibility to discriminate between multiple species captured in the field, in Section 5.3, with reasonable accuracy. Furthermore, we utilised a similar CNN as in Chapter 4 with 40

log-mel spectrogram features to balance computational efficiency with classification performance. When benchmarking against the version of MozzWear currently in deployment (May 2020), we have shown a consistent improvement in classification score which warrants research into the revision of the smartphone app.

Section 5.4 then introduced the notion of asymmetric cost, where the user may want to determine algorithm behaviour by setting thresholds for error types such as the false positive rate. An example use of this is to prevent overly aggravating the user and conserving limited hardware resources such as storage. To aid in this task, a variational autoencoder is pre-trained offline to reduce the dimension of the latent space required for online classification.

Section 5.5 is dedicated to the processing of data labels which have been collected with a concerted citizen science effort. We reported on the release of a public dataset, and provided a baseline implementation for classification. We note that the data collected is noisy, highly imbalanced, and furthermore we may want to consider asymmetrical cost. As a baseline, we apply a similar neural network architecture as found in Chapter 4 and Section 5.3. We made use of the additional data by adapting the architecture to use a greater depth and higher feature resolution. In conjunction with the baseline we have shown how weighting the cross-entropy can affect the compromise between false positives and true positives.

As a result of the challenges introduced by the combination of an asymmetric cost, high label uncertainty, as well as imbalanced classes, we now look to Bayesian neural networks and their combination with Bayesian decision theory in the next chapter. We leverage the classification performance strength of deep learning methods, while taking advantage of the principled framework of Bayesian decision theory.

Chapter 6

Loss-calibrated Bayesian neural networks for noisy acoustic mosquito detection

6.1 Overview

In this chapter we focus on a Bayesian approach that explicitly incorporates uncertainty within its formulation. With the greater availability of data for this work, we are in a position to improve upon limitations encountered in Chapter 5. With the cost-sensitive SVM of Section 5.4, we lack principled handling of utility functions. Furthermore, due to the crowdsourcing of labels through our citizen science project of Section 5.5, there is a high level of label uncertainty, as well as class imbalance, introduced.

In order to overcome these limitations we move to a Bayesian framework, whereby data imbalance can be re-cast as a Bayesian decision theory task. To retain the powerful classification performance offered by traditional deep learning models, we look for ways of combining deep neural networks with Bayesian decision theory. Therefore, in Section 6.2 we introduce Bayesian decision theory for neural networks. We then give an overview of the mathematical foundations of inference in Bayesian

neural networks in Section 6.3. Section 6.4 then returns to our mosquito detection problem with these new tools, and offers more depth in comparison to methods explored with cost-sensitive VAE SVMs in Section 5.4.

6.2 Bayesian decision theory for Bayesian neural networks

Having introduced Bayesian decision theory in Section 2.3.4, we now use it to improve on our ability to make classifications, with regard to an asymmetric utility in the presence of uncertainty. The motivation for selecting the Bayesian decision framework is due to the principled way in which it deals with uncertainty. Any prediction we make should involve the uncertainty in our knowledge over the model parameters, $\boldsymbol{\omega}$. In our scenario, we can think of our confidence in the model parameters as our knowledge over the world we model, which is given by the posterior over the weights.

For our choice of model, we denote the probability vector output as $\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}_i)$. Based on the model vector output, we must decide upon which label to assign to maximise our utility. For a given input $\mathbf{x}_i$ we refer to the decision as $\mathbf{h}_i$, which can take any label assignment $\mathbf{c} \in \mathcal{C}$. For clarity, we list our definitions as follows:

- $\mathbf{h}$ refers to the decision, or label assignment.
- u refers to the utility, which defines our preferences for certain errors.
- Traditionally, a standard prediction from a neural network is given by the argmax over the classes of the vector output $\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}_i^*)$, denoted by $\mathbf{y}_i^*$.

In Bayesian decision theory, the overall process is divided into two separate tasks: probabilistic inference and decision making. Here, we include details of these two tasks.

6.2.1 Probabilistic inference

When we have access to the true posterior, we can think of probabilistic inference as averaging over the model parameters $\boldsymbol{\omega}$ to infer a predictive distribution $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$, which can be shown as the integration:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int_{\omega} p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}, \tag{6.1}$$

where $\mathbf{x}^*$ is our test point, $\mathbf{y}^*$ is the predicted label, and $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ forms our training dataset.

6.2.2 Decision

Having inferred the predictive distribution, in the context of supervised learning for classification, we must make a decision as to what label to assign for a given input $\mathbf{x}^*$. The label we assign both depends on the specific task and the uncertainty.

In this context, we define a utility function, $u\left(\mathbf{h} = \mathbf{c}, \mathbf{y}^* = \mathbf{c}'\right)$ (or $u\left(\mathbf{c}, \mathbf{c}'\right)$), which dictates what we will gain from each decision, $\mathbf{h}$, for each label.

To combine our uncertainty in our decision with a utility function, we average the utility over the predictions $\mathbf{y}^*$ to give the expected utility in assigning a label $\mathbf{h}$ conditioned on a test input $\mathbf{x}^*$:

$$\begin{aligned} &\mathcal{U}(\mathbf{h} = \mathbf{c}|\mathbf{x}^*) \\ &= \sum_{\mathbf{c}'} u\left(\mathbf{h} = \mathbf{c}, \mathbf{y}^* = \mathbf{c}'\right) p(\mathbf{y}^* = \mathbf{c}'|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}). \end{aligned} \tag{6.2}$$

The decision $\mathbf{h}$ that maximises the expected utility is defined as the optimal decision $\mathbf{h}^*$ for the given input $\mathbf{x}^*$ (Section 2.3.4):

$$\begin{aligned} \mathbf{h}^*(\mathbf{x}^*) &= \underset{\mathbf{c} \in \mathcal{C}}{\operatorname{argmax}}\ \mathcal{U}(\mathbf{h} = \mathbf{c}|\mathbf{x}^*) \\ &= \underset{\mathbf{c} \in \mathcal{C}}{\operatorname{argmax}}\ \log\left(\mathcal{U}(\mathbf{h} = \mathbf{c}|\mathbf{x}^*)\right) \end{aligned} \tag{6.3}$$

conditioned on the dataset $\{\mathbf{X}, \mathbf{Y}\}$.

6.3 Bayesian deep learning

6.3.1 Bayesian neural networks

If we are to use Bayesian decision theory for our problem, we require a model that can provide distributions for each section of audio data. Bayesian Neural Networks (BNNs) offer a probabilistic alternative to neural networks by specifying prior distributions over the weights (MacKay, 1992; Neal, 2012). The placement of a prior $p(\omega_i)$ over each weight ω_i leads to a distribution over a parametric set of functions.

The motivation for working with BNNs comes from the availability of uncertainty in its function approximation, $\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x})$. In training, we infer the posterior over the weights $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$, given a prior, $p(\boldsymbol{\omega})$, and the likelihood, $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})$. As with convention, the prior is defined as a multivariate Gaussian distribution. This comes from the distribution's desirable properties as a prior such as symmetry around zero and a preference for small weight values (Cobb, 2020).

To express the likelihood mathematically, we first define a target vector $\mathbf{t}$, which is *one-hot encoded* with a 1-of-K coding scheme, such that all elements are zero except for element k, corresponding to class C_k, which equals one. A softmax function can be used to output a class probability,

$$p(\mathbf{y} = c_k|\boldsymbol{\omega}, \mathbf{x}) = \frac{\exp\{\mathbf{f}^{\boldsymbol{\omega}}_{c_k}(\mathbf{x})\}}{\sum_{c_j} \exp\{\mathbf{f}^{\boldsymbol{\omega}}_{c_j}(\mathbf{x})\}}. \tag{6.4}$$

In two-class classification with logistic regression, the outcome (or *response variable*) can be modelled by a Bernouilli distribution. Similarly, the likelihood function for multi-class classification can be derived from a generalised Bernoulli distribution (also known as the *categorical distribution*) (Bishop, 2006, p. 209). The categorical distribution is a discrete probability distribution that describes the possible results of a random variable that can take on one of K possible categories, with the probability of each category separately specified (Murphy, 2012, p. 35). With our

neural network model, the likelihood over the entire dataset $\{\mathbf{X}, \mathbf{Y}\}$ is a function of our softmax output, and target variable as follows:

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(\mathbf{y}_n = c_k|\boldsymbol{\omega}, \mathbf{x}_n)^{t_{nk}}. \tag{6.5}$$

Maximising the likelihood function gives the Maximum Likelihood Estimate (MLE) of $\boldsymbol{\omega}$. The usual optimisation objective during training is the negative log-likelihood. For the categorical distribution, this is the *cross-entropy* (Bishop, 2006, p. 209):

$$-\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log p(\mathbf{y}_n = c_k|\boldsymbol{\omega}, \mathbf{x}_n). \tag{6.6}$$

Multiplying the likelihood with a prior distribution $p(\boldsymbol{\omega})$ is, by Bayes' theorem, proportional to the posterior distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})$. Maximizing the posterior gives the Maximum A Posteriori (MAP) estimate of $\boldsymbol{\omega}$. Computing the MAP estimate has a regularising effect and can prevent overfitting. The optimisation objectives here are the same as for MLE plus a regularisation term arising from the logarithm of the prior.*

Both MLE and MAP give point estimates of parameters, akin to traditional approaches in deep learning, where commonly the cross-entropy loss is optimised (with regularisation for MAP). If we instead had a full posterior distribution over parameters, we could make predictions that take weight uncertainty into account. We illustrate how to do this in the following section.

6.3.2 Inference in Bayesian neural networks

This section offers an overview of inference in Bayesian neural networks. The objective is to infer the predictive distribution over the output $\mathbf{y}^*$. With the posterior distribution over the weights, $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$, we can calculate the predictive

*The term appears as $\log\left(p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})\right) = \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) + \log p(\boldsymbol{\omega})$.

distribution, $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$, as:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})\, p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\boldsymbol{\omega}, \tag{6.7}$$

in which the parameters have been marginalised out. This is equivalent to averaging predictions from an ensemble of neural networks weighted by the posterior probabilities of their parameters $\boldsymbol{\omega}$.

In a Bayesian setting, the posterior over the weights is given by Bayes' rule as:

$$p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) p(\mathbf{X}|\boldsymbol{\omega}) p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X}) p(\mathbf{X})} = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X})}, \tag{6.8}$$

as $p(\mathbf{X}|\boldsymbol{\omega})$ reduces to $p(\mathbf{X})$, as the input $\mathbf{X}$ is not dependent on the weights $\boldsymbol{\omega}$. To calculate the posterior we also need to calculate the marginal likelihood $p(\mathbf{Y}|\mathbf{X})$ as follows:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) p(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}. \tag{6.9}$$

This integral is often intractable, for the exception of when the prior over the weights is conjugate to the likelihood $p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\omega})$. This conjugacy is chosen to make the integral tractable, but may not be most appropriate to the data.

One possibility to deal with the intractable integral is to sample from Equation (6.8) directly. We begin by noting that the posterior is proportional to the likelihood multiplied by the prior,

$$p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) p(\boldsymbol{\omega}). \tag{6.10}$$

If we draw a set of samples S, we can obtain an expectation of the predictive distribution (Cobb, 2020, Ch. 3):

$$\mathbb{E}_{p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})} \left[\mathbf{Y}^* = \mathbf{y}^*|\mathbf{x}^*\right] \approx \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}^{(s)}). \tag{6.11}$$

An alternative approach to sampling is to peform *variational inference*. The goal is to obtain an analytical approximation to the posterior by reformulating the intractable integral as an optimisation of a tractable variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$,

which is dependent on variational parameters $\boldsymbol{\theta}$ (Jordan et al., 1999). This can be done by minimising the KL divergence (Kullback and Leibler, 1951),

$$\mathrm{D}_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\|p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})\right) = -\int_{\boldsymbol{\omega}} q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log\left(\frac{p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})}{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}\right) \mathrm{d}\boldsymbol{\omega}, \tag{6.12}$$

with respect to $\boldsymbol{\theta}$. Minimisation of the KL divergence with respect to $\boldsymbol{\theta}$ results in an optimal set of variational parameters $\boldsymbol{\theta}^*$, such that $q_{\boldsymbol{\theta}^*}(\boldsymbol{\omega})$ is as similar as possible to $p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})$ under this particular divergence. We refer the reader to Cobb (2020, Ch. 3) for the advantages and challenges associated with the minimisation of Equation (6.12).

6.3.3 Loss-calibrated approximate inference in Bayesian neural networks

We are now in the position to introduce the loss-calibrated Bayesian neural network (LCBNN), which uses the variational inference framework, and furthermore augments the optimisation problem to incorporate our utility function during the training phase of a Bayesian neural network (Cobb et al., 2018).

The marginal expected utility $\mathcal{U}(\mathbf{H}|\mathbf{X})$ for the entire input data using a conditional independence assumption over the inputs is defined as:

$$\begin{aligned}\mathcal{U}(\mathbf{H}|\mathbf{X}) &:= \int_{\boldsymbol{\omega}} \prod_j \mathcal{U}(\mathbf{h}_j|\mathbf{x}_j,\boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y}) \mathrm{d}\boldsymbol{\omega} \\ &= \int_{\boldsymbol{\omega}} \mathcal{U}(\mathbf{H}|\mathbf{X},\boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y}) \mathrm{d}\boldsymbol{\omega},\end{aligned} \tag{6.13}$$

with the assumption that given the model parameters, the decision depends only on the input $\mathbf{x}_j$.

In order to maximise the expected utility, one must integrate with respect to the parameters $\boldsymbol{\omega}$ and optimise with respect to the decisions $\mathbf{H}$. However, due to the intractability of the integration, a lower bound to the log expected utility is maximised instead:

$$\log\left(\mathcal{U}(\mathbf{H}|\mathbf{X})\right) \geq \mathcal{L}(q(\boldsymbol{\omega}),\mathbf{H}). \tag{6.14}$$

This lower bound is approximated and reformulated as the *standard optimisation objective* loss for a BNN with an additional penalty term:

$$\mathcal{L}(\boldsymbol{\omega}, \mathbf{H}) \propto \underbrace{-\sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \hat{\boldsymbol{\omega}}_i) + \|\boldsymbol{\omega}\|^2}_{\text{Equivalent to standard dropout loss}} \underbrace{-\sum_i \left(\log \sum_{\mathbf{c}} u\left(\mathbf{h}_i, \mathbf{c}\right) p(\mathbf{y}_i = \mathbf{c}'|\mathbf{x}_i, \hat{\boldsymbol{\omega}}_i) \right)}_{\text{Our additional utility-dependent penalty term}}, \tag{6.15}$$

where $\hat{\boldsymbol{\omega}}_i \sim q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$.

Details for both learning the parameter weights and the decisions are given in Algorithm 3, where MC dropout is performed by drawing Bernoulli-distributed random variables $\boldsymbol{\epsilon}$ and applying the re-parameterisation $\boldsymbol{\omega} = \boldsymbol{\theta}\text{diag}(\boldsymbol{\epsilon})$, where $\boldsymbol{\theta}$ are the approximating distribution parameters (weight matrices' means) (Gal and Ghahramani, 2016).

Algorithm 3 LCBNN optimisation as given by Cobb et al. (2018)

Given dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, utility function $u(\mathbf{h}, \mathbf{y})$ and set of all possible labels $\mathcal{C}$
Define learning rate schedule η
Randomly initialise weights $\boldsymbol{\omega}$
repeat
  Sample S index set of training examples
  for $i \in S$ **do**
    for t from 1 to T **do**
      Sample Bernoulli-distributed random variables $\boldsymbol{\epsilon}^t \sim p(\boldsymbol{\epsilon})$ ▷ for each each $\mathbf{x}_i$ we sample T dropout masks $\boldsymbol{\epsilon}^t$
      $\mathbf{y}_i^t = \mathbf{f}^{g(\boldsymbol{\omega}, \boldsymbol{\epsilon}^t)}(\mathbf{x}_i)$ ▷ Perform a stochastic forward pass with the sampled dropout mask $\boldsymbol{\epsilon}^t$ and $\mathbf{x}_i$
    end for
    $\mathbf{h}_i^* \leftarrow \underset{\mathbf{h} \in \mathcal{H}}{\text{argmax}} \frac{1}{T} \sum_t u(\mathbf{h}, \mathbf{y}_i^t)$ ▷ Choose class $\mathbf{h} = \mathbf{c} \in \mathcal{C}$ which maximises average utility
  end for
  Calculate the derivative w.r.t. $\boldsymbol{\omega}$:

$$\nabla \boldsymbol{\omega} \leftarrow -\frac{1}{T} \sum_{i \in S} \frac{\partial}{\partial \boldsymbol{\omega}} \log p\left(\mathbf{y}_i|\mathbf{f}^{g(\boldsymbol{\omega}, \boldsymbol{\epsilon}_i)}(\mathbf{x}_i)\right) + \frac{\partial}{\partial \boldsymbol{\omega}} \text{KL}\left(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})\right) - \frac{1}{T} \sum_{i \in S} \frac{\partial}{\partial \boldsymbol{\omega}} \log \mathcal{U}\left(\mathbf{h}_i^*|\mathbf{x}_i, \boldsymbol{\omega}\right)$$

  with $\boldsymbol{\epsilon}_i \sim p(\boldsymbol{\epsilon})$ a newly sampled dropout mask for each i.
  Update $\boldsymbol{\omega}$: $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \eta \nabla \boldsymbol{\omega}$
until $\boldsymbol{\omega}$ has converged

6.4 Acoustic mosquito detection with loss-calibrated Bayesian neural networks

We extend the work of Cobb et al. (2018) by formulating appropriate utility functions for audio classification tasks by using our HumBug mosquito detection example of Section 5.5. We show how a Bayesian form of convolutional neural networks combined with Bayesian decision theory are well suited to acoustic mosquito detection and how utility functions can result in certain desirable behaviours for real-world deployment.

We use BNNs to infer predictive distributions over outputs and incorporate this uncertainty as part of a decision-making framework. We train a LCBNN in conjunction with user-defined utility matrices on challenging crowdsourced data. In our experiments we demonstrate how the traditional approach of up-weighting the less prevalent class (here mosquito events) in a weighted cross-entropy loss function leads to worse performance than that of the LCBNN. Furthermore, we highlight the importance of incorporating Bayesian decision theory into our detection pipeline. Our results show that our theoretically sound approach to detection in audio data warrants the use in a variety of other decision-driven applications.

6.4.1 Problem statement

We adopt a loss-calibrated variational inference framework to maximise the expected utility of mosquito detection from their acoustic flight tones. Performing acoustic classifications, or classifications in any other domain more generally, often involves an asymmetric cost associated with different incorrect decisions. Furthermore, making a decision to minimise a cost is made ever more difficult in the presence of a data imbalance, which arises due to the difficulty of gathering data for some events (such as the detection of a rare mosquito species). For example, in detecting mosquitoes that transmit malaria (Li et al., 2017a,b), we are more concerned with

models concentrating their predictive power to distinguish disease-prevalent species with higher accuracy. These species are, however, under-represented in the dataset. In this work we present an application where a monitoring system is required to detect the presence of a mosquito in real-time (Li et al., 2017b). This involves two operational modes, where we are in turn interested in minimising a false positive rate, and maximising the recall of events with a trade-off in accuracy (Section 6.4.4.1).

Our contribution is the novel application of principled Bayesian decision theory to learning and decision making under uncertainty in acoustic signal processing, with the following key points:

1. We encode our preferences on error type using utility functions (e.g. false positive versus false negative).
2. We show how Bayesian decision theory provides a theoretically sound way of detecting mosquitoes, whilst ensuring our preferences are met.
3. We highlight the superior performance of the LCBNN and show how it is especially suited to detection applications with asymmetrical utility functions.

The remainder of this work is structured as follows. Section 6.4.2 gives an overview of prior work. Section 6.4.3 describes the challenges we face with data imbalance. Section 6.4.4 then provides details of the dataset, and our corresponding utility functions. Section 6.4.5 reports the results and Section 6.4.6 provides additional insight into the loss-calibrated approach. We conclude the chapter in Section 6.5.

6.4.2 Prior work on audio detection

Detecting the presence of a mosquito in audio data falls within the broader area of signal detection. Within speech recognition, previous work in applying machine learning techniques has seen approaches evolve from using HMMs for making

classifications on *phenomes* or MFCCs (Juang and Rabiner, 2005), to using CNNs for end-to-end learning (Sainath et al., 2015). Similarly to computer vision, audio event recognition has undergone a paradigm shift from hand-crafted representations to models which also learn end to end (Dieleman and Schrauwen, 2014). Recently, much of the success in this area has been seen from applying CNNs (Piczak, 2015; Salamon and Bello, 2017), where the task is to classify signals in the spectral feature space (such as short-time Fourier and log-mel transforms). However, the vast majority of this work has focused on deterministic approaches to classification, where uncertainty over predictions is not factored in. To account for uncertainty, we adopt the approach of using current state-of-the-art methods in signal classification and place them in a Bayesian framework. This allows us to make detections in the presence of uncertainty by using Bayesian decision theory.

6.4.3 The challenge of data imbalance

In classification tasks a dataset is deemed imbalanced when certain classes are disproportionately represented. For example, in our application the data contains 178,094 labels, where 12 % are mosquito events, and 88 % are non-mosquito events. This imbalance presents itself as a challenge for most machine learning techniques. Specifically for deep learning, a common workaround is to rely on weighting the cross-entropy, such that errors on less common classes are more heavily penalised relative to errors on more common classes. This aims to overcome the unequal distribution of class labels in the data (e.g. Mostajabi et al. (2015); Xu et al. (2014)). However, in real-world applications, especially for crowdsourced data, these labels are also noisy and therefore by simply placing weights over these classes we may be up-weighting noisy labels (Cobb et al., 2018, Sec. 4). Up-weighting labels in this manner can produce models that are erroneously overconfident in their deterministic output. Instead, we move to a Bayesian framework, where the use of Bayesian decision theory provides a more theoretically sound way of making detections in the presence of uncertainty.

6.4.4 Experiment

We now compare the LCBNN to two baselines for detecting mosquitoes in the crowdsourced data of Section 5.5, which was collected as part of the HumBug project. The data consists of four sources containing a mixture of laboratory and field recordings. The SNR is challenging to the human listener, as indicated by the high rate of disagreement between user classifications.

We begin by defining our utility functions that encourage the desired behaviour for the detection system.

6.4.4.1 Utility function

The detection system has one passive mode (Table 6.1), and one active mode (Table 6.2). We design a utility function to capture our preference for behaviour in each mode. In the passive mode, the system is listening for a potential positive candidate. On detection, the system records the audio and uploads to a central database. In this mode, in order not to disturb the user and conserve hardware resources, we wish to only trigger a detection with high confidence, and therefore a low false positive rate. We therefore penalise false positives more strongly than false negatives by increasing utility of a false negative, FN. In the active mode, we are concerned with not missing any potential lethal parasites, especially when in an area at a higher risk of malaria. We want to increase the recall of positive events, at the expense of incurring a higher false positive rate. Therefore, for this scenario we increase the relative utility of the false positives FP (i.e. encourage a preference for the mosquito class).

6.4.4.2 Model

We employ a deep Bayesian convolutional neural network, by adding appropriate dropout layers following similar architectures used previously in Chapter 4 and 5.

Table 6.1: **Utility A** matrix to favour precision (passive mode). Applies to scenarios where false positives are too costly. In our application, this is the requirement not to trigger too many recordings and disturb the user to the point where the app is disabled for being too intrusive or consuming too much power.

		Decision	
	Utility A	Noise	Mosquito
True	Noise	100	0
	Mosquito	20	100

Table 6.2: **Utility B** matrix to favour recall (active mode). Applies to scenarios where false negatives have potentially lethal consequences. In our application, this is the requirement not to miss detection of a deadly species.

		Decision	
	Utility B	Noise	Mosquito
True	Noise	100	50
	Mosquito	0	100

To showcase the problem with the weighted cross-entropy, we use a more reserved, as well as more aggressive, class re-weighting. Each audio recording of two second duration is transformed into the log-mel spectrogram domain, as this feature space is commonly found to perform well in the literature (Gemmeke et al., 2017; Hershey et al., 2017). Each transformed data sample is treated as a 2D image with dimensions $h = 128, w = 21$. The number of rows, or height h, corresponds to the number of log-mel features, and the width w, contains 21 samples (a result of a 0.1 second window length for the feature transform with padding). The Bayesian CNN is trained on the feature-transformed data, with its corresponding one-hot-encoded majority label, $\{\mathbf{X}, y\}$.

6.4.5 Results

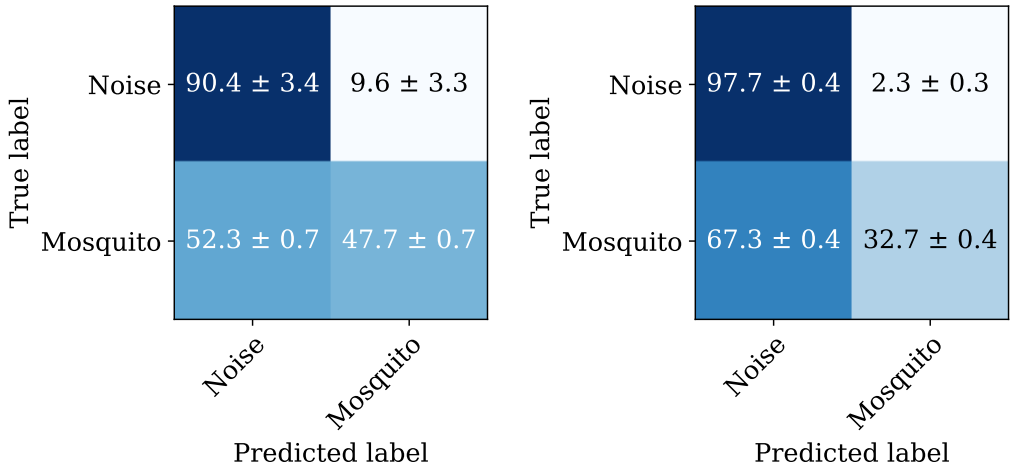
In passive mode we set the utility matrix to have values $FP = 0$, $FN = 20$ to set our preference for minimising the false positive rate. In active mode we set the values to $FP = 50$, $FN = 0$ to maximise the recall of the mosquito detections while maintaining acceptable precision. Table 6.3 displays the results of applying

Table 6.3: Results over 10 folds of cross-validated data for two utilities. Acc. corresponds to the results after the integration. We show that the LCBNN achieves both the highest accuracy in decisions as well as the highest utility with consistently lower standard deviation in comparison to standard and weighted cross-entropy methods.

Models	Acc.	Exp. Util.
	Min FPR: **Utility A**	
Standard	86.4 ± 0.6	0.887 ± 0.007
Weighted [1,2]	85.8 ± 1.2	0.879 ± 0.013
Weighted [1,5]	84.1 ± 1.0	0.859 ± 0.011
LCBNN	$\mathbf{86.9 \pm 0.2}$	$\mathbf{0.891 \pm 0.001}$
	Max recall: **Utility B**	
Standard	84.4 ± 2.3	0.873 ± 0.009
Weighted [1,2]	80.1 ± 4.8	0.861 ± 0.019
Weighted [1,5]	51.1 ± 8.6	0.734 ± 0.038
LCBNN	$\mathbf{85.8 \pm 0.8}$	$\mathbf{0.879 \pm 0.003}$

Bayesian decision theory with all the models. The accuracy in the table is calculated in accordance with the optimal Bayesian decision for each model when integrating over the utility function. The expected utility shows the mean utility of selecting these decisions across the dataset when comparing to the known labels. The key result is that the LCBNN outperforms all baselines in both metrics, but notably in expected utility.

To highlight the poor performance of weighting the cross-entropy, we compare this approach to that of the LCBNN in the confusion matrices of Figure 6.1. When using the passive utility function that discourages false positives, the weighted-cross entropy is unable to capture our preferences as well as the LCBNN. In fact, when switching to the LCBNN, the false positive rate reduces from 10 % to 2 % and the expected utility correspondingly increases. Therefore, in this case applying a large weight to the less frequent class has had a detrimental effect on the overall expected utility over the test sets for the weighted cross-entropy model.



(a) Cross-entropy with weights $\mathbf{w} = [1, 5]$, $\mathcal{U} = 0.859$.

(b) LCBNN, $\mathcal{U} = 0.891$.

Figure 6.1: Confusion matrices of the decisions with respect to the utility function of Table 6.1 with $FP = 0, FN = 20$, and their corresponding expected utility, $\mathcal{U}$. The LCBNN achieves a lower false positive rate and therefore a higher utility.

6.4.6 Utility calibration

In this section we illustrate how the expected utility in making classifications varies depending on the model and utility function. In order to accomplish this, we construct a synthetic observation by combining an example noise sample additively with a mosquito signal, for which labels were known a priori. We vary the SNR linearly as a fraction along the x-axis (from purely noise to purely mosquito). We then display the expected utility of predicting the mosquito class conditioned on the input $\mathbf{X}$, $\mathcal{U}(\mathbf{h} = \text{mosquito}|\mathbf{X})$. This quantity is evaluated by the BNNs before the models have knowledge of the true label.

We display the results for the weighted cross-entropy model and the LCBNN in Figure 6.2. The figure shows how the LCBNN is better calibrated to our preferences. When the utility is built to encourage false positives, as shown in the lower plot, the LCBNN consistently expects higher utility in the medium to high SNR range for detecting a mosquito in comparison to the weighted cross-entropy model. In the passive mode, the utility has calibrated the LCBNN to deem it too risky to infer a

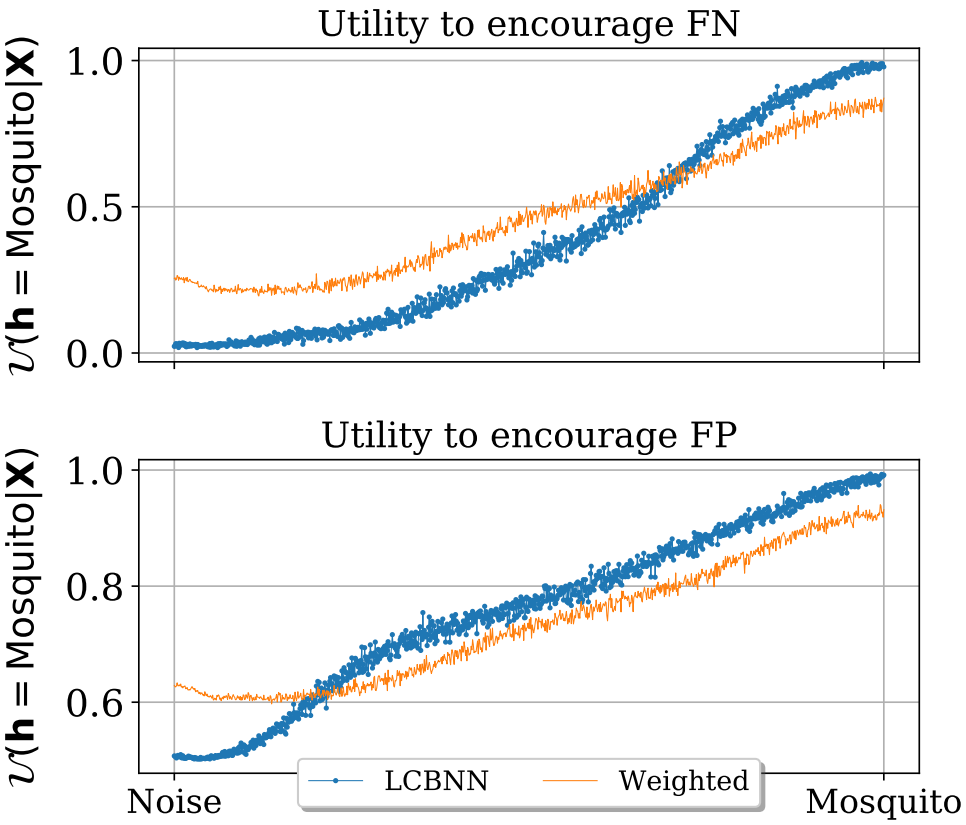


Figure 6.2: A comparison of the effect of SNR on the predicted utility of choosing the mosquito. **Top:** The LCBNN's behaviour is better calibrated as it only expects a high utility for high SNR. **Bottom:** Above a threshold, the LCBNN expects higher utility, aligned with our preference for reduced false negatives. By jointly training the model with the utility, the LCBNN is able to calibrate its behaviour to our preferences in a way the weighted model cannot.

mosquito is present until the SNR has reached a higher threshold. We compare to the weighted cross-entropy model, which does not demonstrate this well-calibrated behaviour.

6.5 Chapter summary

In this chapter we have tackled the research problem arising from the acquisition of data through crowdsourcing, as outlined in Section 5.5. We have shown a principled approach to dealing with the uncertainty that arose when detecting mosquitoes,

while maintaining the performance benefits of deep learning frameworks. We first introduced the mathematical foundations of inference in Bayesian neural networks, and described how we can combine Bayesian decision theory to form a LCBNN in Sections 6.2 and 6.3.

In Section 6.4 we have showcased a novel application of a LCBNN. We have shown that our method outperforms baseline Bayesian neural network methods when making decisions based on asymmetric utility functions. These utility functions encode the users' preferences explicitly. Two example scenarios for detecting mosquitoes using a Bayesian CNN have been considered, both aimed at future real-world deployment strategies. In one, we have aimed to favour reducing false positives, while in the other we wish to improve the recall of events whilst also maximising the accuracy of detections. We have shown that we achieve a higher expected utility in both of these scenarios, when compared to making decisions from a regular BNN trained with a weighted cross-entropy.

We have further demonstrated how the LCBNN is better calibrated to the task by demonstrating performance as SNR varies. We have therefore demonstrated how to perform acoustic detection with BNNs and believe that more work in audio signal processing may benefit from utilising our pipeline. Furthermore, we have shown how the data imbalance problem can be re-phrased as one of Bayesian decision theory and we have introduced utility functions that best represent our preferences.

In the following chapter we tackle a further research challenge that presented itself from the aggregation of crowdsourced data, and various sources of labels, namely combining knowledge from a mixture of label resolutions.

Chapter 7

Machine learning with variable label resolution data

7.1 Overview

In this chapter we focus on a further challenge arising from crowdsourcing labels over sections of data. Crowdsourcing, via the Zooniverse citizen science platform (see Section 5.5), allows us to collect large numbers of human-curated labels. However, to ensure a reasonable throughput, and to avoid volunteer loss due to task difficulty, exact onset and offset timing was not requested. Instead, the data was labelled over two second intervals into the binary categories: "no mosquito heard" and "likely mosquito sound"[†]. This means that intervals labelled as a "mosquito" may only contain a mosquito audio event for a fragment of the labelled data section. Although this label resolution mismatch is a common problem in many applications, it is not well accommodated for in typical machine learning approaches. Indeed, most methods in machine learning do not provide a convenient mechanism for training on inputs supplied with variable resolution data (i.e. a different size of feature vector input) or variable resolution target labels. This becomes a problem when we wish to train a model jointly on expert (*fine* labels, with onset and offset timing) and crowdsourced data (*weak* labels). We show that naïvely scaling crowdsourced

[†]as well as the "not sure" category, which is discussed in Section 5.5.

labels to mimic expert labels (e.g. by converting a two-second label into twenty 0.1 second labels) results in an excess of incorrect target labels and hence degrades the performance of the model.

In this chapter we develop a novel framework that enables an automated approach to learning from datasets that contain sparse quantities of expertly labelled data, larger quantities of weakly (or *coarsely*) labelled data and a large volume of unlabelled data. We develop a nested loop method, with a Kernel Density Estimator (KDE) at its core, to super-resolve the abundant low-quality data labels, thereby enabling effective training of a second algorithm (in this case a CNN). We demonstrate two key results: a) The KDE is able to super-resolve labels more accurately, and with better calibrated probabilities, than well-established classifier baselines; b) Our neural network, trained on super-resolved labels created from the KDE, achieves an improvement in F_1 score of 22.1 % over the next best baseline system in our candidate problem domain.

7.2 Introduction

Finely labelled data are crucial to the success of both supervised and semi-supervised approaches to classification algorithms. In particular, common deep learning methods (LeCun et al., 2015) typically require an abundance of data samples to train effectively (Krizhevsky et al., 2012; Rolnick et al., 2017). Often, datasets lack a high density of fine, high-resolution labels, as producing them is extremely costly (in our example, requiring exact marking of start and end times of audio events). The typical distribution of data in such problems is such that there are scarce quantities of expertly labelled data, large quantities of weakly labelled data and a high volume of unlabelled data. Here, *weak* labels refer to labels that indicate one or more events are present in the sample, although do not contain the information as to the event frequency, nor to the exact location of occurrence(s) (as illustrated in Section 7.4.2, Figure 7.2). Our goal therefore is to improve classification performance in domains

with variable quality datasets.

Our key contribution is as follows. We propose a framework that combines the strengths of traditional machine learning techniques with more recent deep learning methods. Using a Bayesian approach, we generate pseudo-fine labels from weak labels by making use of the statistics of the data. The probabilistic pseudo-fine labels can then be used to more effectively train a neural network.

We demonstrate our results on an audio dataset which, although synthetic, mimics a realistic distribution of data labels, for which ground truth values are known, to allow exact quantification of each element in the classification pipeline. Our event signal consists of a section of (known) mosquito audio recording, embedded in non-stationary sections of ambient (forest) noise. Our challenge is thus to detect mosquito onset and offset times within the data. As the dataset is synthesised, this enables us to experiment across a range of signal-to-noise ratios.

The remainder of this chapter is organised as follows. Section 7.3 gives the context of our work in the literature. Section 7.4 discusses the structure of the framework as well as the baseline classifiers we test against. Section 7.5 describes the datasets we use and the details of the experiments carried out. Section 7.6 presents and discusses the experimental results. We conclude with a chapter summary in Section 7.7.

7.3 Background

We address particularly the domain of audio data, where the aim is to detect the onset and offset times for sound events in an audio recording and associate a label with each of these events. This task is known as Sound Event Detection (SED) and has many applications such as surveillance (Foggia et al., 2015), biodiversity monitoring (Piczak, 2015), healthcare (Goetze et al., 2012) and in smart cars (Mesaros et al., 2017).

With the rapid development in computational power and software, deep learning methods have become the main approach to solve the SED task (Cakir et al., 2015; Cakır et al., 2017). These supervised learning approaches ideally require large quantities of finely labelled data where the onset and offset times of events have been annotated. However, as emphasised, collecting these fine labels is very time consuming, so the set of labelled recordings is often limited to a few minutes or hours (Mesaros et al., 2016; Giannoulis et al., 2013). On the other hand, weakly labelled data takes less time to annotate manually, since the annotator has to mark only the active sound event classes and not the exact event time boundaries. This method of labelling is common to crowdsourcing tasks. Our crowdsourcing application of Section 5.5 forms the main motivation for this research, as well as the basis for our signal.

Methods that can exploit such weak labels effectively enable learning over larger quantities of data. Current approaches tackle this problem with the added constraint that there is no finely labelled data available. Hence, the problem is re-framed as a Multiple Instance Learning (MIL) problem (Amores, 2013): instead of receiving a set of instances (frames in a recording) which are individually labelled, the learner receives a set of labelled bags, each containing many instances. A bag is positive if it contains at least one positive instance, and negative otherwise. Although the labels of the bags are known, the labels of the instances in the bags are unknown.

In Tseng et al. (2017), the MIL framework is approximated using a time-distributed CNN with a global max-pooling layer to predict the temporal locations of each event. Adavanne and Virtanen (2017) propose a stacked convolutional and recurrent neural network architecture with two prediction layers for the same task. Furthermore, Xu et al. (2017a) use an Attention Neural Network (ANN) to predict onset and offset times from weakly labelled data.

Although these approaches allow supervised methods to be trained using low-resolution, weakly labelled data, to the best of our knowledge the models do not support a mechanism to augment the models with higher-resolution fine labels. In

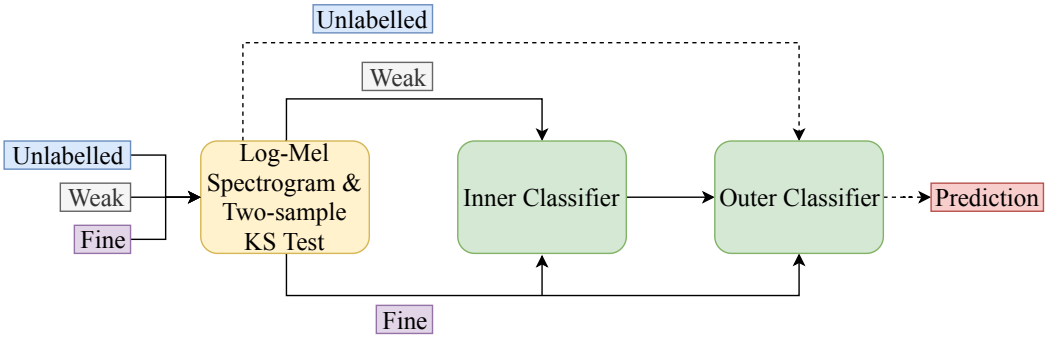


Figure 7.1: Framework comprising a feature extraction & selection layer, an inner classifier and an outer classifier. The arrows represent data flows.

an attempt to avoid such shortcomings, we concentrate on a method which enables the estimation of pseudo-fine targets from the lower resolution weakly labelled data. The approach we take is described in more detail in the following section.

7.4 Methodology

7.4.1 Framework overview

Our framework is separated into an inner and outer classifier in cascade, as shown in Figure 7.1. For the inner classifier we extract features from the finely and weakly labelled audio data using the two-sample Kolmogorov–Smirnov (KS) test for features of a log-mel spectrogram (Section 7.4.2). We train our inner classifier, the Gaussian KDE (Section 7.4.3), on the finely labelled data and predict fine labels for the weakly labelled data.

For the outer classifier we extract the feature vectors from an unlabelled audio dataset using the log-mel spectrogram. We then train our outer classifier, a CNN, (Section 7.4.4) on the finely labelled data and the resulting pseudo-finely labelled data output by the Gaussian KDE. The details of our data and problem are found in Section 7.5.

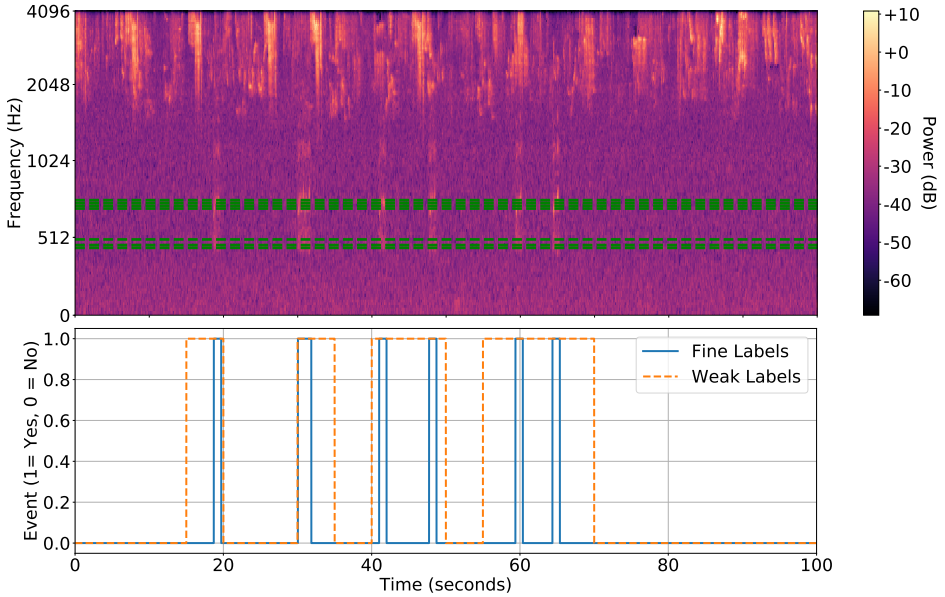


Figure 7.2: Top: log-mel spectrogram of 100 seconds of audio data at a signal-to-noise ratio of −15 dB. The KS-selected features are shown as green dashed lines (see text for details). Bottom: the corresponding fine and weak labels for the log-mel spectrogram.

7.4.2 Feature extraction and selection

We continue to use the log-mel spectrogram (as in Figure 7.2) due to its simplicity, but also as it is widely seen as the gold standard in feature representation for audio data (Hayashi et al., 2017; Kong et al., 2019). The input signal is divided into 0.1 second windows and we compute 128 log-mel filterbank features. Thus, for a given 100 seconds of audio input, feature extraction produces a 1000×128 output.

The two-sample KS test (Ivanov and Riccardi, 2012) is a non-parametric test for the equality of continuous, one-dimensional probability distributions that can be used to compare two samples. This measure of similarity is provided by the KS statistic which quantifies a distance between the empirical distribution functions of the two samples. The KS statistic provides a rapid, yet robust, measure of feature saliency.

The two-sample KS statistic is defined as

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)|, \tag{7.1}$$

where $F_{1,n}(x)$ and $F_{2,m}(x)$ are the empirical distribution functions of the first sample of size n and the second sample of size m respectively, and sup is the supremum function (which over a discretised space is merely the maximum function). The null distribution of the statistic is calculated under the null hypothesis that the samples are drawn from the same distribution. The null hypothesis is rejected at a level α if

$$D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{nm}}, \tag{7.2}$$

where $c(\alpha) = (-\frac{1}{2}\ln\alpha)^{\frac{1}{2}}$.

We use the KS test to select a subset of the 128 log-mel features that are maximally different between the two classes. This reduced feature space is then used as a canonical observation space to input into the classification pipeline. We choose the N features with the largest KS statistics, where N is chosen empirically (as detailed later) to ensure computational speed in analysis, yet provide competent performance. Figure 7.2 illustrates that the process to find maximally different feature pairs correctly chooses frequencies of interest. For example, if the noise file is concentrated in high frequencies (as in Figure 7.2), the KS feature selection process chooses lower harmonics of the training signal (a mosquito flying tone) as features to feed to the algorithms. Conversely, for low-frequency dominated noise, higher audible harmonics of the event signal are identified.

7.4.3 Gaussian kernel density estimation

Kernel density estimation or Parzen estimation (Scott, 2015; Parzen, 1962) is a non-parametric method for estimating a d-dimensional probability density function (pdf), $f_{\mathbf{X}}(\boldsymbol{x})$, from a finite sample $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, $\boldsymbol{x}_i \in \mathbb{R}^d$, by convolving the empirical density function with a kernel function $K(\boldsymbol{u})$. The resulting estimated pdf is

$$\hat{f}_{\mathbf{X}}(\boldsymbol{x} \mid \mathcal{D}; \mathbf{H}) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i), \tag{7.3}$$

where $K_{\mathbf{H}}(\boldsymbol{u}) = |\mathbf{H}|^{-\frac{1}{2}} K(\mathbf{H}^{-\frac{1}{2}}\boldsymbol{u})$ is a multivariate kernel function and $\mathbf{H}$ is a symmetric, positive definite $d \times d$ bandwidth matrix. We choose the Gaussian kernel

$$K_{\mathbf{H}}(\boldsymbol{u}) = (2\pi)^{-\frac{d}{2}} |\mathbf{H}|^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}\boldsymbol{u}^T \mathbf{H}^{-1}\boldsymbol{u}\Big), \tag{7.4}$$

where the bandwidth matrix $\mathbf{H}$ plays the role of the covariance matrix.

The choice of the bandwidth matrix $\mathbf{H}$ is the single most important factor affecting the KDE's accuracy since it controls the amount and orientation of smoothing applied to the empirical distribution function (Wand and Jones, 1994). There are many techniques for choosing the bandwidth in a data-driven manner (Cao et al., 1994; Turlach, 1993). We choose Silverman's *Rule of Thumb* (Silverman, 1986):

$$\sqrt{\mathbf{H}_{ii}} = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} \sigma_i n^{-\frac{1}{d+4}}, \tag{7.5}$$

where σ_i is the standard deviation of the i^{th} variable and $\mathbf{H}_{ij} = 0$ if $i \neq j$. Our choice is based on its computational efficiency of $\mathcal{O}(1)$. Furthermore, Silverman's Rule of Thumb tends to smooth the estimation (Páerez et al., 2006), and so when applied to non-stationary noise it can obtain robust estimations.

We then use Bayes' theorem to calculate the posterior over the classes C_k. We define classes C_1 as the event class and C_0 the noise class (i.e. non-event).

$$p(C_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_0)p(C_0) + p(\boldsymbol{x}|C_1)p(C_1)}, \tag{7.6}$$

where we replace $p(\boldsymbol{x}|C_k)$ of Equation 7.6 by our kernel density $\hat{f}_k(\boldsymbol{x}|\mathcal{D}_k; \mathbf{H}_k)$.

7.4.4 Convolutional neural network

We use a CNN and dropout with probability $p = 0.2$ (Srivastava et al., 2014). Our proposed architecture, given in Figure 7.3, consists of an input layer connected sequentially to a single convolutional layer and a fully connected layer. The CNN is trained for 30 epochs with SGD (Bottou, 2010), and all activations are ReLUs. We use this particular architecture due to constraints in data size (Kiskin et al., 2018) and therefore have few layers and fewer parameters to learn.

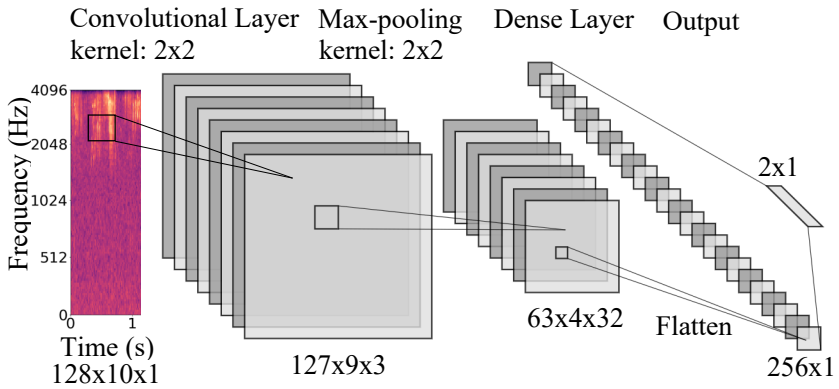


Figure 7.3: The CNN architecture. Spectrogram of mosquito recording fed as input to convolutional layer with 32 filters and kernel $2 \times 2 \times 1$. Generated feature maps are down-sampled by a max-pooling layer from 127×9 to 63×4. It is then connected to a dense layer with 256 neurons and finally the output with 2 neurons.

7.4.5 Traditional classifier baselines

We compare our inner classifier, the Gaussian KDE, with more traditional classifiers that are widely used in machine learning: support vector machines using a radial basis function kernel (RBF-SVM) as used in the previous chapters, random forests and a multilayer perceptron, as introduced in Chapter 2. Furthermore, we trial Linear Discriminant Analysis (LDA) (Balakrishnama and Ganapathiraju, 1998) and the Gaussian Naïve Bayes (GNB) classifier (Rish et al., 2001).

An LDA is a parametric, generative model that makes two assumptions. Firstly, the conditional probability density functions $f(\boldsymbol{x}|C_0)$ and $f(\boldsymbol{x}|C_1)$ are both normally distributed with mean and covariance parameters $(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, respectively. Secondly, the LDA assumes homoscedasticity (i.e. $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$) and that the covariance matrices have full rank.

A GNB is a parametric, generative model which makes the 'naïve' assumption that the features are conditionally independent given the class label, $p(\boldsymbol{x}|C_i) = \prod_{j=1}^{D} p(x_j|C_i)$. Bayes' theorem is then employed to calculate the posterior over class C_i.

7.5 Experiments

7.5.1 Description of datasets

The most common scenario where mixed quality labels can be found is in crowdsourcing tasks (Cartwright et al., 2019; Deng et al., 2009; Lin et al., 2014), or in any challenge where data collection is expensive. The crowdsourced dataset of Section 5.5 forms the main motivation for this research. The event signal consists of a stationary real mosquito audio recording with a duration of 1 second. The noise file is a non-stationary section of ambient forest sound. The event signal is placed randomly throughout the noise file at varying signal-to-noise ratios (as defined in Chapter 2), to create quantifiable data for prototyping algorithms. There is a class imbalance of 1 second of event to 9 seconds of noise in the finely labelled data, which we maintain in the weakly labelled and unlabelled datasets. We include 100 seconds of expert, finely labelled data, 1000 seconds of weakly labelled data, and hold out a further 1000 seconds of unlabelled test data. To report performance metrics, we create synthetic labels at a resolution of 0.1 seconds for the finely labelled data, and on the order of 5 seconds for the weakly labelled data. We choose 5 seconds as to allow the labeller to have temporal context when classifying audio as an event or non-event. As the listener is presented randomly sampled (or actively sampled (Houlsby et al., 2011; Naghshvar et al., 2012)) sections of audio data, a section much shorter than 5 seconds would make the task of tuning into each new example very difficult due to the absence of a reference signal. There is therefore a trade-off between adequate resolution for machine learning models, and the extra resources required to produce high-quality labels.*

*It may be that for a fixed resource budget, a greater number of low-quality labelled samples is more useful than a lower number of high-quality samples.

7.5.2 Data pre-processing

In the data pre-processing pipeline we apply the log-mel spectrogram transform to the audio data using the method described in Section 7.4.2. We then normalise each of the 128 log-mel features. We do this so as to map the features to a common scale, without distorting differences in the ranges of values (Guyon and Elisseeff, 2006). This is particularly important for algorithms sensitive to scaling such as SVMs (Hsu et al., 2003).

We further compress the log-mel feature space by choosing 7 features using the two-sample KS test as detailed in Section 7.4.2. This stems from the KDE's requirement of a balance between the ease of computation and modelling whilst maintaining sufficient complexity for ease of class separation in our feature space.*

7.5.3 Experimental design

We evaluate our inner model against the baseline classifiers with four experiments and finally test the overall performance of the framework when utilising the outputs of the various inner classifiers. We make the assumption that the accuracy of the weak labels is 100 %. Our initial investigation has shown that relaxing this assumption reduces overall classification performance. However, this error rate is reasonably consistent between methods and does not affect conclusions drawn from our experiments.† All classifiers therefore make predictions over the class 1 weakly labelled data only. Additionally, the priors we use in Equation 7.6 for our Gaussian KDE model are set such that $p(C_0) = p(C_1) = 0.5$. This is to reflect that, since the audio sample is weakly labelled as 1, our agnostic belief is that all high-resolution sections of the weakly labelled 1 data could equally be fine label class 0 or 1.

*We note that results are not particularly sensitive, within reason, to the number of features retained.

†Investigating the performance difference resulting from weak label inaccuracy is outside the scope of this work.

We hypothesise that generative models, such as the Gaussian KDE, obtain a performance boost from the additional information provided by the coarse class 0 data, as this allows a better model of the class 0 distribution. Conversely, discriminative models such as the SVM, RF and MLP decrease in performance as the decision boundary they create overfits to class 0 data points due to increased class imbalance. We test this hypothesis in Experiments 1 and 2.

7.6 Classification performance

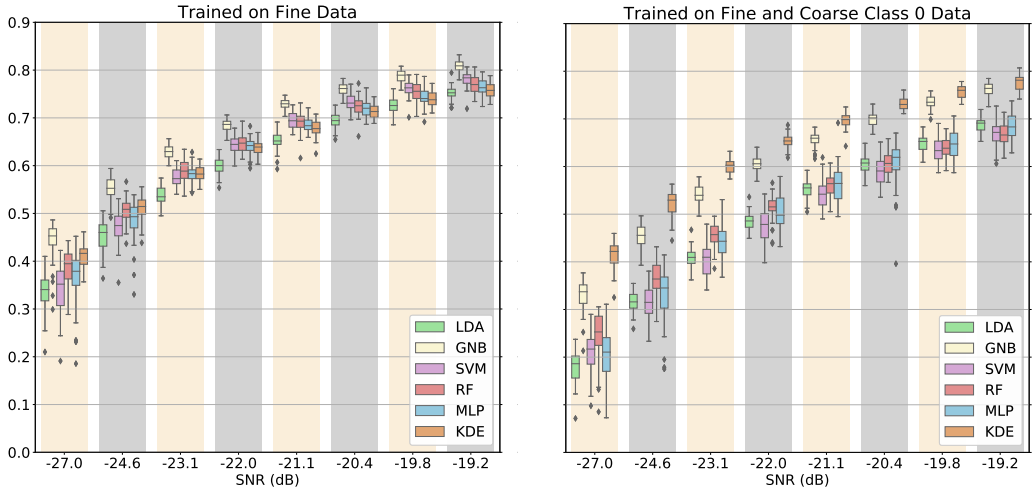
In this section we report the results of five experiments. Figures 7.4, 7.5 and Table 7.1 provide a summary of all the experiments. To account for the class imbalance, the classification performance of each model is evaluated by calculating the recall, its respective precision, and the F_1 score.

Experiment 1: Fine labels For the first experiment we test the classification performance and robustness of the inner loop models by varying the SNR from the threshold of detection to a clearly audible level in 8 discrete steps. For each SNR we synthesise 40 iterations by varying the time location of the injected signals, as well as the random seed of the algorithms. The models are trained on the finely labelled data.

Figure 7.4a shows the models perform similarly, with a small, but significant, competitive edge shown by the GNB. The F_1 score gradually increases as expected from the threshold of detection to more audible SNRs. The decay of performance at the lower SNRs can be partially accounted for by the two-sample KS test used for feature selection failing to choose informative features of interest due to the increased noise floor.*

*We note that removing the KS test does not improve performance at this SNR, as the training fails regardless.

Experiment 2: Fine and coarse labels To test the effect of augmenting the training dataset by including the coarse class 0 labelled data, we follow the same protocol as in the first experiment. We train the models with the finely labelled data and the coarse class 0 labelled data. From Figure 7.4b we observe the effect of including the coarse class 0 data in the training process of the models. The Gaussian KDE obtains a performance boost from the additional information provided by the coarse class 0 data as this allows it to better model the class 0 distribution. However, as hypothesised, the discriminative models such as the SVM, RF and MLP take a hit in performance because the decision boundary that they create overfits to the class 0 data points due to the increased class imbalance. This could be solved partially by re-weighting classes or the cost function (through re-sampling and cost-sensitive learning). However, this is known to cause overfitting (He and Garcia, 2009).



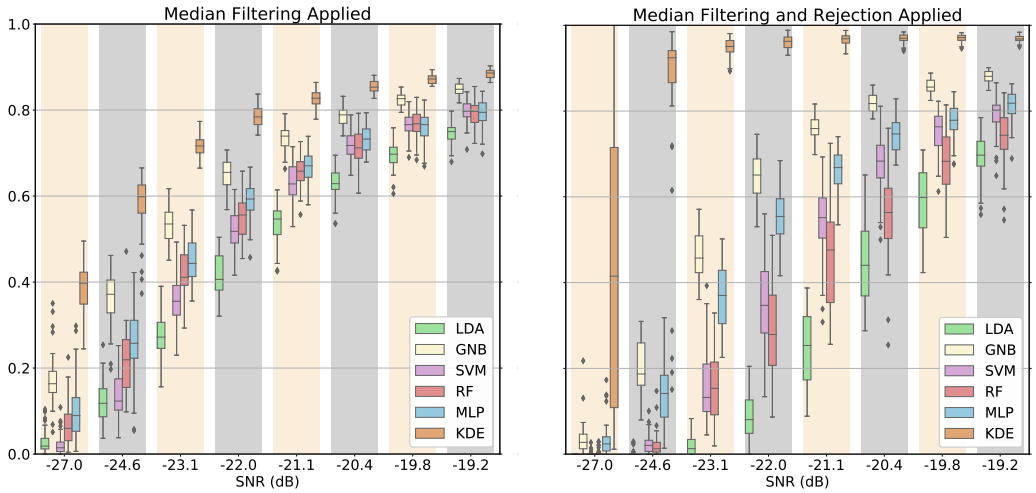
(a) **Experiment 1**, where SNR is varied from threshold of detection to clearly audible for 40 iterations, grouped by SNR. The models are trained only with finely labelled data.

(b) **Experiment 2** following same protocol as Experiment 1, except the models are trained on finely labelled data and the coarse class 0 data.

Figure 7.4: Boxplots of F_1 scores for **Experiments 1** and **2**. The benefit of training on coarse 0 data in (b) for the generative models such as the KDE is demonstrated relative to the baseline scores of (a). The discriminative approaches tend to achieve lower F_1 scores on the whole.

Experiment 3: Median filtering Based on the results from the first and second experiments, we train the LDA, GNB, SVM, RF and MLP on the finely labelled data only, whereas the Gaussian KDE is trained on both the finely labelled data and the coarse class 0 data. After applying median filtering we obtain results as shown in Figure 7.5a. We observe a significant boost in performance for the Gaussian KDE, indicating that applying temporal averaging helps recover the event persistence (there is correlation between neighbouring values in the time-series during an event). However, not every model receives uniform benefit. If the model has high precision, but poor recall of the positive class overall (despite high F_1 scores), median filtering appears to over-smooth the predictions and lower the F_1 score. Specifically, if the model tends to under-predict the number of events, median filtering will degrade performance. On the other hand, the median filter will output a greater number of successive positive labels if a model predicts events at a reasonable (or overly high) rate, which is likely to improve the F_1 score.

Experiment 4: Rejection and median filtering Figure 7.5b shows that the Gaussian KDE predicts better calibrated probabilities than the other baseline classifiers. This is shown by applying rejection (Bishop et al., 1995; Hanczar and Dougherty, 2008) in addition to the median filtering. The rejection window for the output probabilities is $0.1 < p(C_1|\boldsymbol{x}) < 0.9$. The Gaussian KDE improves significantly in performance, especially at the lower SNRs; however it should be noted that the F_1 score is evaluated on the remaining data after rejection. The proportion of data rejected can be seen in Figure 7.6. The Gaussian KDE rejects a large proportion of the data at lower SNRs, showing that the probabilities are at either extreme only when the model is confident in its predictions.



(a) **Experiment 3**, showing the results of the application of a median filter.

(b) **Experiment 4**, application of rejection with $0.1 < p(C_1|\boldsymbol{x}) < 0.9$ after median filtering.

Figure 7.5: Boxplots of F_1 scores for **Experiments 3** and **4**. LDA, SVM, RF and MLP are trained on finely labelled data only whilst the Gaussian KDE is trained on the finely labelled data and coarse class 0 data. The KDE most strongly benefits from median and rejection filtering, due to its reasonable distribution of positive and negative predictive accuracy and principled handling of uncertainty, respectively.

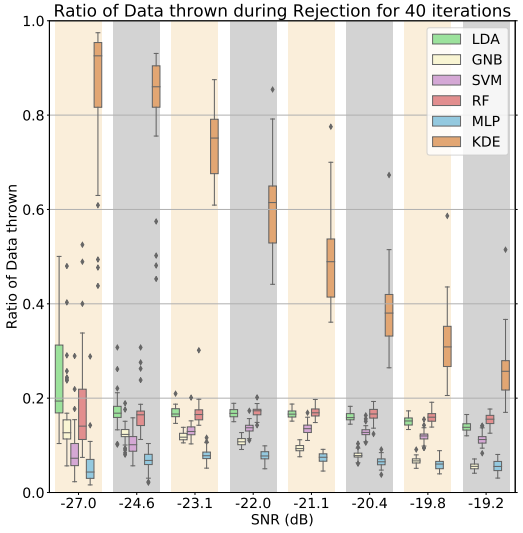


Figure 7.6: Ratio of data rejected during **Experiment 4**, grouped by SNR.

Table 7.1: **Experiment 5**: CNN outer classifier: metrics reported as means ± one standard deviation at an SNR of −19.8 dB for 40 iterations.

Classifier	F_1 score	Precision	Recall
CNN(KDE)	**0.729 ± 0.034**	**0.719 ± 0.029**	**0.744 ± 0.031**
CNN(MLP)	0.435 ± 0.022	0.667 ± 0.026	0.322 ± 0.029
CNN(RF)	0.320 ± 0.031	0.419 ± 0.035	0.259 ± 0.034
CNN(SVM)	0.338 ± 0.024	0.484 ± 0.024	0.259 ± 0.022
CNN(GNB)	0.597 ± 0.023	0.654 ± 0.028	0.549 ± 0.023
CNN(LDA)	0.307 ± 0.027	0.571 ± 0.026	0.210 ± 0.023
CNN(Coarse)	0.174 ± 0.036	0.095 ± 0.031	0.923 ± 0.039
KDE	0.506 ± 0.021	0.518 ± 0.021	0.502 ± 0.024

Experiment 5: Outer stage classification The final experiment tests the overall framework with input to a CNN of pseudo-fine labelled data with median filtering and rejection applied. Table 7.1 shows that using the framework in conjunction with any of the inner classifiers' outputs outperforms a regular CNN trained on the coarse data. Furthermore, training the CNN on the output of the Gaussian KDE significantly improves detection of events by 22.1 % over the best baseline system, the CNN(GNB). We also show that using the strongest inner classifier (KDE) alone results in far lower precision and recall scores compared to the approach advocated here, which sees an improvement by incorporating the CNN into the pipeline with the KDE.

7.7 Chapter summary

In this chapter we have proposed a novel framework which utilises a Gaussian kernel density estimator for super-resolving weakly labelled data, which is subsequently used in conjunction with a CNN to predict fine labels for unlabelled data. Our framework was evaluated on synthetic data and has achieved an improvement of 22.1 % in F_1 score over the best baseline system. We thus highlight the value such label super-resolution provides in domains with sparse quantities of finely labelled data. This work was motivated by the challenges that arise from combining inaccurate labels from crowdsourcing with expert labels available in the HumBug project. We address future research directions, including scaling to real data and consideration of a broader range of models, in Section 8.2.

Chapter 8

Conclusion and further work

8.1 Conclusion

This thesis has primarily focused on machine learning approaches for detecting and identifying mosquito signals from acoustic data. We have provided an in-depth analysis of the problems and solutions specific to mosquito detection. However, these solutions are general in the sense that they apply to wider challenges associated with detection in audio data and beyond. We summarise our findings in this section and suggest further work in Section 8.2.

Our overarching goal has been the detection of mosquitoes from their acoustic signature. In addition to detecting mosquitoes, we have also shown that it is possible to identify the specific species. This information has allowed us to estimate regional risk of malaria resulting from mosquito-vectored infections. We defined our problem statement in Chapter 1. In Chapter 2 we then introduced the definitions and mathematical principles that formed the backbone of the research in this thesis. These research challenges were further highlighted within the context of the HumBug project in Chapter 3. Therein we summarised key events and described our contributions to the community and literature.

Throughout the duration of the project, the constraints of the problem presented numerous challenges, including data sparsity and label uncertainty, compounded by

low signal-to-noise ratios. In the absence of quality and quantity of acoustic data, Chapter 4 presented how a deep learning approach can be used successfully in a data-scarce scenario when allied with wavelet transformations of audio data streams. We have shown that our convolutional neural network remains effective when transferred to other problem domains, such as bird recognition, and furthermore gives interpretable results. By backpropagating the samples that most strongly activate the output of the neural network, we showed that the frequencies that represent the known signal are responsible for high scores predicted on the test set. This indicates that the network has learnt the informative signal contents, such as the tone and harmonics of the bird song, and not peculiarities of the recording such as the microphone noise profile.

Chapter 5 explored work that revolves around data collection and modelling under data and computational constraints. In Section 5.2 we developed a smartphone app that allows efficient detection of mosquitoes with an SVM operating on MFCCs. Thereafter, we deployed the smartphone app in the field in Section 5.3. By collecting data in the field trial, we showed how it is possible to discriminate between six species using a convolutional neural network trained on logarithmically-encoded spectral features. In Section 5.4, we introduced cost-sensitive classification, following the requirement of an asymmetric utility in misclassification. We utilised a parameter with an SVM that thresholds a false positive rate to achieve the classification performance as desired by the user. Furthermore, we conducted experiments in a compressed feature space using a variational autoencoder, resulting in a more efficient data prediction pipeline that retained the classification performance of less compact models. In Section 5.5 we introduced a new crowdsourced mosquito dataset, compromised of four sources of data, containing a mixture of laboratory and field recordings. This composition, of 22 hours of labelled acoustic recordings, has provided a realistic scenario for training machine learning techniques. In order to demonstrate the utility of this new dataset, we have offered a CNN baseline to showcase the ability to construct a mosquito detection system.

In Chapter 6 we have built on the data of Section 5.5 and have addressed a new key theme – uncertainty in classification, where we highlighted that propagating uncertainty in machine learning models is important and suggested mechanisms for how to best incorporate it. With the availability of uncertainty, we showed that it is possible to incorporate user utility into the training objective of a Bayesian neural network. By doing so, we have achieved higher expected utility than with standard Bayesian neural networks that handle inference and decision separately.

In Chapter 7 we have discussed work which combines sources of labels at different temporal resolution, as well as accuracies. We have proposed a novel framework utilising a Gaussian kernel density estimator for super-resolving weakly-labelled data to be fed into a convolutional neural networks for the purpose of predicting fine labels for unlabelled data. Our framework has been evaluated on synthetic data with promising results, which we see as an important avenue of future work.

8.2 Future work

8.2.1 The future of HumBug

In late 2019 HumBug secured funding from the Bill and Melinda Gates Foundation to enable further development and demonstration of our ecosystem in two Least Developed Countries (LDCs), Tanzania and the Democratic Republic of the Congo (DRC), to provide high quality, real-time data on the diversity, distribution and abundance of mosquitoes. Combining these data with environmental variables from remote sensing satellite data, we will also demonstrate the potential of this innovative system to create dynamic heat maps of their distribution, providing information to i) health care practitioners to better target mosquito control methods, ii) health policymakers to enhance more effective intervention policies and, iii) the academic community, providing unprecedented levels of occurrence, behavioural and ecological mosquito data.

Specifically, the project aims to:

1. Conduct more extensive field trials to compare and evaluate our sensor with other vector monitoring methods such as CO_2 baited Centers for Diseases Control Light Traps (CDC-LT) and Mosquito Electrocuting grid Traps (MET).

2. Establish the feasibility of using HumBug to monitor mosquitoes in data-poor areas where it is difficult to conduct traditional surveys.

3. Integrate mobile phone-based airtime or cash payments into the MozzWear app to facilitate community involvement in HumBug mosquito surveys.

4. Develop a web-based platform with vector occurrence and abundance data. The project will run for 36 months and aims to sufficiently demonstrate the proof of concept to ensure future academic and commercial involvement, allowing HumBug to be a globally accessible tool used in the detection of primary mosquito vector species.

8.2.2 Label super-resolution

With the success of crowdsourcing, we hope to further focus our attention on the problem of label super-resolution as we see this as a current challenge with our database. To the best of our knowledge, label super-resolution research is still in the process of early growth, with the closest problem we find to be the formulation of *DCASE 2019* (Turpault et al., 2019, Task 4). The task is to learn onset and offset times for sound events given weakly labelled data, unlabelled data, as well as synthetic audio (to mimic finely labelled data). The leading submissions found little benefit (and even a detriment) in utilising the synthetic audio, but instead obtained their predictive power through carefully incorporating the weak labels (Lin et al., 2019). We suspect this may be due to the challenge in previous years being structured to only provide weak labels (Xu et al., 2017b), or that the difficulty of obtaining a comprehensive mixture of label resolutions creates a further challenge.

Several solutions in the domain of audio tagging rely on attention neural networks, which aim to learn the salient parts of the input space automatically (Xu et al., 2017a).

Furthermore, thanks to the Bayesian approach taken in Chapter 7, our weak label framework is able to produce well-calibrated probabilities as outputs. We can consider a combination of this framework with our findings from Chapter 6, to re-cast learning from probabilistic inputs as learning with a specific utility per data point in the loss-calibrated framework. Thus we can use uncertain labels to their full potential, and avoid having to threshold and discard data that is deemed unreliable based on user heuristics.

8.2.3 HumBug smartphone app

We aim to improve on the smartphone app by making use of the benefits of wavelet transforms that we demonstrated in Chapter 4. A relatively unexplored area in the mid-level representation space is the utilisation of discrete wavelet transforms, which can be implemented efficiently on limited resources (Shukla and Tiwari, 2013). If we are able to maintain predictive performance as strong as that of the continuous wavelet transform, given the mobile implementation of neural networks available, we would seek to port a wavelet-CNN model to be used on a smartphone. We justify the transition to a CNN by demonstrating consistent performance improvements in moving to a CNN from an SVM in Chapters 4 and 5. A further source of improvement, and advocate for the use of a CNN, is the ability to adapt large-scale pre-trained models to categorise different sources of noise for more accurately detecting mosquitoes. In current systems, we expect not to be robust to noise which occurs in similar bandwidth that occupies the same regions of the spectrogram, especially if the temporal behaviour is only taken into account over a limited timeframe.

8.2.4 Feature learning

Further work may also look to other methods of performing feature learning from audio. This area of research is relevant to any learning problem in the space of audio signals. We observe a paradigm shift in learning towards fully end-to-end trained models, where the goal is to learn fully the feature transformations from raw data. In recent applications, audio learning methods that learn without a spectral representation have shown success in the aspect of generative modelling (van den Oord et al., 2016), and we believe that the ideas of dilated convolutions have strong applicability to the problem of detection. An early application thereof is given in Zhang et al. (2017).

Bibliography

S. Adavanne and T. Virtanen. Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network. *arXiv preprint arXiv:1710.02998*, 2017. (Cited on page 119.)

O. C. Ai, M. Hariharan, S. Yaacob, and L. S. Chee. Classification of speech dysfluencies with MFCC and LPCC features. *Expert Systems with Applications*, 39(2):2157–2165, 2012. (Cited on page 51.)

M. Akay. *Time Frequency and Wavelets in Biomedical Signal Processing*. IEEE Press Series in Biomedical Engineering, 1998. (Cited on page 51.)

A. K. Alexandridis and A. D. Zapranis. Wavelet neural networks: a practical guide. *Neural Networks*, 42:1–27, 2013. (Cited on page 53.)

L. Alphey, M. Benedict, R. Bellini, G. G. Clark, D. A. Dame, M. W. Service, and S. L. Dobson. Sterile-insect methods for control of mosquito-borne diseases: an analysis. *Vector-Borne and Zoonotic Diseases*, 10(3):295–311, 2010. (Cited on page 47.)

J. Amores. Multiple instance classification: review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105, 2013. (Cited on page 119.)

Audacity. Audacity(R): Free audio editor and recorder [computer application], 2018. URL https://audacityteam.org/. Version 2.2.2 accessed: 2018-05-03. (Cited on page 59.)

S. Balakrishnama and A. Ganapathiraju. Linear discriminant analysis – a brief tutorial. *Institute for Signal and Information Processing*, 18:1–8, 1998. (Cited on page 124.)

A. Bansal and A. Kumar. Heisenberg uncertainty inequality for Gabor transform. *arXiv preprint arXiv:1507.00446*, 2015. (Cited on page 54.)

D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley. Acoustic scene classification: classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3):16–34, 2015. (Cited on pages 73 and 78.)

R. E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961. (Cited on page 12.)

P. Belton. Sounds of insects in flight. In *Insect Flight*, pages 60–70. Springer, 1986. (Cited on page 36.)

J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Science & Business Media, 1985. (Cited on page 14.)

S. Bhatt, D. J. Weiss, E. Cameron, D. Bisanzio, B. Mappin, U. Dalrymple, K. E. Battle, C. L. Moyes, A. Henry, P. A. Eckhoff, E. A. Wenger, O. Briet, M. A. Penny, T. A. Smith, A. Bennett, J. Yukich, T. P. Eisele, J. T. Griffin, C. A. Fergus, M. Lynch, F. Lindgren, J. M. Cohen, C. L. J. Murray, D. L. Smith, S. I. Hay, R. E. Cibulskis, and P. W. Gething. The effect of malaria control on Plasmodium falciparum in Africa between 2000 and 2015. *Nature*, 526(7572): 207–211, 10 2015. (Cited on page 47.)

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. (Cited on pages 10, 14, 16, 20, 21, 102, and 103.)

C. M. Bishop et al. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. (Cited on pages 22 and 129.)

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. (Cited on page 123.)

Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010. (Cited on page 28.)

G. E. Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976. (Cited on page 13.)

G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015. (Cited on pages 7 and 8.)

E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015. (Cited on page 119.)

E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017. (Cited on page 119.)

R. Cao, A. Cuevas, and W. G. Manteiga. A comparative study of several smoothing methods in density estimation. *Computational Statistics & Data Analysis*, 17(2): 153–176, 1994. (Cited on page 123.)

M. Cartwright, G. Dove, A. E. Méndez Méndez, J. P. Bello, and O. Nov. Crowdsourcing multi-label audio annotation tasks with citizen scientists. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 292. ACM, 2019. (Cited on page 125.)

R. Caruana, A. Munson, and A. Niculescu-Mizil. Getting the most out of ensemble selection. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 828–833. IEEE, 2006. (Cited on page 87.)

Y. Chen, B. Yang, and J. Dong. Time-series prediction using a local linear wavelet neural network. *Neurocomputing*, 69(4-6):449–465, 2006. (Cited on page 53.)

Y. Chen, A. Why, G. Batista, A. Mafra-Neto, and E. Keogh. Flying insect classification with inexpensive sensors. *Journal of Insect Behavior*, 27(5):657–677, 2014. (Cited on pages 36, 48, and 50.)

E. Chesmore and E. Ohya. Automated identification of field-recorded songs of four British grasshoppers using bioacoustic signal recognition. *Bulletin of Entomological Research*, 94(04):319–330, 2004. (Cited on pages 49 and 50.)

H.-G. Chew, R. E. Bogner, and C.-C. Lim. Dual ν-support vector machine with error rate and training size biasing. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1269–1272, Salt Lake City, USA, May 2001. (Cited on page 86.)

F. Chollet et al. Keras, 2015. URL `https://keras.io`. Accessed: 2018-06-07. (Cited on page 62.)

C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, 2000. (Cited on page 53.)

M. Claesen, F. De Smet, J. A. Suykens, and B. De Moor. Fast prediction with SVM models containing RBF kernels. *arXiv preprint arXiv:1403.0736*, 2014. (Cited on page 62.)

A. N. Clements. *The Biology of Mosquitoes, Volume 2: Sensory Reception and Behaviour*. CABI Publishing, 1999. (Cited on pages 12, 35, 37, 41, and 59.)

P. J. Clemins, M. T. Johnson, K. M. Leong, and A. Savage. Automatic classification and speaker identification of African elephant (Loxodonta africana) vocalizations. *The Journal of the Acoustical Society of America*, 117(2):956–963, 2005. (Cited on page 49.)

A. D. Cobb. *The Practicalities of Scaling Bayesian Neural Networks to Real-World Applications*. PhD thesis, University of Oxford, 2020. (Cited on pages 102, 104, and 105.)

A. D. Cobb, S. J. Roberts, and Y. Gal. Loss-calibrated approximate inference in Bayesian neural networks. *arXiv preprint arXiv:1805.03901*, 2018. (Cited on pages 105, 106, 107, and 109.)

A. D. Cobb, A. G. Baydin, I. Kiskin, A. Markham, and S. J. Roberts. Semi-separable Hamiltonian Monte Carlo for inference in Bayesian neural networks. In *Advances in Neural Information Processing Systems Workshop on Bayesian Deep Learning*, 2019. (Cited on page ix.)

M. A. Cody. The fast wavelet transform: beyond Fourier transforms. *Dr. Dobb's Journal*, 17(4):16–28, 1992. (Cited on page 63.)

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, Sept. 1995. (Cited on page 81.)

I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005, 1990. (Cited on page 53.)

I. Daubechies, J. Lu, and H.-T. Wu. Synchrosqueezed wavelet transforms: an empirical mode decomposition-like tool. *Applied and Computational Harmonic Analysis*, 30(2):243–261, 2011. (Cited on page 55.)

M. A. Davenport, R. G. Baraniuk, and C. D. Scott. Tuning support vector machines for minimax and Neyman-Pearson classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1888–1898, 2010. (Cited on page 86.)

J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240, 2006. (Cited on page 13.)

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009. (Cited on page 125.)

L. Deng, D. Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014. (Cited on page 91.)

S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, 2014. (Cited on pages 28, 33, 58, and 109.)

D. E. Difallah, G. Demartini, and P. Cudré-Mauroux. Mechanical cheat: spamming schemes and adversarial techniques on crowdsourcing platforms. In *CrowdSearch*, pages 26–30, 2012. (Cited on page 12.)

P. Domingos. A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238, 2000. (Cited on page 18.)

D. L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995. (Cited on page 53.)

P. Du, W. A. Kibbe, and S. M. Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22(17):2059–2065, 2006. doi: 10.1093/bioinformatics/btl355. (Cited on page 55.)

M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani. Exploiting spectro-temporal locality in deep learning based acoustic event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):26, 2015. (Cited on page 33.)

E. Fanioudakis, M. Geismar, and I. Potamitis. Mosquito wingbeat analysis and classification using deep learning. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2410–2414, 2018. (Cited on page 92.)

P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento. Reliable detection of audio events in highly noisy environments. *Pattern Recognition Letters*, 65: 22–28, 2015. (Cited on page 118.)

J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer Series in Statistics New York, 2001. (Cited on page 18.)

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016. (Cited on page 106.)

J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: an ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017. (Cited on pages 96 and 111.)

T. Giannakopoulos. pyAudioAnalysis: An open-source Python library for audio signal analysis. *PloS one*, 10(12):e0144610, 2015. (Cited on page 58.)

D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley. Detection and classification of acoustic scenes and events: an IEEE AASP challenge. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013. (Cited on page 119.)

H. Goëau, H. Glotin, W.-P. Vellinga, R. Planqué, A. Rauber, and A. Joly. LifeCLEF bird identification task 2015. In *CLEF2015*, 2015. (Cited on pages 52 and 59.)

S. Goetze, J. Schroder, S. Gerlach, D. Hollosi, J.-E. Appell, and F. Wallhoff. Acoustic monitoring and localization for social care. *Journal of Computing Science and Engineering*, 6(1):40–50, 2012. (Cited on page 118.)

B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou. Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm. *ACM computing surveys (CSUR)*, 48(1), 2015. (Cited on page 72.)

I. Guyon and A. Elisseeff. An introduction to feature extraction. In *Feature Extraction*, pages 1–25. Springer, 2006. (Cited on page 126.)

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, 2002. (Cited on page 61.)

G. Gwardys and D. Grzywczak. Deep image features in music information retrieval. *International Journal of Electronics and Telecommunications*, 60(4):321–326, 2014. (Cited on page 53.)

B. Hanczar and E. R. Dougherty. Classification with reject option in gene expression data. *Bioinformatics*, 24(17):1889–1895, 2008. (Cited on page 129.)

T. Hayashi, S. Watanabe, T. Toda, T. Hori, J. Le Roux, and K. Takeda. BLSTM-HMM hybrid system combined with sound activity detection network for polyphonic sound event detection. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 766–770. IEEE, 2017. (Cited on page 121.)

H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. (Cited on page 128.)

S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135. IEEE, 2017. (Cited on pages 28, 91, 96, and 111.)

D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied Logistic Regression*, volume 398. John Wiley & Sons, 2013. (Cited on page 10.)

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3): 651–674, 2006. (Cited on page 18.)

N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011. (Cited on page 125.)

T.-J. Hsieh, H.-F. Hsiao, and W.-C. Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11(2):2510–2525, 2011. (Cited on page 53.)

C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. *Internal report, Department of Computer Science, National Taiwan University*, 2003. (Cited on page 126.)

E. J. Humphrey, J. P. Bello, and Y. LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013. (Cited on pages 33, 51, 52, and 65.)

A. Ivanov and G. Riccardi. Kolmogorov-Smirnov test for feature selection in emotion recognition from speech. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5125–5128. IEEE, 2012. (Cited on page 121.)

M. John. *A Dictionary of Epidemiology*. Oxford University Press, 2001. (Cited on page 47.)

A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W.-P. Vellinga, J. Champ, R. Planqué, S. Palazzo, and H. Müller. LifeCLEF 2016: multimedia life species identification challenges. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 286–310. Springer, 2016. (Cited on pages 48 and 52.)

M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999. (Cited on page 105.)

W. H. O. Jr. and M. C. Kahn. The sounds of disease-carrying mosquitoes. *The Journal of the Acoustical Society of America*, 21:462 – 463, 1949. (Cited on page 72.)

B.-H. Juang and L. R. Rabiner. Automatic speech recognition – a brief history of the technology development. *Georgia Institute of Technology and the University of California*, 1:67, 2005. (Cited on page 109.)

S. Kadambe and P. Srinivasan. Adaptive wavelets for signal classification and compression. *AEU - International Journal of Electronics and Communications*, 60(1):45–55, 2006. (Cited on pages 26 and 53.)

S. M. Kamruzzaman, A. N. M. R. Karim, M. S. Islam, and M. E. Haque. Speaker identification using MFCC-domain support vector machine. *International Journal of Electrical and Power Engineering*, 1:274–278, 2007. (Cited on page 96.)

B. Karlik and A. V. Olgac. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011. (Cited on page 26.)

D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. *arXiv:1412.6980*, 2014. (Cited on pages 26, 81, and 90.)

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. (Cited on pages 85 and 86.)

I. Kiskin, B. P. Orozco, T. Windebank, D. Zilli, M. Sinka, K. Willis, and S. Roberts. Mosquito detection with neural networks: the buzz of deep learning. *arXiv preprint arXiv:1705.05180*, 2017. (Cited on pages viii, 47, and 89.)

I. Kiskin, D. Zilli, Y. Li, M. Sinka, K. Willis, and S. Roberts. Bioacoustic detection with wavelet-conditioned convolutional neural networks. *Neural Computing and Applications: Special Issue on Deep Learning for Music and Audio*, Aug 2018. ISSN 1433-3058. doi: 10.1007/s00521-018-3626-7. URL https://doi.org/10.1007/s00521-018-3626-7. (Cited on pages viii, 4, 42, 47, and 123.)

I. Kiskin, U. Meepegama, and S. Roberts. Super-resolution of time-series labels for bootstrapped event detection. *Time-series Workshop at the International Conference on Machine Learning*, 2019. (Cited on pages ix, 4, and 44.)

I. Kiskin, L. Wang, A. Cobb, et al. Humbug Zooniverse: a crowd-sourced acoustic mosquito dataset. *International Conference on Acoustics, Speech, and Signal Processing 2020, NeurIPS Machine Learning for the Developing World Workshop 2019*, 2019, 2020. (Cited on pages ix, 4, 44, and 91.)

I. Kiskin, A. Cobb, L. Wang, and S. Roberts. Loss-calibrated Bayesian neural networks for noisy acoustic mosquito detection. *Submitted*, 2020. (Cited on pages 4 and 44.)

Q. Kong, Y. Xu, I. Sobieraj, W. Wang, and M. D. Plumbley. Sound event detection and time–frequency segmentation from weakly labelled data. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 27(4):777–787, 2019. (Cited on page 121.)

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. (Cited on pages 28, 52, 57, and 117.)

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. (Cited on page 105.)

N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Community Magazine*, 48(9):140–150, Sept 2010. (Cited on page 72.)

N. D. Lane, P. Georgiev, and L. Qendro. DeepEar: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the ACM International Joint Conference on Pervasive Ubiquitous Computing (UbiComp)*, pages 283–294, 2015. (Cited on page 72.)

M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014. (Cited on page 91.)

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. (Cited on page 117.)

H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1096–1104, 2009. (Cited on pages 51 and 52.)

C. Lengeler. Insecticide-treated nets for malaria control: real gains. *Bulletin of the World Health Organization*, 82(2):84–84, 2004. (Cited on page 47.)

Z. Leqing and Z. Zhen. Insect sound recognition based on SBC and HMM. In *IEEE International Conference on Intelligent Computation Technology and Automation (ICICTA)*, volume 2, pages 544–548, 2010. (Cited on page 50.)

D. Li, I. K. Sethi, N. Dimitrova, and T. McGee. Classification of general audio data for content-based retrieval. *Pattern Recognition Letters*, 22(5):533–544, 2001. (Cited on page 52.)

Y. Li, I. Kiskin, D. Zilli, M. Sinka, H. Chan, K. Willis, and S. Roberts. Cost-sensitive detection with variational autoencoders for environmental acoustic sensing. *NeurIPS Workshop on Machine Learning for Audio Signal Processing*, 2017a. (Cited on pages ix, 43, and 107.)

Y. Li, D. Zilli, H. Chan, I. Kiskin, M. Sinka, S. Roberts, and K. Willis. Mosquito detection with low-cost smartphones: data acquisition for malaria research. *NeurIPS Workshop on Machine Learning for the Developing World*, 2017b. (Cited on pages viii, 4, 40, 41, 42, 70, 78, 80, 81, 88, 92, 107, and 108.)

Y. Li, I. Kiskin, M. Sinka, D. Zilli, H. Chan, E. Herreros-Moya, T. Chareonviriyaphap, R. Tisgratog, K. Willis, and S. Roberts. Fast mosquito acoustic detection with field cup recordings: an initial investigation. *Detection and Classification of Acoustic Scenes and Events*, 2018. (Cited on pages ix, 43, and 70.)

L. Lin, X. Wang, H. Liu, and Y. Qian. Guided learning convolution system for DCASE 2019 Task 4. *arXiv preprint arXiv:1909.06178*, 2019. (Cited on page 136.)

S. Lin and D. Zilli. MozzWear v1, 2015. URL https://github.com/HumBug-Mosquito/MozzWear. Accessed: 2020-04-15. (Cited on page 40.)

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. (Cited on page 125.)

B. Liu, I. Kiskin, and S. Roberts. An overview of Gaussian process regression for volatility forecasting. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 681–686, 2020. (Cited on page ix.)

D. J. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992. (Cited on page 102.)

A. Mesaros, T. Heittola, and T. Virtanen. TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132. IEEE, 2016. (Cited on page 119.)

A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen. DCASE 2017 challenge setup: tasks, datasets and baseline system. 2017. (Cited on page 118.)

J. Moffitt. Ogg vorbis—open, free audio—set your media free. *Linux Journal*, 2001 (81es):9, 2001. (Cited on page 29.)

A. Moore, J. R. Miller, B. E. Tabashnik, and S. H. Gage. Automated identification of flying insects by analysis of wingbeat frequencies. *Journal of Economic Entomology*, 79(6):1703–1706, 1986. (Cited on page 50.)

M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2015. (Cited on page 109.)

H. Mukundarajan, F. J. H. Hol, E. A. Castillo, C. Newby, and M. Prakash. Using mobile phones as acoustic sensors for high-throughput mosquito surveillance. *eLife*, 6:e27854, Oct 2017. ISSN 2050-084X. doi: 10.7554/eLife.27854. URL https://doi.org/10.7554/eLife.27854. (Cited on pages 48 and 92.)

M. Müller. *Information Retrieval for Music and Motion*, volume 2. Springer, 2007. (Cited on page 33.)

M. Muller, D. P. Ellis, A. Klapuri, and G. Richard. Signal processing for music analysis. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1088–1110, 2011. (Cited on page 29.)

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012. (Cited on pages 10 and 102.)

M. Naghshvar, T. Javidi, and K. Chaudhuri. Noisy Bayesian active learning. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1626–1633. IEEE, 2012. (Cited on page 125.)

V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010. (Cited on page 25.)

R. M. Neal. Bayesian learning for neural networks. *Lecture Notes in Statistics*, volume 118, 2012. (Cited on page 102.)

M. A. Nielsen. *Neural Networks and Deep Learning*, volume 2018. Determination press San Francisco, CA, USA:, 2015. (Cited on pages 24 and 26.)

J. N. Oswald, J. Barlow, and T. F. Norris. Acoustic identification of nine delphinid species in the eastern tropical Pacific Ocean. *Marine Mammal Science*, 19(1): 20–037, 2003. ISSN 1748-7692. doi: 10.1111/j.1748-7692.2003.tb01090.x. (Cited on page 50.)

D. O'Shaughnessy. Automatic speech recognition: history, methods and challenges. *Pattern Recognition*, 41(10):2965–2979, 2008. (Cited on page 58.)

A. Páerez, P. Larrañaga, and I. Inza. Information theory and classification error in probabilistic classifiers. In *International Conference on Discovery Science*, pages 347–351. Springer, 2006. (Cited on page 123.)

M. Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005. (Cited on page 19.)

S. Parsons and G. Jones. Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks. *Journal of Experimental Biology*, 203(17):2641–2656, 2000. (Cited on page 49.)

E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962. (Cited on page 122.)

K. J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015. (Cited on pages 109 and 118.)

J. Pinhas, V. Soroker, A. Hetzoni, A. Mizrach, M. Teicher, and J. Goldberger. Automatic acoustic detection of the red palm weevil. *Computer and Electronics in Agriculture*, 63:131–139, 2008. URL http://www.sciencedirect.com/science/article/pii/S0168169908000628. (Cited on page 50.)

I. Potamitis. Classifying insects on the fly. *Ecological Informatics*, 21:40–49, 2014. (Cited on page 51.)

I. Potamitis. Deep learning for detection of bird vocalisations. *arXiv preprint arXiv:1609.08408*, 2016. (Cited on page 53.)

I. Potamitis, T. Ganchev, and N. Fakotakis. Automatic acoustic identification of crickets and cicadas. In *Nineth International Symposium on Signal Processing and Its Applications. ISSPA 2007*, 2007. URL http://ieeexplore.ieee.org/document/4555462/. (Cited on page 50.)

J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987. (Cited on page 18.)

D. R. Raman, R. R. Gerhardt, and J. B. Wilkerson. Detecting insect flight sounds in the field: implications for acoustical counting of mosquitoes. *Agricultural and Biosystems Engineering Publications and Papers*, 2007. (Cited on pages 37 and 72.)

A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519. IEEE, 2014. (Cited on page 28.)

I. Rish et al. An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, volume 3, pages 41–46, 2001. (Cited on page 124.)

S. Roberts. Spectral estimation. *B14 Signal Processing Lecture Notes*, 2010. (Cited on page 40.)

D. Rolnick, A. Veit, S. Belongie, and N. Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017. (Cited on page 117.)

T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A.-r. Mohamed, G. Dahl, and B. Ramabhadran. Deep convolutional neural networks for large-scale speech tasks. *Neural Networks*, 64:39–48, 2015. (Cited on pages 53 and 109.)

J. Salamon and J. P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24 (3):279–283, 2017. (Cited on page 109.)

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computing and Applications*, 12(5):1207–1245, May 2000. ISSN 0899-7667. (Cited on page 86.)

C. Scott. Performance measures for Neyman-Pearson classification. *IEEE Transactions on Information Theory*, 53:2852–2863, August 2007. (Cited on pages 84 and 87.)

D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015. (Cited on page 122.)

A. Sevilla, L. Bessonne, and H. Glotin. Audio bird classification with Inception-v4 extended with time and time-frequency attention mechanisms. *Working Notes of CLEF*, 2017. (Cited on page 52.)

K. K. Shukla and A. K. Tiwari. *Efficient Algorithms for Discrete Wavelet Transform: With Applications to Denoising and Fuzzy Inference Systems*. Springer Science & Business Media, 2013. (Cited on page 137.)

D. F. Silva, V. M. De Souza, G. E. Batista, E. Keogh, and D. P. Ellis. Applying machine learning and audio analysis techniques to insect recognition in intelligent traps. In *IEEE 12th International Conference on Machine Learning and Applications (ICMLA)*, volume 1, pages 99–104, 2013. (Cited on page 57.)

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*, volume 26. CRC press, 1986. (Cited on page 123.)

P. Y. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, Washington, DC, USA, Aug. 2003. (Cited on page 81.)

R. Simpson, K. Page, and D. De Roure. Zooniverse: observing the world's largest citizen science platform. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 1049–1054, 2014. doi: 10.1145/2567948.2579215. (Cited on pages 40, 92, and 95.)

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. (Cited on pages 57 and 123.)

D. L. Streiner and J. Cairney. What's under the ROC? An introduction to receiver operating characteristics curves. *The Canadian Journal of Psychiatry*, 52(2): 121–128, 2007. (Cited on page 87.)

C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852, 2017. (Cited on page 1.)

S.-Y. Tseng, J. Li, Y. Wang, J. Szurley, F. Metze, and S. Das. Multiple instance deep learning for weakly supervised small-footprint audio event detection. *arXiv preprint arXiv:1712.09673*, 2017. (Cited on page 119.)

B. A. Turlach. Bandwidth selection in kernel density estimation: a review. In *CORE and Institut de Statistique*. Citeseer, 1993. (Cited on page 123.)

N. Turpault, R. Serizel, J. Salamon, and A. P. Shah. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. *Workshop on Detection and Classification of Acoustic Scenes and Events*, 2019. (Cited on page 136.)

C. Valens. A Really Friendly Guide to Wavelets, 1999. URL https://www.cs.unm.edu/~williams/cs530/arfgtw.pdf. Accessed: 2020-04-20. (Cited on page 55.)

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: a generative model for raw audio. *CoRR abs/1609.03499*, 2016. (Cited on pages 28 and 138.)

L. R. Varshney and J. Z. Sun. Why do we perceive logarithmically? *Significance*, 10(1):28–31, 2013. (Cited on page 32.)

F. B. Vialatte, J. Solé-Casals, J. Dauwels, M. Maurice, and A. Cichocki. Bump time-frequency toolbox: a toolbox for time-frequency oscillatory bursts extraction in electrophysiological signals. *BMC Neuroscience*, 10(1):46, 2009. (Cited on page 56.)

M. P. Wand and M. C. Jones. *Kernel Smoothing*. CRC Press, 1994. (Cited on page 123.)

B. Warren, G. Gibson, and I. J. Russell. Sex recognition through midflight mating duets in culex mosquitoes is mediated by acoustic distortion. *Current Biology*, 19 (6):485–491, 2009. (Cited on page 35.)

World Health Organization et al. World Health Organization fact sheet 387, vector-borne diseases, 2014. URL http://www.who.int/kobe_centre/mediacentre/vbdfactsheet.pdf. Accessed: 2017-04-21. (Cited on page 47.)

World Health Organization et al. World malaria report 2016. *Geneva: WHO*, 13 December, 2016. (Cited on page 47.)

World Health Organization et al. World malaria report 2019, 2019. (Cited on page 1.)

J. Xu, A. G. Schwing, and R. Urtasun. Tell me what you see and I will show you where it is. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3190–3197, 2014. (Cited on page 109.)

Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley. Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging. *arXiv preprint arXiv:1703.06052*, 2017a. (Cited on pages 119 and 137.)

Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley. Surrey-CVSSP system for DCASE 2017 challenge Task 4. *arXiv preprint arXiv:1709.00551*, 2017b. (Cited on page 136.)

Z. Yang, L. Shangguan, W. Gu, Z. Zhou, C. Wu, and Y. Liu. Sherlock: micro-environment sensing for smartphones. *IEEE Transaction on Parallel Distributed Systems*, 25(12):3295–3305, Dec. 2014. (Cited on page 72.)

X. Zhang, Y. Zou, and W. Shi. Dilated convolution neural network with LeakyReLU for environmental sound classification. In *2017 22nd International Conference on Digital Signal Processing (DSP)*. IEEE, 2017. (Cited on page 138.)

D. Zilli, O. Parson, G. V. Merrett, and A. Rogers. A hidden Markov model-based acoustic cicada detector for crowdsourced smartphone biodiversity monitoring. *Journal of Artificial Intelligence Research*, 51:805–827, 2014. (Cited on pages 49 and 50.)

U. Zölzer. *DAFX: Digital Audio Effects*. John Wiley & Sons, 2011. (Cited on page 31.)