# Two Little Birds: A Study in Birdsong Binary Classification Problems

Candidate Number: YPDR0[1]

MSc Data Science & Machine Learning

Supervisor: Dr. Ifat Yasin

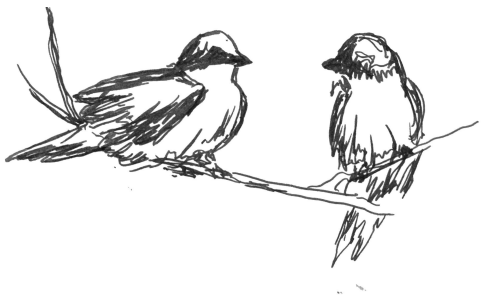
Submission date: 11 September 2023

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Data Science & Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged

**Abstract**

Automated or semi-automated birdsong classification plays a key role in a wide range of important fields, such as ecological surveying, climate modelling, and birdwatching for pleasure. With a recent surge in popularity of machine learning techniques and a wealth of of publicly available datasets, these problems are increasingly being tackled using state-of-the-art machine learning approaches with better and better results. This work will focus on investigating and evaluating popular approaches to birdsong classification problems, as well as experimenting with recent research into wider audio classification problems and evaluating them in a birdsong context.

All code used in this work can be found at `https://github.com/dbatten5/msc-thesis`.



# Contents

**1 Introduction** **2**

**2 Background and related work** **4**

2.1 Evaluation metrics . . . . . 4

2.2 Feature extraction . . . . . 6

2.2.1 Features inspired by the human auditory system . . . . . 8

2.2.2 Feature stacking . . . . . 11

2.3 Classification of samples . . . . . 13

2.3.1 SVMs . . . . . 13

2.3.2 Neural Networks . . . . . 15

2.4 Chapter summary . . . . . 24

**3 Methodology** **25**

3.1 Collecting the datasets . . . . . 25

3.2 Pre-processing audio recordings . . . . . 26

3.3 Segmenting audio recordings . . . . . 28

3.3.1 Energy based syllable segmentation . . . . . 28

3.3.2 Sequence sample generation . . . . . 32

3.4 Extracting features from samples . . . . . 32

3.4.1 Cepstral coefficients . . . . . 33

3.4.2 MRCG . . . . . 38

3.5 Training models . . . . . 39

3.5.1 SVMs . . . . . 39

3.5.2 Neural networks . . . . . 41

3.6 Evaluating models . . . . . 48

3.6.1 AUC . . . . . 49

3.6.2 Classification accuracy . . . . . 50
3.7 Chapter summary . . . . . 50

**4 Results and discussion** **51**
4.1 Hypothesis 1 . . . . . 51
4.1.1 MFCC . . . . . 51
4.1.2 GTCC . . . . . 52
4.2 Hypothesis 2 . . . . . 53
4.2.1 CNN . . . . . 53
4.2.2 RNN & CRNN . . . . . 55
4.3 Hypothesis 3 . . . . . 56
4.4 Comparison of models . . . . . 57
4.5 Discussion . . . . . 62
4.6 Further work and extensions of methodology . . . . . 63
4.7 Chapter summary . . . . . 65

**5 Conclusion** **67**

# Chapter 1

# Introduction

Birds play a crucial role in ecosystems around the world. They form an important link in the food chain, pollinate plants, and even plant trees [9]. The quantity and diversity of birds observed in an area can therefore often be regarded as a key indicator of the strength of its ecosystem.

Aside from being important ecological agents, birds also colour our lives with their sights, sounds and behaviours. The community of birdwatchers, known colloquially in the U.K. as 'birders' or 'twitchers', has grown considerably over the past few years. The Royal Society for the Protection of Birds (RSPB) reported a 70% increase in their website views over the first lockdown, with more than 50% of those views on pages looking at bird identification. The Bird Bird Garden Watch, an annual event in the U.K. that encourages people to note bird sightings in their residences, brought in 1 million participants in January 2021, more than double the previous year's tally. As such there is a growing commercial demand for accurate and easily accessible bird identification tools.

Birds are typically shy and protective creatures and tend to reside out of harm's way, obscured in shrubs, trees and nests. The majority of their communication is done through distinctive vocalisations, known as birdsong, that are usually unique to an individual species. Due to this, birds are typically identified through their birdsong rather than their visual sighting. Birdsong used for identification is usually recorded on microphones that may be running for a long time and/or are located in a place that isn't necessarily optimum to capture the vocalisation. The recordings may be corrupted from a wide range of sources, such as ambient noise, large changes in birdsong amplitude, long periods of silence, and vocalisations from other birds or animals. Recordings captured by microphones may also be several hours in duration so it would be impractical to have a human expert identify the

birds manually. There is therefore a need for robust birdsong identification tools that can handle these corruptions and that require minimal human intervention in order to classify unknown birdsong.

Modern solutions to this problem are increasingly relying on machine learning and as with most machine learning classification problems, the work boils down to two key components. The first is feature extraction, that is, given an input signal in the form of an amplitude varying over time, how can it be transformed to a vector or matrix representation that captures the key information of the signal. The second is classification, i.e. given this representation in the feature space, how can it be used to train a model that can then perform classification on unseen samples of birdsong. There are further steps that can be taken to improve this process, such as pre-processing the signal in order to remove or reduce the influence of background noise [54].

There has been a lot of research into audio classification problems both in birdsong and in other types of audio, such as speaker identification from human speech [35]. However, there remain novel methods that have been shown to have good results in wider audio classification problems that have yet to be tried out in birdsong identification problems. These methods include both feature extraction and classification. The broad aim of this thesis is to attempt to evaluate these methods. In more detail, the aims of this thesis can be summarised as follows:

1. Explore some of the hyperparameters available during the feature extraction process. The hypothesis is that using hyperparameter values more suited to birdsong classification problems will yield a higher classification accuracy.

2. Explore the performance of deep learning architectures such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) and compare the results with simpler statistical models. The hypothesis is that more complex and flexible architectures such as these will have superior performance when compared to simpler statistical models, such as support vector machines (SVM).

3. Explore the birdsong classification performance of feature representations shown to have promising results for non-birdsong related audio classification problems. The hypothesis is that feature representations shown to have good results in audio classification problems will have good results for birdsong identification.

# Chapter 2

# Background and related work

In this chapter, we first discuss the evaluation metrics typically used for classification problems. In Section 2.2 we explore classic and novel forms of feature extraction and feature combinations for audio problems. Section 2.3 categorises popular models used for audio classification problems, both old and new.

## 2.1 Evaluation metrics

For multiclass classification problems, e.g. identifying a sample as a particular bird species from a list of $n > 2$ species, a simple classification accuracy is often used as an evaluation metric [11, 56]. This is calculated as the percentage of correctly identified samples from a set of labelled samples previously unseen by the model. A high accuracy usually indicates a performant model, although it should be noted that this metric can be misleading if there is a large imbalance of classes in the test set. For example, consider a test set of 200 samples of class A and 10 each of classes B and C. For a model that always predicts class A, an accuracy of 91% is achieved, however, this model would likely be unsuitable since it will never predict classes B and C.

Acevedo et al. [1] used true positive rate (TPR) and false positive rate (FPR) to evaluate their models used to classify bird and amphibian calls, defined as

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}}, \quad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} \tag{2.1}$$

where TP and FP are the numbers of true positives and false positives respectively, and P and N are the number of real positive and real negative classes in the data. This evaluation

method is sometimes preferred when analysing long recordings which may include several different species to be classified. TPR indicates how well the model correctly identifies species present in the recording. FPR indicates how often the model erroneously identifies species absent in the recording. A high-performing model will have high values for TPR and low values for FPR.

Potamitis et al, [54] used an alternative form of evaluation presented in terms of precision ($P$) and recall ($R$) which are defined as

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad R = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

where FN is the number of false negatives. Informally, $P$ and $R$ can be thought of as

$$P = \frac{\text{relevant retrieved instances}}{\text{all } \textbf{retrieved} \text{ instances}}, \quad R = \frac{\text{relevant retrieved instances}}{\text{all } \textbf{relevant} \text{ instances}} \tag{2.3}$$

$P$ and $R$ are unhelpful metrics when reported in isolation. For example, it's possible to achieve perfect recall in a multiclass birdsong classification problem by simply assigning a sample to all possible classes. Note that in this example the precision will be close to 0. It's therefore desirable to achieve a high precision and a high recall, however, it's well known that there is a trade-off between $P$ and $R$. Improving one metric often leads to a decrease in the other metric. Due to this, the $F$-score is often reported along with the $P$ and $R$ metrics. The $F$-score is defined as

$$F = \frac{(1 + \beta^2)PR}{\beta^2 P + R} \tag{2.4}$$

for some $\beta \in \mathbb{R}$. The $\beta$ is chosen such that recall is considered $\beta$ times as important as precision. Usually, the $F_1$ score is presented, where $\beta = 1$.

For binary classification problems, i.e. classifying a sample as one of two classes (bird species), the Area Under Curve (AUC) metric is sometimes preferred [41]. It is calculated as the area under a receiver operating characteristic (ROC) curve, which plots TPR vs FPR at different classification thresholds, see Figure 2.1 for an example. The AUC ranges from 0 to 1, with a higher value indicating a better-performing model. The AUC is sometimes desirable because it is scale-invariant (it measures how well predictions are ranked, rather than their absolute values) and classification-threshold-invariant (it measures the quality of the model's predictions regardless of what classification threshold is chosen).

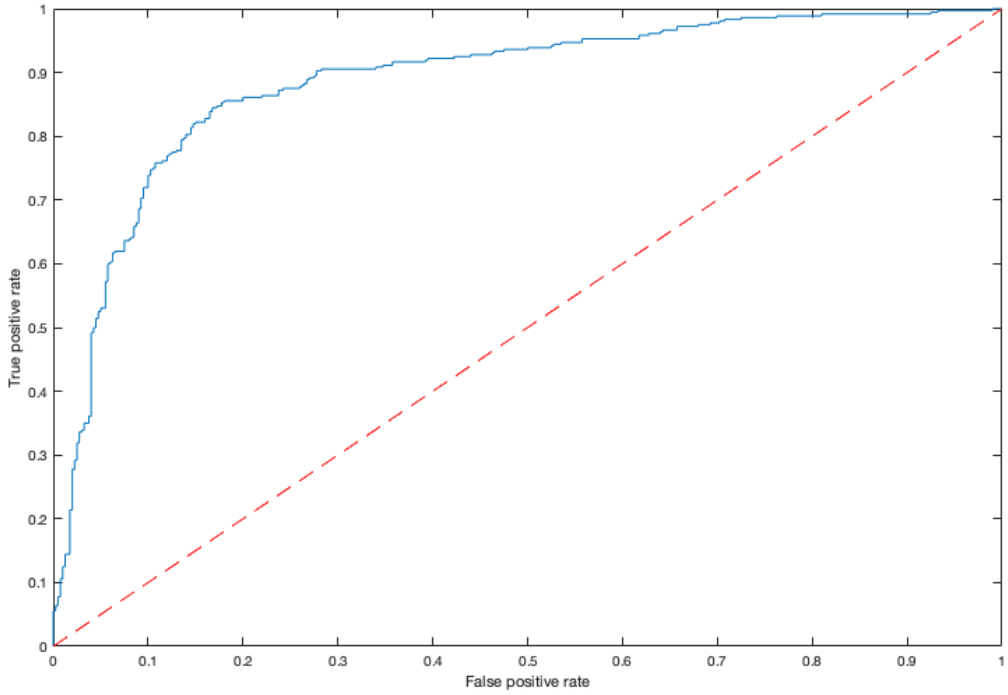



Figure 2.1: Example ROC curve for a reasonably performant model, shown in blue. The red dashed line indicates the baseline for binary classification problems: a model outputting random predictions. The AUC is calculated as the area under the blue curve.

## 2.2 Feature extraction

As mentioned in Chapter 1, birds use vocalisations as a way to communicate with others. Birds have evolved over many thousands of years to make this form of communication very efficient in that it can travel long distances and cut through local ambient noise to be received by other birds clearly. Some birds have even adapted their vocalisations so that they can be heard in local environments that have rapidly changed over the past decades, such as urban areas with increasing anthropogenic noise [44].

Bird vocalisations can be broadly divided into two main categories, calls and songs. Calls are typically short vocalisations that carry some specific function, such as warning others of the presence of a predator or calling others to flight [45]. Songs are usually longer, more acoustically complex, and occur more spontaneously. They are typically employed as breeding calls or territorial defence. While all birds produce calls, in many species of birds only the males utilise songs and often only during the breeding season. The song and call

of a Eurasian wren can be seen and contrasted in Figure 2.2.

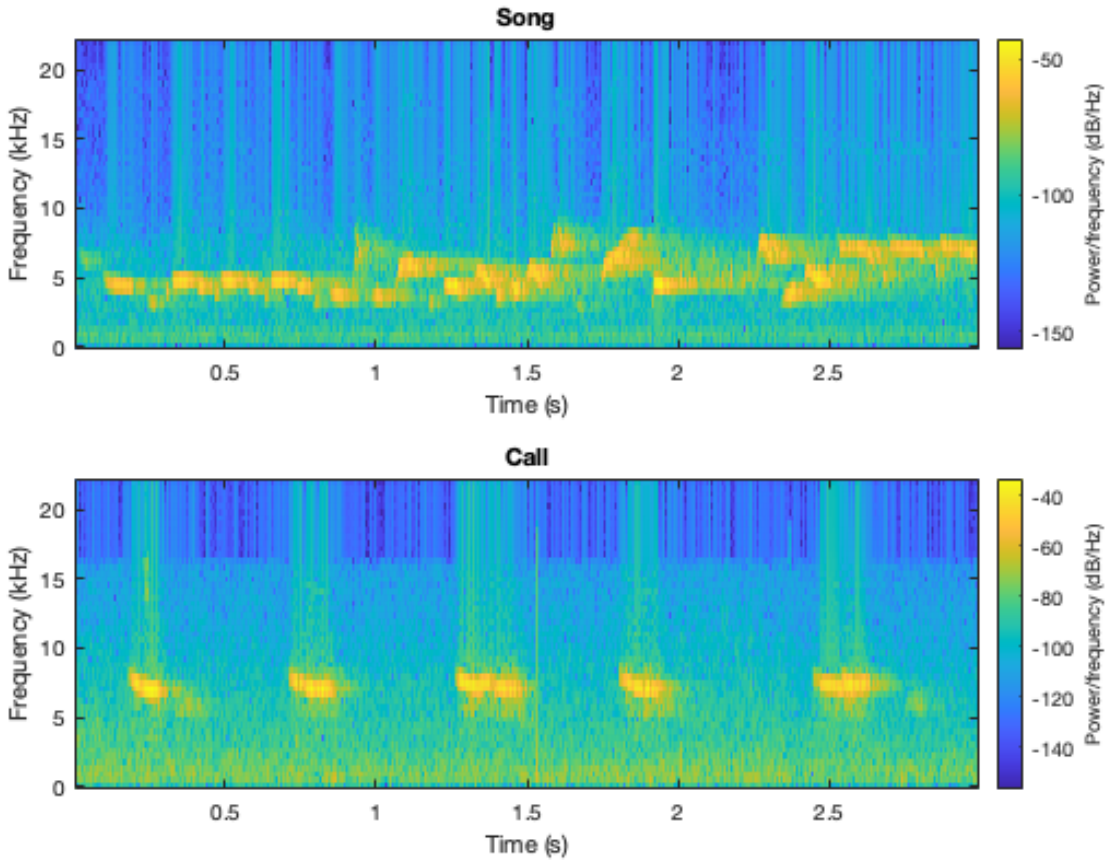



Figure 2.2: Spectrograms of a call and song of a Eurasian wren (*Troglodytes troglodytes*). As can be seen, the song is much more complex in terms of pitch and acoustic structure compared to the call.

Doupe et al. [20] showed that there are striking similarities between birdsong and human language. Similar to human language, birdsong can be thought of as consisting of hierarchical levels of phrases, syllables, and elements [10]. The levels for a common chaffinch can be seen in Figure 2.3. Raw recordings of birdsong often contain periods of silence that occur between phrases which are unlikely to provide useful information to train a machine learning model. Fagerlund [23] showed that a good level of accuracy can be achieved by training models on features extracted from segmented syllables which were used as training samples. An algorithm for robust syllable segmentation was proposed in [22] and is used in this thesis. The algorithm is described in more detail in Section 3.3.1.

Once the syllables have been segmented, there exists an abundance of options to turn

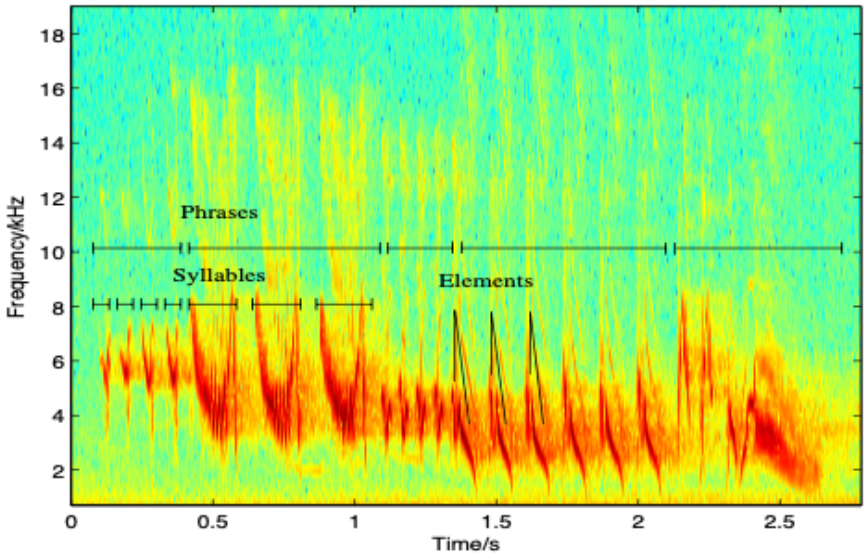



Figure 2.3: Spectrogram showing the hierarchical levels of the song of a common chaffinch. Figure taken from [22].

the raw syllable input signal into features that might be used as training samples. In the following sections, some of the more popular feature representations are described, along with some novel methods that have yet to be tried on birdsong. Note that the following list is certainly not exhaustive and emphasis has been placed on feature representations that are relevant to this thesis.

### 2.2.1 Features inspired by the human auditory system

Research has shown that birdsong and human language share many similarities in terms of the neural mechanisms employed to form the language/song and the impact of social contact in learning the language/song [20]. Given that the human auditory system has evolved over thousands of years to best process human speech, it seems reasonable to suggest that using features based on the human auditory system may be effective when it comes to birdsong.

At a high level, the human auditory system works by translating changes in air pressure originating from a source and reaching the outer ear of a listener into vibrations that travel along an internal organ known as a cochlea. The vibrations trigger electrical signals that move along auditory nerves to the brain, where they are interpreted as sounds. The

physiology of the cochlea means that specific parts of it are more sensitive to specific frequencies of vibrations, so different frequencies will lead to different electrical signals moving to the brain. This allows the brain to learn to differentiate between frequencies.

### Mel frequency cepstrum coefficients

The makeup of the human auditory system means that humans have a greater ability to differentiate pitch at lower frequencies than they do at higher frequencies. In other words, humans perceive pitch non-linearly. This has led to the development of a logarithmic frequency scale, known as the mel scale, such that equal distances on the scale have the same *perceptual* distance.

The mel frequency cepstrum coefficients (MFCC) are part of a family of cepstrum coefficients that capture information about the rate of change in different spectrum bands. MFCC differs from other cepstrum coefficients in that it uses the mel scale to transform the spectrum of an input signal, thus utilising the human auditory system in the calculation of its coefficients. This is done by passing a signal through a bank of filters that are spaced according to the mel scale.

The $i$-th mel cepstral coefficient is computed as [17]

$$\mathrm{MFCC}_i = \sum_{k=1}^{K} X_k \cos\left(\frac{i(k-0.5)\pi}{K}\right) \tag{2.5}$$

where $X_k$ is the logarithmic energy of the $k$-th mel-spectrum band, and $K$ is the total number of the mel-spectrum bands. Usually 8–13 MFCC coefficients are used as the feature vector representing one time frame of the signal. The $0^{th}$ coefficient is often excluded as it represents the average log energy of the signal and is unlikely to carry any relevant information to help with classification. MFCCs are often presented with their delta ($\Delta$) and double-delta ($\Delta\Delta$) values that capture the local temporal dynamics and temporal changes of the delta values respectively.

MFCCs are used as feature representations since they are simple to compute and have been shown to have good performance in a wide range of audio classification tasks, such as speaker identification [48] and emotion recognition [43]. MFCCs have also been shown to lead to good classification accuracy for birdsong identification problems [23, 56].

### Gammatone cepstrum coefficients

Gammatone cepstrum coefficients (GTCCs) are similar to MFCCs except their calculation uses gammatone filterbanks instead of mel scale filterbanks. Gammatone filterbanks are designed to simulate the motion of the membrane inside the cochlea, known as the basilar membrane, when it is exposed to vibrations transmitted by the outer and middle ear [53].

A gammatone filter with a centre frequency $f_c$ is defined as

$$g(t) = at^{n-1}e^{-2\pi bt}\cos(2\pi f_c + \psi) \tag{2.6}$$

where $t$ is the time in seconds, $\psi$ is the phase in radians (usually set to 0), $a \in \mathbb{R}$ controls the gain, $n$ is the filter's order and $b$ is the filter's bandwidth in Hz. To simulate the human auditory system, the centre frequencies are uniformly spaced on the equivalent rectangular band-width (ERB) scale.

Similar to MFCCs, GTCCs are typically used with their $\Delta$ and $\Delta\Delta$ values and have also been shown to have a good performance in non-speech audio classification problems [69].

For the Eurasian wren song shown in Figure 2.2, the MFCC and GTCC coefficients can be seen in Figure 2.4. Note that this visualisation does not include the $\Delta$ and $\Delta\Delta$ values and also includes the $0^{\text{th}}$ coefficient, denoted as $\log E$ in the figure.

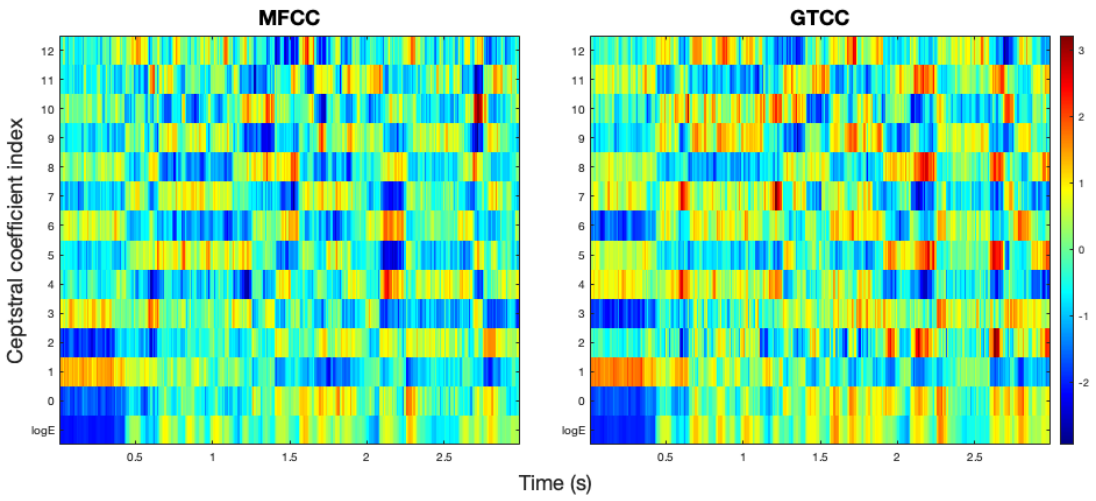



Figure 2.4: MFCC and GTCC coefficients for a sample of Eurasian wren birdsong.

#### Multi resolution cochleagram

Chen et al. [12] proposed a novel feature known as a multi-resolution cochleagram (MRCG) that is formed by combining four cochleagrams at different resolutions, designed to capture both local and contextual information.

A cochleagram is formed by passing an input signal through a gammatone filterbank, similar to the first steps of the formation of the GTCCs. Each response signal from the gammatone filterbank is then divided into frames of a certain length with a certain overlap or frame shift. The cochleagram is then generated by calculating the power of each frame at each channel.

A MRCG is formed by first taking one cochleagram with a frame length of 20ms and frame shift of 10ms, using a gammatone filterbank with 64 output channels. A log operation is applied to each time-frequency (T-F) unit of the output cochleagram, denoted CG1. CG2 is calculated in the same way, but with a frame length of 200ms and the same frame shift. CG3 is calculated by averaging CG1 across a square window of 11 frequency channels and 11 time frames centred at the given T-F unit using zero padding. CG4 is calculated in the same way as CG3, but with a $23 \times 23$ square window. CG1–4 are then stacked vertically to obtain the full MRCG feature, which will be a $256 \times p$ matrix, where $p$ is the number of time frames. The MRCG for the Eurasian wren song shown in Figure 2.2 can be seen in Figure 2.5.

The MRCG feature has been shown to outperform both MFCC and GTCC at separating human speech at various SNR levels with different types of background noise [12]. Abdullah et al. [7] showed that the MRCG outperforms the Auditory Image Model (AIM) when classifying noise. Wang et al. [72] showed that improved speech enhancement for hearing impaired listeners was achieved when the MRCG feature was added to the feature set.

One potential drawback when comparing the MRCG with MFCC and GTCC is the higher dimensionality of the MRCG feature. When including the $\Delta$ and $\Delta\Delta$ features, as is standard practice [7, 72], the MRCG feature vector for a single time frame has a dimensionality of 768. For MFCC or GTCC the equivalent dimensionality is 24–39, depending on the number of the coefficients used.

### 2.2.2 Feature stacking

It has been shown that an appropriate combination of features to be used as training inputs can lead to improved classification accuracy in audio related problems such as speech

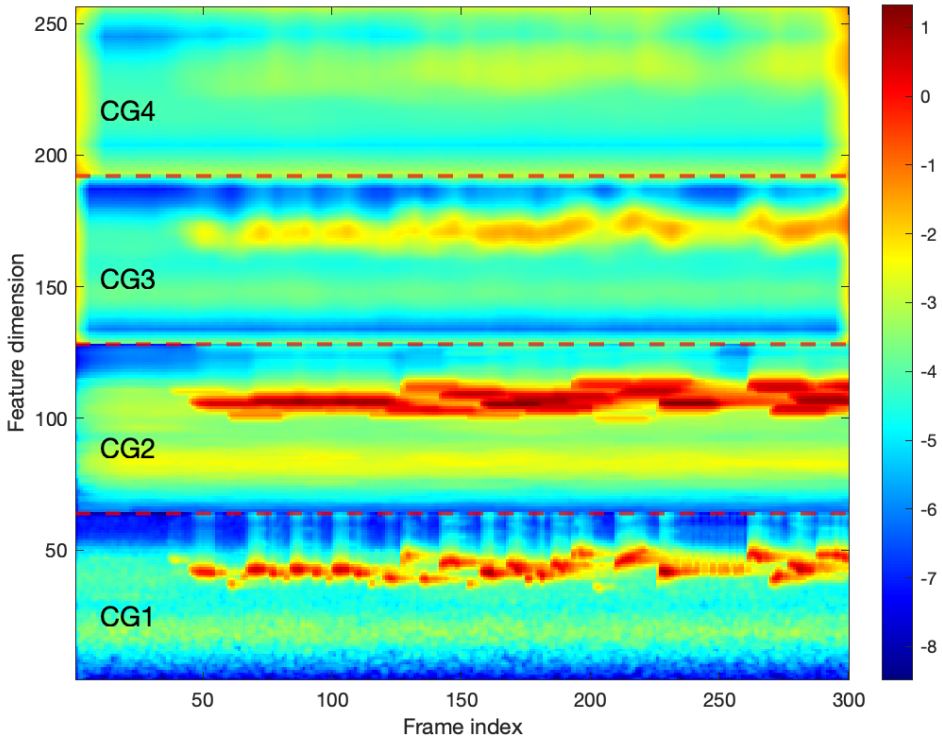



Figure 2.5: MRCG of sample of Eurasian Wren birdsong. The component cochleagrams are labelled CG1–4 and the red dashed lines indicate the boundaries between cochleagrams.

separation [71]. Ramashini et al. [56] showed that a combination of GTCC and MFCC yielded higher birdsong classification accuracy than MFCC alone. However, the combination did not improve on the accuracy of GTCC alone. Yan et al. [73] demonstrated that a combination of MFCC with two other feature types (Log-mel spectrogram and Chroma) yielded higher birdsong classification accuracy than combinations of two of the features alone. Fagerlund [23] showed that combining MFCC with spectral and temporal features, such as frequency range and zero crossing rate, can lead to higher birdsong classification accuracies.

A consequence of feature stacking is that feature vectors will have increased size. This can lead to problems such as longer training times and overfitting, especially when used in conjunction with deep learning (DL) architectures. As a result, dimensionality reduction techniques such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) [57] are often used to reduce the dimensionality of the feature vectors while still capturing enough information to be able to train a model effectively.

## 2.3 Classification of samples

Once a suitable feature representation has been selected and extracted, the remaining task is to pick an appropriate model and train the model using the set of training samples generated from the feature extraction.

There exist many different options in terms of models, and no one model is superior to the others. Each problem must be considered and a model which fits the problem at hand must be selected, or indeed a selection of suitable models tested and then evaluated for performance.

In the literature, birdsong classification has been tackled with a wide variety of models. Ramashini et al. [57] used the simple Nearest Centroid (NC) classifier to assign birdsong samples to classes. It was compared to a K Nearest Neighbours (KNN) classifier and shown to have superior accuracy. Lasseck [37] used decision trees with bagging to identify birds. Trifa et al. [68] used Hidden Markov Models (HMM) to identify species of birds based on their vocalisations. They considered each sample as a discrete-time dynamical system, where an unobserved state generates observed features, such as pitch and MFCC. A different HMM can be learned for each class and a new sample can be assigned to a class by calculating which HMM gives the highest likelihood of the observed features. Kwan et al. [36] used a Gaussian Mixture Model (GMM) to classify birds. In their experiments, a Gaussian was learned for each class of bird. A new sample was then classified according to whichever class best describes the new sample.

The following sections describe some models that are relevant to this thesis in more detail.

### 2.3.1 SVMs

SVMs are widely used in machine learning applications as a classification tool. They are well-established due to their high accuracy results [23] and relative simplicity to implement. Many implementations of SVMs also require little or no tuning of hyperparameters.

At its core, a SVM is a binary classifier that separates two classes by finding a hyperplane that maximises the margin from the nearest vectors in the feature space from both classes. Classifications are then made by computing on which side of the hyperplane a test feature vector lies.

**Binary classification**

Let $\mathbf{x}_i \in \mathbb{R}^m$ be a feature vector of dimensionality $m$. Let $y_i \in \{+1, -1\}$ be its class label. For linearly separable data, the separating hyperplane satisfies

$$y_i \left( \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \geq 1, \quad i = 1, \ldots, n \tag{2.7}$$

where $\mathbf{w} \in \mathbb{R}^m$ is a vector of weights and $b \in \mathbb{R}$ is a bias term. The margin between the hyperplane and the nearest feature vectors from each class is given by

$$\mathrm{d}(\mathbf{w}, b) = \min_{\mathbf{x}_i, y_i = 1} \frac{|\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|} + \min_{\mathbf{x}_j, y_i = -1} \frac{|\langle \mathbf{w} \cdot \mathbf{x}_j \rangle + b|}{\|\mathbf{w}\|} \tag{2.8}$$

$$= \frac{2}{\|\mathbf{w}\|}. \tag{2.9}$$

The optimal hyperplane can now be found by maximising (2.9) subject to (2.7). This can be solved using Lagrange multipliers.

Often with real world data, classes are not linearly separable. This can be remedied by projecting the data using a nonlinear mapping into a new feature space where the data are linearly separable. Equation (2.7) can then be rewritten as

$$y_i \left( \langle \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}_i) \rangle + b \right) \geq 1, \quad i = 1, \ldots, n \tag{2.10}$$

where $\mathbf{\Phi}$ is the nonlinear mapping. Instead of explicitly finding $\mathbf{\Phi}$, (2.10) can be re-written in dual form

$$y_i \left( \sum_{j=1}^{l} \alpha_j y_j \langle \mathbf{\Phi}(\mathbf{x}_j) \cdot \mathbf{\Phi}(\mathbf{x}_i) \rangle \right) + b \geq 1, \quad i = 1, \ldots, n \tag{2.11}$$

and replacing the inner product with a kernel function $K(\mathbf{x}_j, \mathbf{x}_i) = \langle \mathbf{\Phi}(\mathbf{x}_j) \cdot \mathbf{\Phi}(\mathbf{x}_i) \rangle$.

In practice, a hyperplane that separates the classes perfectly may suffer from poor generalisation ability. To improve this, nonnegative slack variables $\xi_i$ are introduced to (2.10) to allow for some missclassification of training samples to improve generalisation ability. The slack variables are introduced like so

$$y_i \left( \langle \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}_i) \rangle + b \right) \geq 1 - \xi_i, \quad i = 1, \ldots, n \tag{2.12}$$

The amount of regularization is controlled by the constant $C$ such that the maximisation

problem (2.9) becomes

$$\frac{2}{\|\mathbf{w}\|} - C\sum_{i=1}^{n}\xi_i \tag{2.13}$$

Therefore for large values of $C$ the classifier will behave like a hard-margin SVM and attempt to perfectly classify all training samples.

The last remaining question is around what determines a valid kernel function. A function in the input space is a kernel function if its kernel matrix $\mathbf{K} = [K(\mathbf{x}_j, \mathbf{x}_i)]_{i,j=1}^{n}$ is positive semidefinite. Typical kernel functions include

- linear: $K(\mathbf{x}_j, \mathbf{x}_i) = \langle \mathbf{x}_j, \mathbf{x}_i \rangle$
- polynomial: $(\langle \mathbf{x}_j, \mathbf{x}_i \rangle + c)^p$ for some $c \in \mathbb{R}$
- Gaussian or RBF: $\exp\left(-\gamma \|\mathbf{x}_j - \mathbf{x}_i\|^2\right)$ for some $\gamma \in \mathbb{R}, \gamma > 0$.

#### Multiclass classification

SVMs can easily be extended to work with multiclass classification using standard procedures. One such procedure which is optimal with respect to the Hamming Loss, which relates to the number of false positives or false negatives, is the one-versus-all technique. Here, multiple SVMs are trained, each one learning the hyperplane for samples belonging to one class versus samples belonging to all the other classes. A new sample can then be tested against all the models and predictions are made using the model that is most confident.

### 2.3.2 Neural Networks

Although neural networks (NN) have existed in some form since the 1950s with the inception of the perceptron algorithm [59], they have experienced a huge surge in popularity over the past decade or so. This has largely been due to remarkable progress being made thanks to deep neural networks in fields such as image classification [34] and language models [47].

Another reason for DL's increasing popularity in recent years is due to the availability of more performant hardware and software. The 'deep' aspect to deep learning refers to the fact that the software architecture depends on large amounts of parameters and needs massive amounts of training data to learn. In order to run training routines in a

reasonable time and store large amounts of data in memory, access to powerful hardware and/or machine parallelism tools can be an essential part of the process.

The progress of audio related problems has also been accelerated due to deep neural networks. Hinton et al. [28] showed improved performance in speech recognition tasks using feed-forward neural networks when compared to more classical approaches like HMMs and GMMs. Speech enhancement [3] and speech separation [21] problems leveraging DL paradigms have also been shown to outperform more classical statistical models.

The world of birdsong classification has also benefited from DL, but perhaps to a lesser extent than other audio related problems so far. Approaches have mainly focused on Convolutional Neural Networks (CNNs), applying convolutional layers to an image representation of an input signal, such as a (Mel-)spectrogram [6, 49]. Further birdsong classification solutions have approached the problem in a similar fashion, but utilised transfer learning to fine-tune an existing model [19, 38], such as ResNet [26] and ImageNet [18].

Disabato et al. [19] went a step further with their research in that they considered the computational and memory demands of their proposed bird detection DL network, ToucaNet. As mentioned earlier, DL applications can be extremely demanding in terms of computational resources which can make them unsuitable for running on devices with limited resources, such as smartphones and autonomous recording units (ARUs). Their research showed that a level of accuracy in line with the literature can be achieved but with lower computational complexity and memory demands.

Although the architecture of DL tools for birdsong classification may vary widely, the final layer is typically the same. This consists of a fully connected layer of $n$ units, where $n$ is the number of classes, with a softmax activation so that the output of the network can be considered as a probability distribution over the classes. The softmax function is defined as

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, \ldots, K \tag{2.14}$$

where $K$ is the number of classes. A classification of an unseen sample can then be performed by assigning the sample to the class with the highest corresponding probability.

### Feedforward Neural Network

A feedforward neural network (FNN) consists of a system of connected nodes, organised in layers, where information flows in one direction only — forward — from the input nodes, through the hidden nodes (if any), and to the output nodes. This differs fundamentally

from e.g. a RNN, where information can flow forward and backward. However, both FNNs and its derivatives and RNNs share many similarities, such as learning using backpropagation, and using nonlinear activation functions to model nonlinear data.

A FNN in perhaps its most basic form while still being able to solve non-trivial problems is known as a multilayer perceptron (MLP). An MLP consists of at least three layers of fully connected nodes with a nonlinear activation function, see Figure 2.6. Since their inception in late 21$^{st}$ century MLPs have had extensive research conducted into their capabilities and efficiency [30].

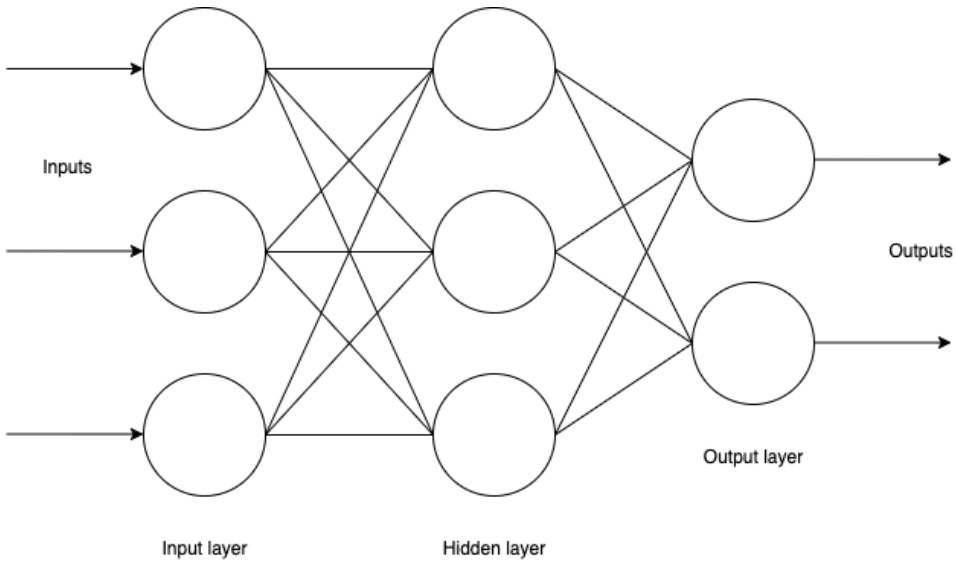



Figure 2.6: An example of a simple MLP. Data flows into the network from the input layer, through the hidden layer, and out the output layer.

MLPs have largely been superseded by more modern DL architectures such as CNN and RNN when it comes to audio problems such as birdsong classification, however there have been examples of more vanilla neural networks being used. McIlraith and Card [46] used a two layer perceptron network using backpropagation to minimize a mean squared error loss to classify samples from 7 birds. Murcia and Paniagua [50] used a simple FNN to classify birdsong from 35 different bird species, achieving 0.74 AUC and in doing so winning the International Conference on Machine Learning Bird Challenge 2013.

### Convolutional Neural Network

CNNs have seen a big surge in development and popularity over the last decade, largely thanks to huge strides being made in fields like machine vision and image classification

using architectures such as AlexNet [34]. Since image and audio share many conceptual similarities, it is reasonable to suggest that CNNs may enjoy similar success in the world of audio classification problems. This hypothesis is strengthened when one considers that many feature representations of audio signals can be displayed as images, such as spectrograms (Figure 2.2), MRCG outputs (Figure 2.5) and MFCC outputs (Figure 2.4). Indeed, there has been significant progress in recent years thanks to CNNs in fields such as audio event recognition [67] and automatic speech recognition [62].

A CNN is a feedforward neural network with at least one convolutional layer. A convolutional layer is so called because it performs a convolution on an input, usually an image, using a small matrix of weights, known as a filter or kernel. The output is considered as a matrix of activations. The activation is higher when the values in the corresponding area of the input are similar to those in the kernel, meaning that the activation output encodes the location of where the input matches certain features. Usually, several kernels are applied at each layer, and so the output takes the shape of a 3D matrix, or a volume. As with all neural nets, the output is passed through a nonlinear activation function in order to allow for the network to be able to learn from nonlinear data. The weights in the kernel are learned through backpropagation. Usually a max pooling layer is added to downsample the activation volume and reduce the number of weights, or parameters, needed by the network. An example of a simple CNN can be seen in Figure 2.7.

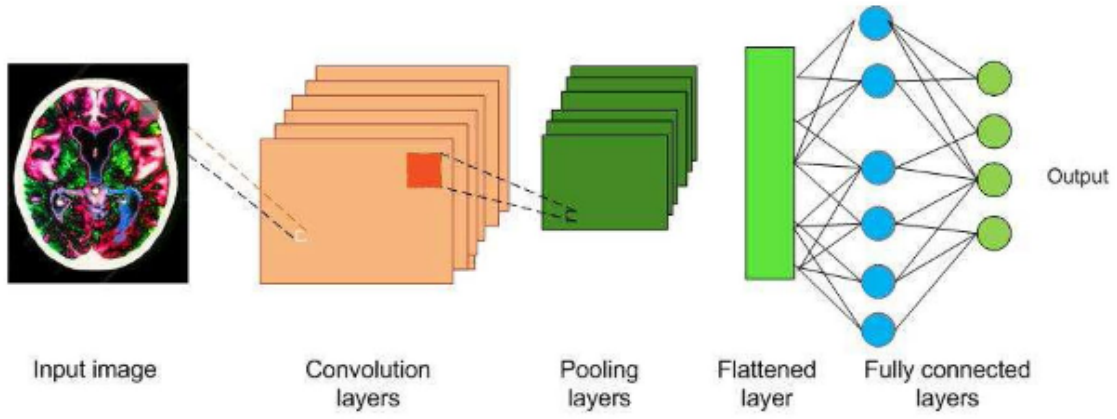



Figure 2.7: An example of a simple CNN architecture. This network operates on images of brain scans and outputs a class from one of 4 options. Figure taken from [61].

CNNs lend themselves well to image related problems since their structure allows them to parse an image as a composition of meaningful elements rather than a collection of

unrelated pixels [39]. This translates well to birdsong as e.g. a spectrogram of a sample of birdsong can conceptually be thought of as a composition of syllables, i.e. elements.

CNNs have been applied to birdsong classification challenges with great success. Kahl et al. [32] achieved a remarkable mean average precision of 0.605 for a dataset of 1500 different bird species using a CNN trained on spectrograms of 4 second samples. The spectrograms were pre-processed to reduce noise and augmented with a variety of augmentations, such as adding Gaussian noise and real noise samples from the original dataset. Ruff et al. [60] used CNNs to detect owl vocalisations in spectrograms from unprocessed field recordings, performing as well or better than human experts. Narasimhan et al. [51] used a CNN with encoder-decoder architecture based on Segnet [4] to simultaneously segment and classify birdsong spectrograms. This approach benefits from the fact that segmentation and classification are intuitively strongly correlated, since when comparing two different segmentations, the one that leads to a higher classification accuracy will likely be a better segmentation.

### Recurrent Neural Network

Since FNNs have a fixed size of input and output, they become unsuitable when either the input size is variable and the output is fixed (e.g. with AI image generation tools like Dall-E [58]), the input size is fixed and the output is variable (e.g. image captioning) or both the input size and output size are variable (e.g. language translation). RNNs solve this problem by allowing information in the network to flow in directed cycles. This is done by feeding the output from a previous step as input to the current step, meaning that the state of the hidden units depends on the previous state of the network. This gives the network the ability to learn long term dependencies thanks to an unbounded previous history. The architecture of a vanilla RNN cell is shown in Figure 2.8.

Similar to FNNs, classification RNNs learn by minimizing a loss function, usually the cross-entropy loss [63], by backpropagation. However, a slight variation on the algorithm is used to incorporate the sequential nature of the input data. The variation is known as backpropagation through time (BPTT). Here, to update the weights matrices, all weights are first treated as independent. The standard backpropagation algorithm is then run and all the corresponding gradients are averaged together. BPTT however suffers from a problem whereby gradients larger than 1 are multiplied together, which causes the gradient updates further back the network to exponentially increase, which may cause numerical instability and memory problems during training. This is known as the gradient exploding

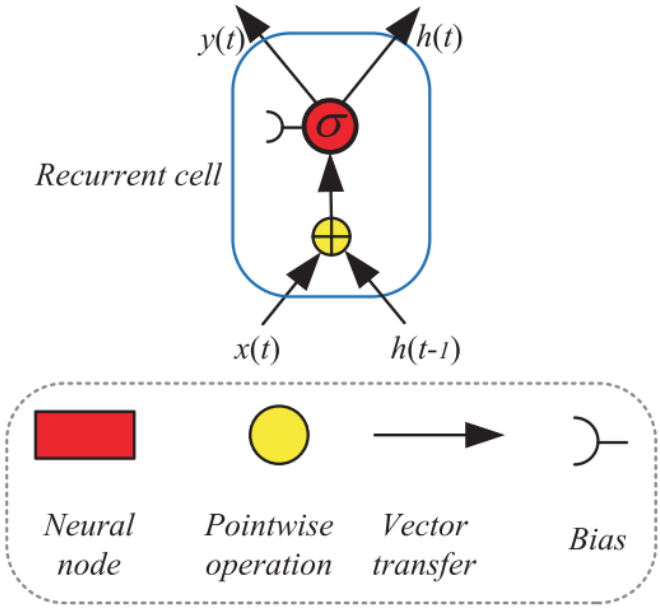



Figure 2.8: Diagram of a vanilla RNN cell. Here, the input $x(t)$ at time $t$ is combined with the hidden state at the previous timestep $h(t-1)$ and fed into the neural node. The output takes the form of a traditional activation $y(t)$ and a new hidden state $h(t)$ to be used with the next input $x(t+1)$. Figure taken from [74].

problem. The equivalent problem whereby gradients less than 1 vanish towards 0 is known as the gradient vanishing problem. This results in the model having no ability to learn long term dependencies. Both of these problems are compounded by RNNs learning from long sequences of inputs, causing longer multiplicative chains of updates. The gradient exploding problem can be tackled using strategies such as gradient clipping, whereby the gradient is reset to a small number after it exceeds a threshold value using

$$g = \frac{\text{threshold}}{\|g\|} g \tag{2.15}$$

A common approach to dealing with the gradient vanishing problems is to replace vanilla RNN cells with long short-term memory (LSTM) units or gated recurrent units (GRU).

### LSTM

LSTM was first proposed by Hochreiter and Schmidhuber [29] and has seen excellent results in fields such as speech recognition [27] and acoustic modelling [55]. LSTM describes an alternative structure for RNN whereby the hidden units are replaced with a memory cell which can store information for extended periods. The flow of information in and out of the cell is controlled by three boolean gates: a forget gate (2.16) which controls how information is lost from memory, an input gate (2.17) which controls how new information is stored in memory, and an output gate (2.18) which controls which information in memory should be output to the next layer and for the next time step.

$$f(t) = \sigma(W_f \cdot [h(t-1), x(t)] + b_f) \tag{2.16}$$

$$i(t) = \sigma(W_i \cdot [h(t-1), x(t)] + b_i) \tag{2.17}$$

$$o(t) = \sigma(W_o \cdot [h(t-1), x(t)] + b_o) \tag{2.18}$$

$$\tilde{c}(t) = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_{\tilde{c}}\right) \tag{2.19}$$

$$c(t) = f(t) \odot c(t-1) + i_t \odot \tilde{c}(t) \tag{2.20}$$

$$h(t) = o(t) \odot \tanh(c(t)) \tag{2.21}$$

Here, $\odot$ represents the Hadamard product (element-wise multiplication) and $W_{\{f,i,o,c\}}$ and $b_{\{f,i,o,c\}}$ are trained weights matrices and biases respectively. The above equations describe how an input vector $x(t)$ at time step $t$, the previous hidden state $h(t-1)$ and cell state $c(t-1)$ combine to compute the next hidden state $h(t)$ and cell state $c(t)$. This process can be visualised in Figure 2.9.

The key conceptual difference between LSTM and RNN is that LSTM separates memory from the hidden state. LSTM deals with the gradient vanishing problem since the memory cells are additive with respect to time [16].

### GRU

GRU was first proposed by Cho et al. [13]. GRU shares many similarities with LSTM in that it replaces hidden units in a RNN with a memory cell, however it improves on LSTM in that it has fewer parameters, and therefore has lower computational burden. The reduction in parameters is achieved by combining the input gate and the forget gate

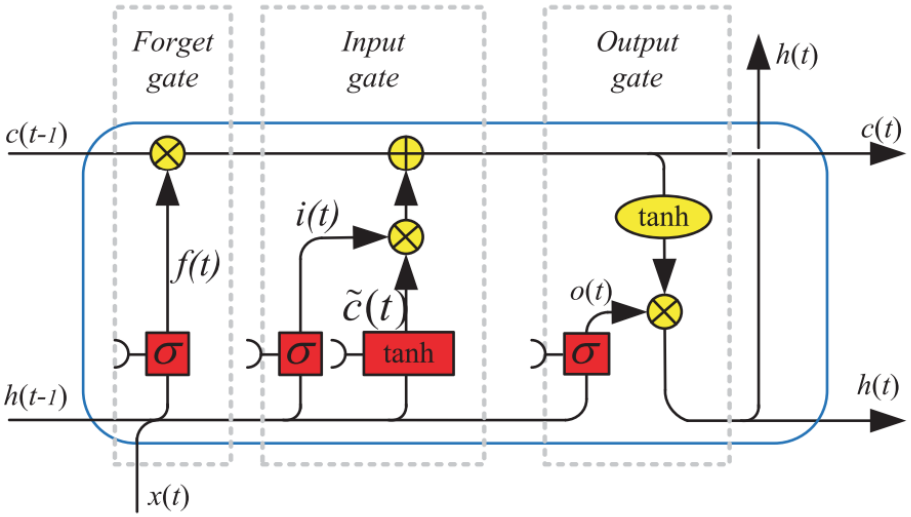



Figure 2.9: Diagram showing the structure of a LSTM cell. As with the vanilla RNN cell shown in Figure 2.8, the $x(t)$ and $h(t-1)$ combine but this time with a cell state $c(t-1)$. The output now takes the form of a new cell state $c(t)$ and hidden state $h(t)$. Figure taken from [74].

of an LSTM to form an update gate, meaning that the GRU has only two gates, an update gate (2.23) and a reset gate (2.22). The GRU cell can be seen in Figure 2.10.

$$r_t = \sigma\left(W_{rh}h_{t-1} + W_{rx}x_t + b_r\right) \tag{2.22}$$

$$z_t = \sigma\left(W_{zh}h_{t-1} + W_{zx}x_t + b_z\right) \tag{2.23}$$

$$\tilde{h}_t = \tanh\left(W_{hh}(r_t \odot h_{t-1}) + W_{hx}x_t + b_z\right) \tag{2.24}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.25}$$

The reduction in parameters and number of gates does mean that the GRU is less powerful than the LSTM and does not work for problems such as translation [8]. However, it has been applied to birdsong related problems with promising results [52, 2].

While the usage of RNNs, LSTMs or GRUs alone for birdsong classification has been rare in the literature, there are several examples of experiments where convolutional layers and recurrent layers have been combined to form a Convolutional Recurrent Neural Network (CRNN) [73, 49] to leverage the benefits of both architectures. A typical ap-

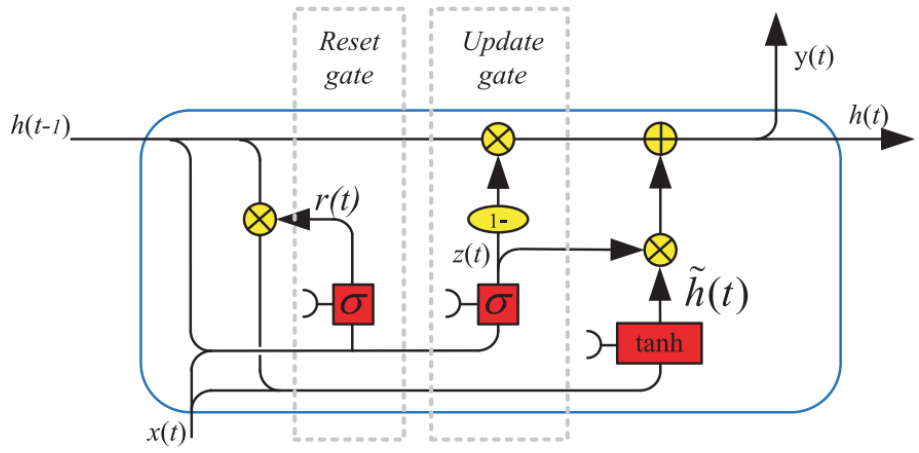



Figure 2.10: Diagram showing the structure of a GRU cell. This follows the same concept as LSTM in that cells have a memory, but now only the input $x(t)$ is combined with the previous hidden state $h(t-1)$ to form the new output $y(t)$ and hidden state $h(t)$. Figure taken from [74].

proach using a CRNN with regards to birdsong classification might look something like the following [15]:

1. Convert an audio segment to a frequency domain representation to be interpreted as an image, e.g. a spectrogram. Ideally the image will encode some form of temporal evolution to be utilised by the recurrent layers later on.

2. Apply a convolutional layer (or layers) to each time step of the representation to extract local features. A time step is defined as a single unit along the time axis of the spectrogram.

3. Apply a recurrent layer to spread the local features of a single time step to its temporal surroundings. The spread features can then be processed by a fully connected layer with softmax activation as described in Section 2.3.2 to make a prediction in the form of a distribution over output classes for each time step.

The recurrent layer provides two key advantages compared to a plain CNN. Firstly, the network will be able to learn from previous and future features in order to strengthen predictions. For example, if the current time step equally supports a blackbird or a nightingale prediction by itself, it will be skewed towards a blackbird if the previous time step and future time step both predicted a blackbird. Secondly, the network can learn to include

moments of silence in a prediction. Note that this benefit is lost if just considering audio segments of isolated syllables as these segments should contain little or no silence.

## 2.4 Chapter summary

In this chapter we started by examining different ways of evaluating the performance of models used in the literature. We then discussed the structure of birdsong at a high level and used that as a motivation for examining popular features to be used in this thesis. We concluded the chapter by looking in detail at models commonly used to classify audio data and determining their suitability for this work.

# Chapter 3

# Methodology

In this chapter we first discuss the data collection and processing steps involved to generate a database of samples to be used for training and testing. In Section 3.4 we explore the different options for feature representation and lay out some experiments to be conducted. In Section 3.5 we explore the structure of the models used for classification and discuss how training and/or optimisation will be performed. We finish the chapter by commenting on how the models will be evaluated.

All computational work for this thesis was performed in MATLAB vR2023a unless stated otherwise.

## 3.1 Collecting the datasets

Since all the models described in Section 2.3 are trained in a supervised learning fashion, this requires a dataset of labelled training examples. The website xeno-canto[1] (XC) houses the largest and most comprehensive publicly available collection of birdsong samples in the world. It has made an enormous impact in the field of birdsong recognition since its inception in 2005 and has been the source of the datasets for the annual BirdCLEF challenge since 2014 [70]. All samples available on XC are labelled with the bird species and contain rich metadata, such as the time and location of the recording. Importantly, all samples have a crowd-sourced rating from A — E which signifies how clear the recording sample is, where A denotes samples with the highest quality and clarity and E denotes the lowest. This is especially important should we wish to experiment with different signal-to-noise ratios (SNR) as the noise can be manually added to a clean recording at precise

[1] `http://xeno-canto.org`

SNR levels.

Due to its scale and reputation in the birdsong classification community, all samples used in this thesis have been downloaded from the XC repository. All samples downloaded have an A rating and have been labelled as a 'song' rather than a 'call'. The samples are stored in a directory labelled according to the first three letters of the species' binomial name. For example, samples from the common blackbird (*Turdus merula*) are stored in a directory called 'TURMER'. This directory name acts as a label for training and test samples.

In this work we are mostly concerned with testing relative improvements in model performance after tuning various components, such as novel feature representations. To this end, we select the most simple form of classification experiment: binary classification. The two bird classes selected for all experiments herein are the common blackbird and the common nightingale (*Luscinia megarhynchos*) due to the two birds' vocalisations being acoustically similar and hence more difficult to differentiate resulting in more variable accuracy results, and the author's personal preference.

Roughly 38 individuals from each class were downloaded, with 28 used for training and the remaining kept for testing. All samples were checked for suitability and downloaded manually.

## 3.2 Pre-processing audio recordings

Some recordings available on XC are recorded in stereo sound, so to reduce dimensionality without losing too much information, all stereo recordings are first converted to mono by taking a mean average of both channels. Leading and trailing sections of background noise are then stripped from the recording using the `detectSpeech` function provided by MATLAB. This function uses a thresholding algorithm to detect onset and offset indices of speech [24]. The function works for birdsong since birdsong frequencies reside in a similar range to that of human speech. The function accepts various arguments to determine properties such as window length and threshold value, but from initial experiments the function was shown to work well with birdsong using the defaults, see Figure 3.1. The detected onset and offset of birdsong allows for the leading and trailing bits of background noise to be removed from the recording, thus reducing unnecessary computational time in segmenting the audio.

A highpass filter is then applied to the recording with a passband frequency of 450 Hz

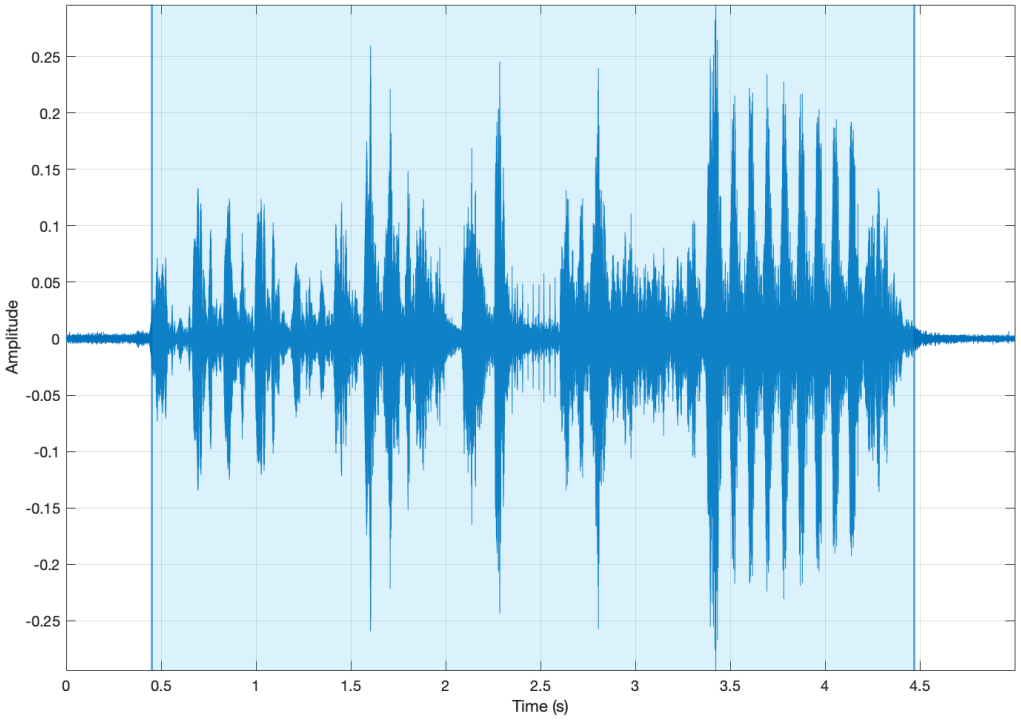



Figure 3.1: Output of the `detectSpeech` function for a recording of birdsong from a Eurasian wren. The highlighted section shows where the algorithm has detected 'speech', or in this case, birdsong.

using MATLAB's `highpass` function. This value was chosen as most birdsong sits in the range of 1 kHz — 10 kHz, and so lower frequencies can be attributed to background noise and therefore their removal should reduce noise in the system without losing important information. The reason why frequencies above 10 kHz are not removed is that, as can be seen from Figure 2.2, birds typically produce higher frequency harmonics when vocalising, shown as vertical bands moving upwards from where syllables are located on the spectrogram. These harmonics may capture useful information for the model to utilise and should therefore not be filtered out.

After these pre-processing steps have been taken, the audio is ready for syllable segmentation.

## 3.3 Segmenting audio recordings

Recordings downloaded from XC were also chosen based on their length. Recordings of duration 40 seconds to 120 seconds were preferred since they were likely to be long enough to include enough variety of vocalisations from one individual bird, but not too long so as to take up too much space on disk. In this thesis it was preferred to have fewer samples from more individuals as opposed to more samples from fewer individuals for each species. The intention of this was to introduce more variation in the training samples for each class, especially when considering that many birds of the same species but residing in different locations have demonstrated subtle variations in vocalisations, known as dialects [5]. As a general rule of thumb, a target of roughly 50 training samples per individual was aimed for in this thesis.

This leads to the question of how to generate samples. In the literature there seem to be two main ways of segmenting recordings into samples to be used for training. The first is segmenting a recording into overlapping segments of a certain length, typically in the range of 4 — 11 seconds [73, 15]. This has the advantage that all samples will be of fixed length and that the segmentation algorithm is very simple. However, it will likely mean that some samples will contain only background noise which, if used for training, will introduce noise into the system. These noise samples therefore may need to be removed, either manually [73] or using an algorithm [51]. The other method involves segmenting the recording into syllables [23, 56]. This has the advantage that all training samples are likely to contain little or no noise. However the samples will be of variable length, so steps will be needed to be taken to compare the samples, such as adding padding or more advanced techniques like dynamic time warping [65]. Syllable segmentation also has the enticing prospect of being able to identify a bird species from a very short sample. This could be useful in situations where a recording is mostly corrupted by background noise but has a few small segments of clear birdsong.

In this work we attempt to combine the benefits of both approaches by first segmenting a recording to retrieve the syllables, then combining the syllables to form a sequence. The following sections will describe both processes in detail.

### 3.3.1 Energy based syllable segmentation

This algorithm was first proposed by Fagerlund [22] and has been used in various research papers since [65, 56]. Note that Fagerlund proposed two different segmentation methods

in the paper. In this thesis we use the energy based segmentation method as it's relatively simple to implement and its performance can be superior to the other method proposed. At the time of writing there was no open source code available to perform this algorithm so it was written from scratch for this thesis. The algorithm is computed in the following 3 phases:

1. Determine the onsets and offsets of syllable candidates. Special attention has to be paid to the border effect [42], whereby the energy envelope near onsets and offsets fluctuates around a threshold.

2. Syllable candidates that are within a certain temporal distance with each other are merged into one. The initial separation of candidates can likely be attributed to the border effect, and so the candidates can be thought of as originating from one syllable in reality.

3. Syllable candidates that are shorter than a minimum threshold are removed. These candidates are often caused by random fluctuations in the signal energy and are unrelated to bird vocalisations.

**Onset/offset detection**

The first phase considers the energy of an input signal. The energy envelope is the key variable to use in determining the onset and offset of a syllable. The general idea is that when the energy increases past a threshold, this marks the start of a syllable candidate and when the energy then falls below the same threshold, this marks the end of a syllable candidate. To calculate the energy envelope, the recording is divided into overlapping frames. Fagerlund [22] recommends a frame size of 128 samples and an overlap of 50%. This corresponds to a frame length of roughly 3ms. Since distance between syllables can be as short as 20ms, the frame size needs to be small enough so that there are at least a few frames between candidates to be able to efficiently discriminate between syllables.

Frames are first windowed using a Hanning window, which minimises the effects of the discontinuities at the edges of the frames. The energy $E_i$ in the decibel scale for frame $i$ is calculated as

$$E_i = 10 \log_{10} \sum_{j=1}^{K} x(i)_j^2 \tag{3.1}$$

where $x(i)_j$ is the $j^{\text{th}}$ input signal sample value in the $i^{\text{th}}$ frame and $K$ is the total number of samples in each frame, in this case 128. The maximum energy of the entire signal is

normalised to 0dB. The initial noise level estimate is set to be equal to the global minimum value of the energy envelope and the threshold for the onset and offset detection is set to half the noise estimate. The flow diagram for onset and offset detection can be seen in Figure 3.2.

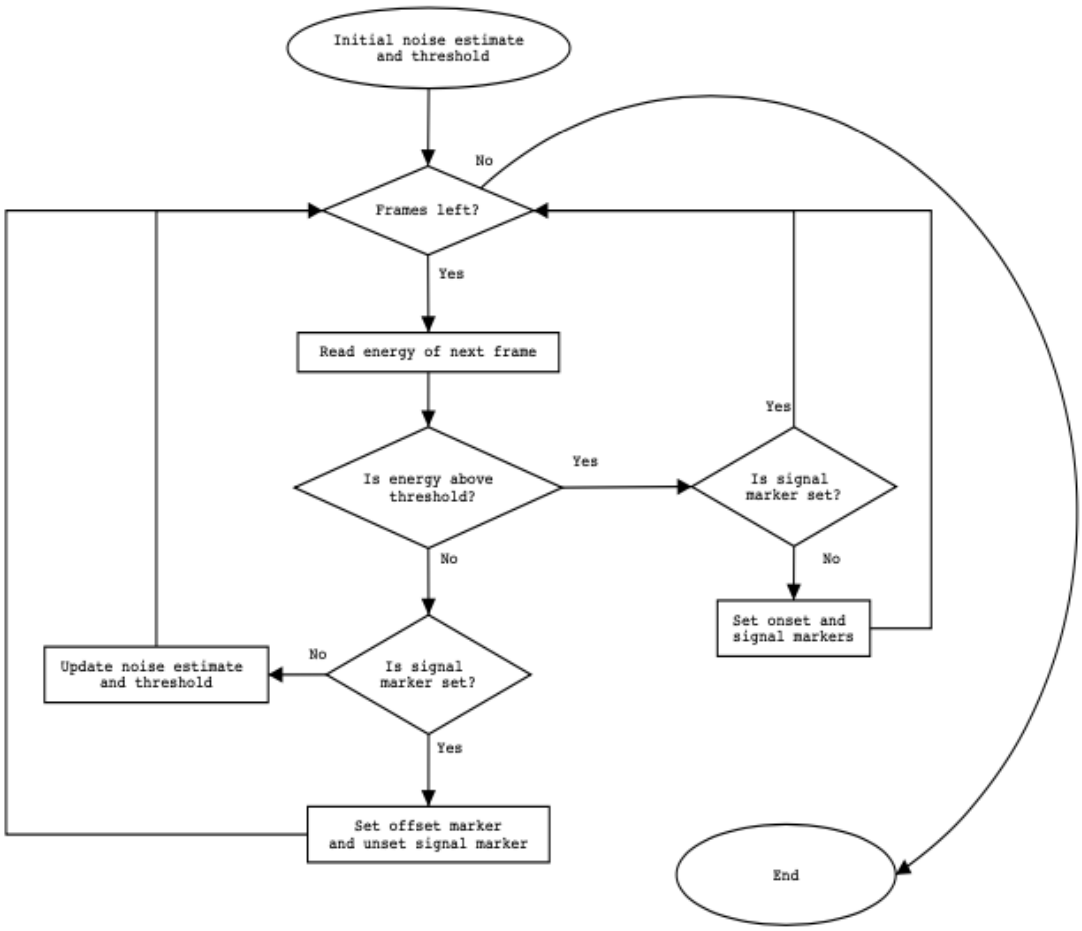



Figure 3.2: Flow chart of the syllable segmentation onset and offset detection algorithm. Figure taken from [22].

In essence, the algorithm moves through each frame and computes whether the frame should be assigned to a syllable or not, updating noise and energy thresholds as it goes. The noise estimate is updated as the average energy of frames already processed by the algorithm that are not assigned to a syllable.

### Merging syllable candidates

The algorithm may produce syllable candidates that are very temporally close together. This may be from the bird producing short bursts of pulses that constitute the same syllable, or it may be from the border effect. In both cases the candidates can be considered as belonging to the same syllable and so can be safely merged together. Candidates that are less than 15ms apart are selected for merging [22]. The process of onset/offset detection and merging can be seen in Figure 3.2.

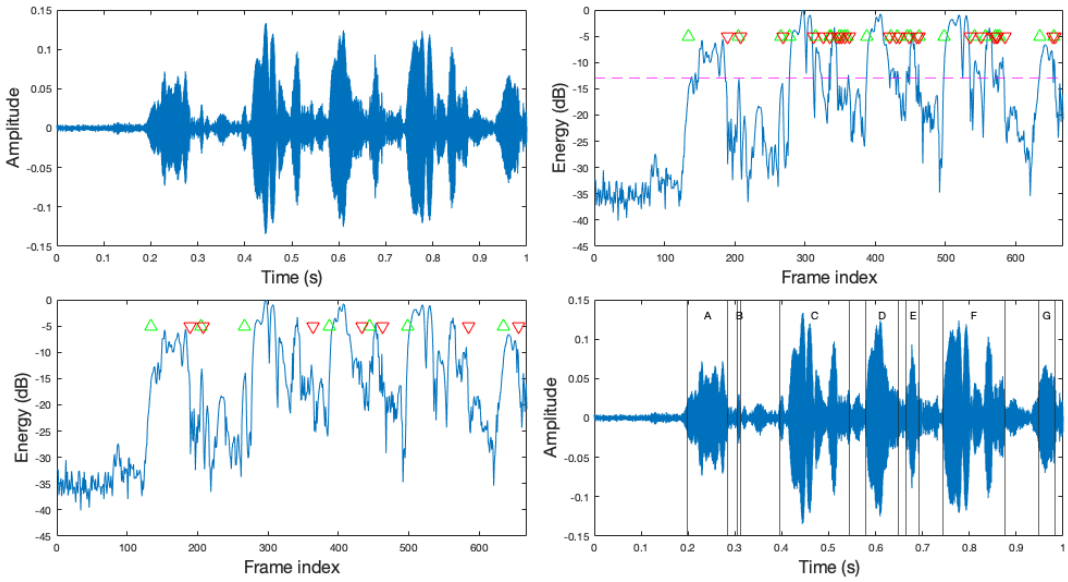



Figure 3.3: Figure showing the syllable detection algorithm. Top left shows the amplitude varying over time for a sample of Eurasian wren birdsong. Top right shows the detected onset (green triangle) and offset (red triangle) markers overlaid on the energy envelope in the decibel scale of the sample. The dashed magenta line shows the final threshold for syllable detection. Bottom left shows the markers after merging has been performed. Bottom right shows the detected segmented syllables, labelled from A — G. Note that the next step, removing syllables less than a minimum duration, would remove syllable candidate B since it is less than the minimum duration threshold.

One thing to note about the algorithm is that the energy threshold is updated iteratively as the algorithm moves through the signal. This means that candidates detected near the start of the sample may not get detected if they were nearer the end of the signal. This is the case in Figure 3.3 with syllable candidate B, since its energy peak is below the final energy threshold, shown in the top right panel of the figure. This shouldn't pose an issue

since candidates less than the minimum duration threshold are removed, as is the case with candidate B.

Each segmented syllable is then stored as an individual *wav* file in the same parent directory as the original recording with a filename in the format $\{XCID\}\_i.wav$ where $\{XCID\}$ is the XC id of the recording and $i$ is the index of the syllable.

### 3.3.2 Sequence sample generation

Once the syllables have been extracted, they are ready to be combined to form a sequence. This style of sequence generation will be used for all training in this work unless otherwise stated.

We first start by grouping syllables by XC id. Then, for each syllable in each group, the following syllables are appended until a fixed sample length is reached. If a syllable is added which pushes the sample length over the limit then the sample is trimmed at the fixed length. If there aren't enough syllables to reach the fixed length, then the sample is discarded. This process can be visualised in Figure 3.4.

This approach is motivated by Somervuo et al.'s work [65] in showing that training and classifying using single syllables returns suboptimal results, whereas using sequences of syllables gives much improved accuracy. Of course there is a tradeoff here between training with longer sequences, and hence more information, versus increased computational demand to perform training.

From Figure 3.5, we can see most syllable lengths fall in the region $[265, 1236]$ (the segmentation algorithm sets a minimum of 265 samples for candidate syllables). Therefore a sequence of fixed length about 14000 (denoted by the red dashed line) should contain enough syllables to capture sufficient information about the vocalisation. The fixed length, equating to about 300ms duration, also ensures that most sequences contain at least one full syllable. This approach ensures that the influence of background noise is kept to a minimum, while still maintaining information related to the temporal evolution of a particular birdsong.

## 3.4 Extracting features from samples

Once the sequences have been generated, they are ready to be converted to features. The following sections describe the feature extraction processes used in this work.

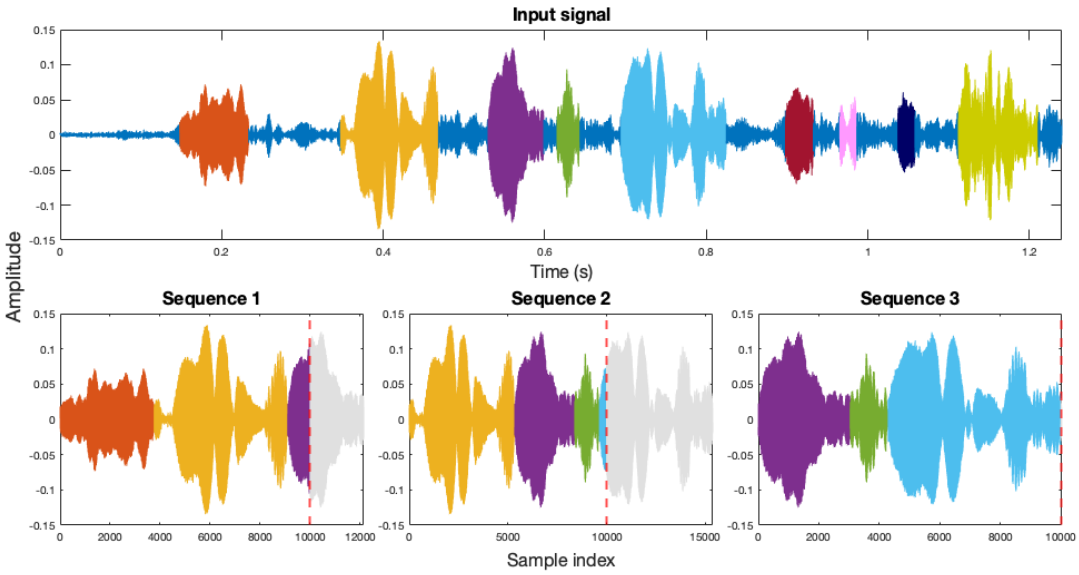



Figure 3.4: Figure demonstrating building sequences out of an input signal. The top panel shows the input signal with syllables discovered during segmentation highlighted in colours. The bottom 3 panels show the first 3 sequences generated from the input signal. The red dashed line indicates the fixed length, in this case 10,000 samples. Syllable segments that have been trimmed as they push the sequence over the fixed length are shown in grey. Note that for Sequence 3 its constituent syllables make up exactly 10,000 samples, and so no trimming was necessary.

### 3.4.1 Cepstral coefficients

The cepstral coefficients on which this thesis focuses are the MFCC and GTCC. Both features are extracted using builtin MATLAB functions `mfcc` and `gtcc` respectively. The functions accept arguments which affect how the routine processes input signals to calculate the coefficients. To the best of the author's knowledge, these arguments have not been explored in the field of birdsong classification problems.

#### MFCC

A potentially overlooked parameter of MFCC extraction with regards to birdsong classification using MATLAB is the `BandEdges` argument. This argument provides the basis of the half-overlapped triangular filters that `mfcc` uses. By default, `BandEdges` is a 42-element vector that results in a 40-band filter bank constructed using 13 linearly-spaced filters, with 133.3 Hz separating their centre frequencies, and 27 log-spaced filters, sepa-

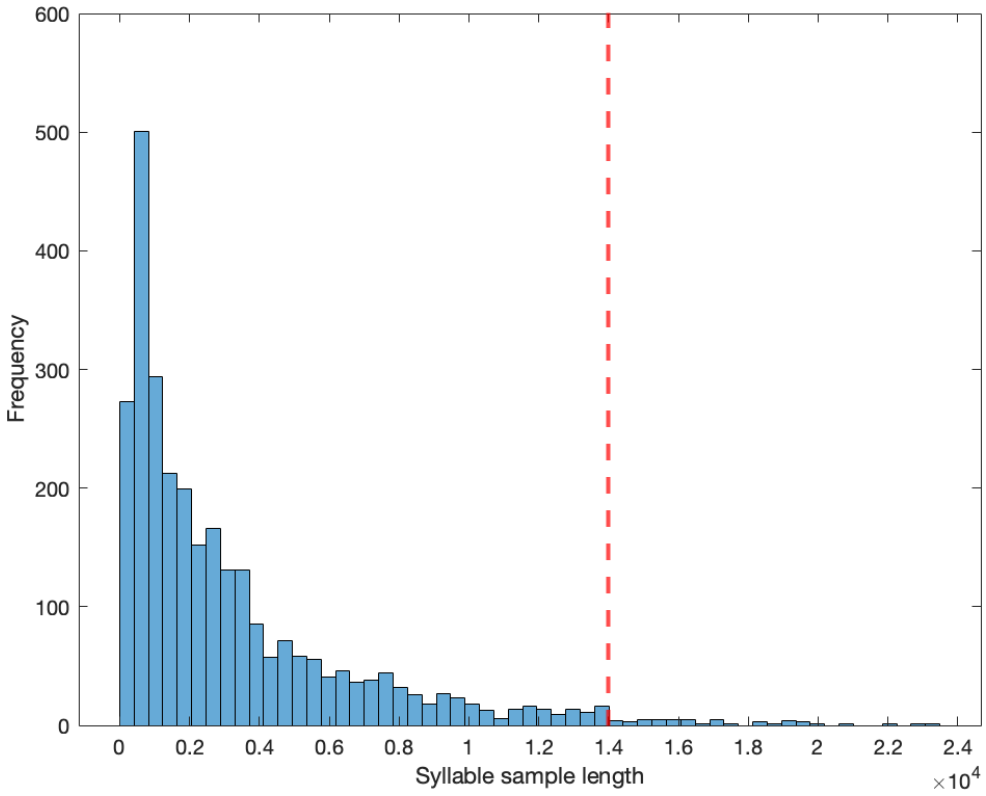



Figure 3.5: Histogram showing the frequency of different sample lengths for all training syllables. A candidate fixed sequence length is denoted by the red dashed line.

rated by a factor of 1.0711703 in frequency [64]. This equates to a frequency range that spans approximately 133 Hz to 6864 Hz. This range is well suited to human speech, of which MFCC is often used as a feature, but may be less suitable for birdsong which can sometimes sit outside the normal frequency range of human speech.

To test the `BandEdges` argument, 7 experiments were devised and evaluated to see if any improvements to binary bird classification models utilising MFCC could be made. The experiments are listed in Table 3.1.

### Comments on experiment setup

In the table, 'fs' refers to the frequency sampling rate for a given audio file, usually 44.1 kHz or 48 kHz. fs/2 is the maximum possible value of the vector supplied for the `BandEdges` argument. 'A-mel', short for anti-mel, denotes a set of mel filterbanks that have been

| Experiment # | Number of bands | Band type | Approximate frequency range (Hz) |
|---|---|---|---|
| 1 | 40 | Mel | 133 — 6864 |
| 2 | 40 | Mel | 50 — fs/2 |
| 3 | 80 | Mel | 50 — fs/2 |
| 4 | 40 | Linear | 50 — fs/2 |
| 5 | 40 | Linear | 133 — 6864 |
| 6 | 40 | A-mel | 133 — 6864 |
| 7 | 40 | Mel | 133 — 10000 |

Table 3.1: Description of experiments used for testing H1 with MFCC.

inverted, such that the centre frequencies are closer together at higher frequencies and further apart at lower frequencies. 'Linear' refers to a set of filterbanks that have been spaced linearly on the frequency scale. This has the effect of transforming the MFCC coefficients to Linear Frequency Cepstral Coefficients (LFCC). Mel band edges were generated by first converting a minimum and maximum frequency to the mel scale using

$$m = 1127 \cdot \ln(1 + \frac{f}{700}) \tag{3.2}$$

where $m$ is the corresponding mel frequency for frequency in Hz $f$. A vector of band edges was created using MATLAB's `linspace` function with the minimum and maximum mel-frequency and desired number of bands. The vector was then converted back to the Hz scale using the inverse of Equation (3.2) and passed to the `BandEdges` argument. Antimel band edges were created by doing the same process and then flipping the result. Linear band edges were created by simply applying a `linspace` function to a minimum and maximum frequency in Hz. The different band types can be visualised in Figure 3.6.

The experiment design was motivated as follows:

Exp 1: These are the defaults provided with the `mfcc` function and acts as a control experiment.

Exp 2: A wider frequency range may be able to make use of the higher frequency harmonics produced by most birdsong.

Exp 3: A wider frequency range means the bands have a larger bandwidth and may lose some distinguishing power. Increasing the number of bands may help to alleviate this.

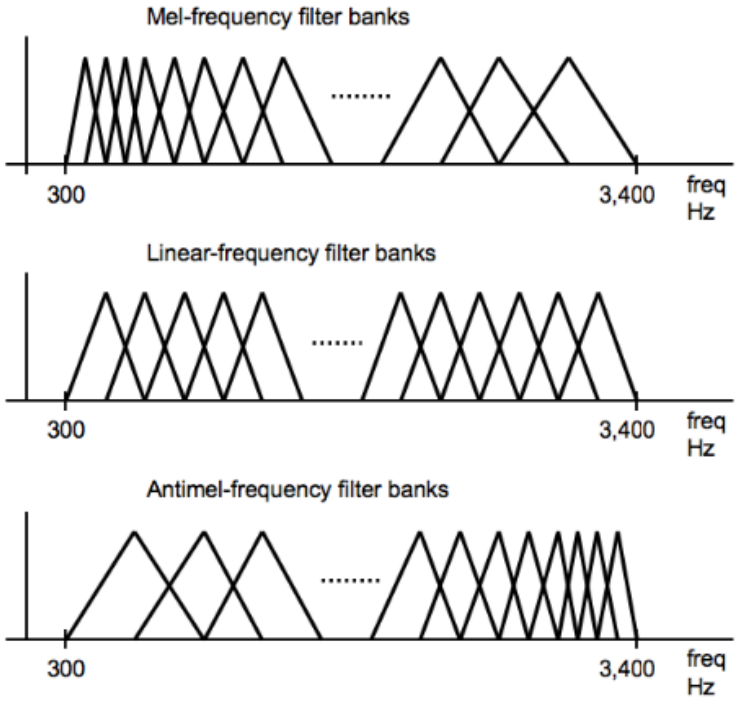



Figure 3.6: Illustration of the mel-, linear-, and antimel- frequency filterbanks. Figure taken from [40].

Exp 4: Since the human auditory system hasn't evolved specifically to be able to distinguish birdsong, it's entirely possible that the mel scale isn't the optimum scale to use. A linearly spaced set of filterbanks may provide a good generalised attempt at distinguishing birdsong.

Exp 5: Similar to Experiment 4 but focused on a frequency range that more tightly fits around the dominant frequencies produced by bird vocalisations.

Exp 6: The anti-mel scale has been used with interesting results in human speech recognition through telephones [40]. It offers more distinguishing power at higher frequencies, and since birdsong is typically produced at higher frequencies than human speech, the anti-mel scale may be able to better distinguish birdsong.

Exp 7: Similar to Experiment 2 but still focused on the main frequencies emitted by birds.

### GTCC

Like `mfcc`, `gtcc` has a potentially overlooked argument in `FrequencyRange`. This two-element row vector specifies the minimum and maximum values in Hz of the gammatone filter bank used by the `gtcc` routine. The default value is `[50, fs/2]`, which could potentially be fine-tuned for birdsong related problems since birdsong frequencies do not typically go below 1 kHz.

To test the `FrequencyRange` argument, 4 experiments were devised and evaluated to see if any improvements to binary bird classification models utilising GTCC could be made. The experiments are listed in Table 3.2.

| Experiment number | Frequency range (Hz) |
|---|---|
| 1 | 50 — fs/2 |
| 2 | 100 — 7000 |
| 3 | 50 — 10000 |
| 4 | 400 — 6000 |

Table 3.2: Description of experiments used for testing H1 with GTCC.

### Comments on experiment setup

As with the MFCC experiments, 'fs' refers to the frequency sampling rate for a given audio file. fs/2 is the maximum possible value for the `FrequencyRange` argument. By default the filter bank is a gammatone filter bank, which can be visualised in Figure 3.7. Note that it's not possible to change the number of bands used by `gtcc`.

The experiment design was motivated as follows:

Exp 1: These are the defaults provided with the `gtcc` function and acts as a control experiment.

Exp 2: This more closely matches the effective frequency range for the control experiment for `mfcc`. This range matches human speech and contains most birdsong frequencies.

Exp 3: Similar to Experiment 2, but widens the range to account for frequencies outside of Experiment 2's range.

Exp 4: Similar to Experiment 2, but tightens the range to achieve more discriminatory power.

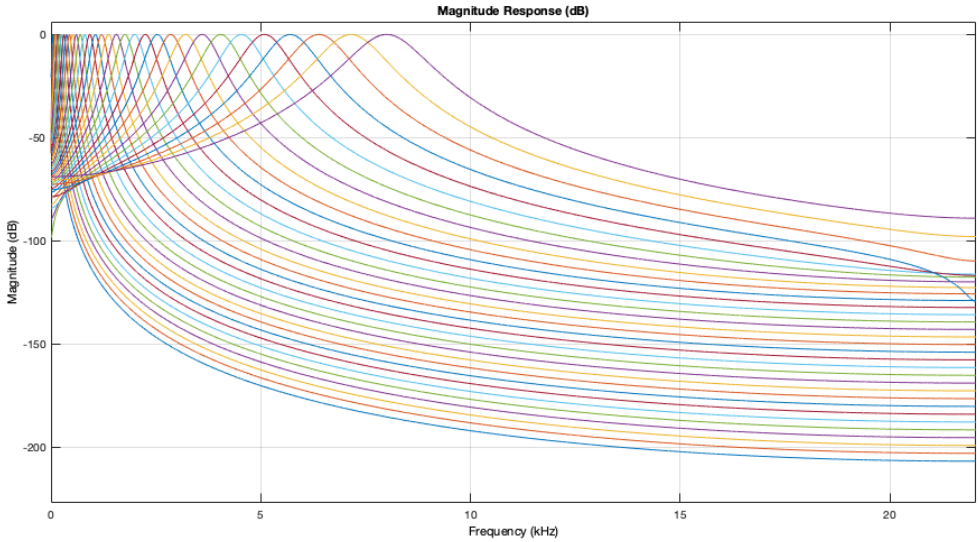



Figure 3.7: Illustration of a typical gammatone filterbank.

### 3.4.2 MRCG

MATLAB provides no builtin functions for computing a MRCG, or even a cochleagram, upon which the MRCG is based. To that end, custom code was developed based on the toolbox developed by Kim and Hahn [33] which implements MRCG. The MRCG routine implemented in [33] uses a gammatone filter bank with a frequency range of `[50, 8000]`, which may not be suitable for birdsong problems. The work done in customising the toolbox for this thesis extends the implementation to work with alternative frequency ranges for the gammatone filter bank.

It's recommended to include the $\Delta$ and $\Delta\Delta$ values along with the MRCG feature [12], so hereafter the term MRCG shall refer to the base MRCG along with its $\Delta$ and $\Delta\Delta$ values. The same applies to MFCC and GTCC. While the code from [33] includes a custom implementation of the delta calculation, in this thesis the `audioDelta` function was used due to it being a builtin function in MATLAB and therefore requiring no customisation.

Due to the high dimensionality of the MRCG feature and considering the computational limitations of the machine upon which the training for this thesis was performed, the MRCG feature would be best suited to a classifier which is optimised for working with image data, such as CNNs. To that end, it won't be used with SVMs in this work.

## 3.5 Training models

To create the training MATLAB cell array, sequences for each class are generated and stored in a cell array. A maximum limit of 40 sequences per individual is imposed to ensure enough variation while keeping training times to a minimum. The number of training sequences used for both classes can be seen in Table 3.3.

| Class | Individuals | Sequences |
|---|---|---|
| Common blackbird (*TURMER*) | 28 | 1107 |
| Common nightingale (*LUSMEG*) | 28 | 1200 |

Table 3.3: Number of training sequences for used for both classes.

The training cell is formed by vertically concatenating the two cell arrays containing the training sequences to form a $2307 \times 1$ cell array. The training labels are contained in a $2307 \times 1$ categorical array with a *TURMER* category labelling a blackbird sequence and *LUSMEG* labelling a nightingale sequence.

All training is performed on an Apple MacBook Pro (late 2013) running OSX 11.5.2.

### 3.5.1 SVMs

All SVM models are implemented using MATLAB's `fitcsvm` function provided in the Statistics and Machine Learning Toolbox.

SVMs require each training sample to be a row vector. To convert the training cell array to the required matrix format, a function can be applied to each cell using the `cellfun` function to extract the desired feature and flatten the output to a row vector. A simple routine can then be applied to convert the cell array to a matrix of numerical data, the required format for `fitcsvm`. The training labels cell array mentioned in the previous section requires no transformation.

#### Linear

Linear SVMs are trained by supplying a `KernelFunction` argument set to `linear` to the `fitcsvm` function. Linear SVMs are reasonably simple to fit as they have fewer hyperparameters than SVMs utilising other kernel functions. The key hyperparameter to tune

here is `BoxConstraint`, which controls the trade-off between maximising the margin between classes and minimise the classification error. This corresponds to the constant $C$ in Equation (2.12). The `Standardize` parameter controls whether data is standardized before fitting, and can also affect the performance of the model. In this work, KFold cross-validation (KF) was selected to optimise the parameters, with $K = 7$ folds. This is motivated by other research taking the same approach with promising results [56]. MATLAB provides builtin ways to optimise hyperparameters, which can be achieved by passing a `cvpartition` to the `HyperparameterOptimizationOptions` argument and supplying an `OptimizeHyperparameters` argument set to the name of the desired hyperparameters to tune to `fitcsvm`, in this case a cell array of `BoxConstraint` and `Standardize`.

Figure 3.8 shows where the optimum values of `BoxConstraint` and `Standardize` lie for a linear SVM. As can be seen, the `Standardize` parameter makes a slight difference on model performance, whereas `BoxConstraint` makes a significant difference, with lower values preferred. A lower value means that the model has greater flexibility in misclassifying training samples for the benefit of better generalisation.

### RBF

RBF SVMs are able to fit more complex decision boundaries by projecting data into a higher dimensional space, which is controlled by the hyperparameter `KernelScale`. RBF SVMs are also sensitive to the `BoxConstraint` parameter. 7-fold cross validation was performed using the same routine as the linear SVM but this time by supplying an `OptimizeHyperparameters` argument set to `auto` to `fitcsvm`. A value of `auto` means that both `KernelScale` and `BoxConstraint` are optimised during training.

Figure 3.9 shows where the optimum values of `BoxConstraint` and `KernelScale` lie for a RBF SVM. Both parameters are shown to have an impact on the model performance, with high values of `BoxConstraint` and mid-range values of `KernelScale` preferred. The `KernelScale` controls the width of the kernel, with a lower value resulting in a narrow kernel which can be interpreted as a more 'wiggly' decision boundary. Higher values of `KernelScale` result in a smoother decision boundary. This corresponds exactly to the bias-variance tradeoff, whereby training fit and generalisation directly compete with each other.

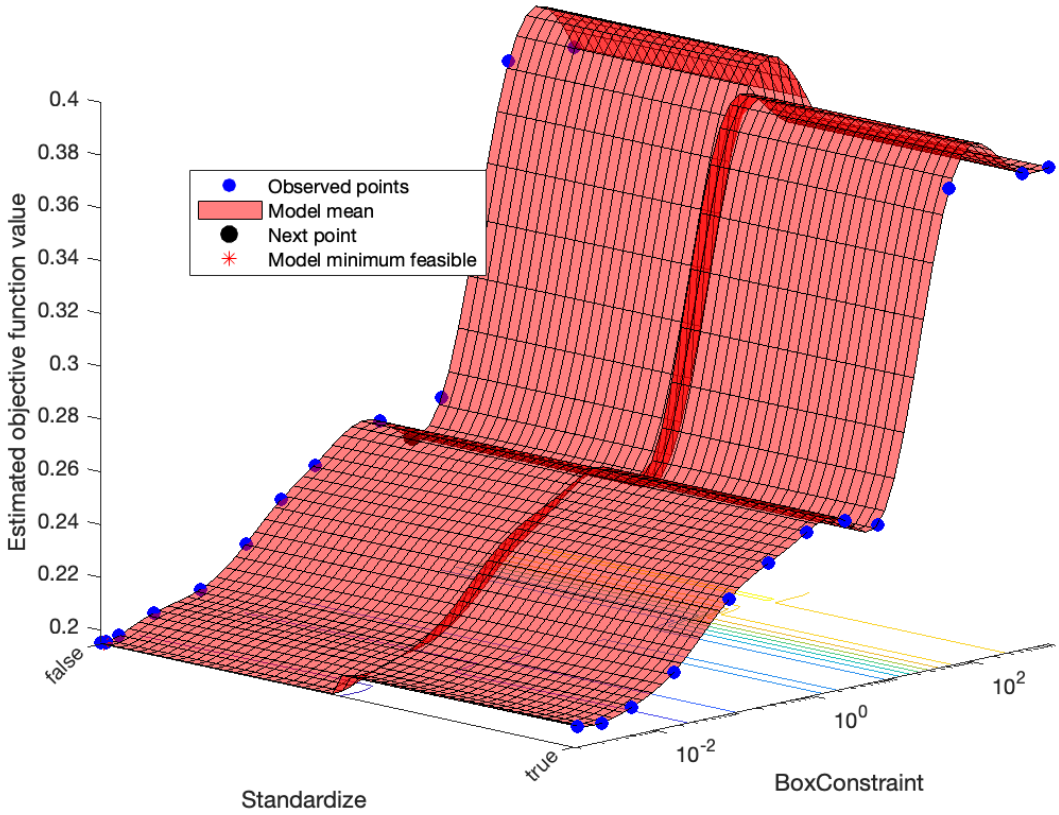



Figure 3.8: Plot showing optimum values of the `BoxConstraint` and `Standardize` parameters when fitting a linear SVM. The optimum values are located where the objective function is at a minimum.

### 3.5.2 Neural networks

All NN models are implemented using tools provided in MATLAB's Deep Learning Toolbox. NN models are trained by first generating an array of sequential layers that describe the network using builtin layer structures such as `reulLayer` (which denotes a layer of ReLU activations), and `convolutional2dLayer` (which denotes a 2D convolutional layer). The layers are then passed to the `trainNetwork` function along with a set of options that describe hyperparameters such as the mini-batch size and the optimisation algorithm used, as well as the training input and corresponding labels.

The training input for NN models depends on the type of NN. Some can work directly on cell arrays whereas others require training data in numerical matrix format. For all NN training runs, 5% of the training set was set aside for validation. If the validation accuracy

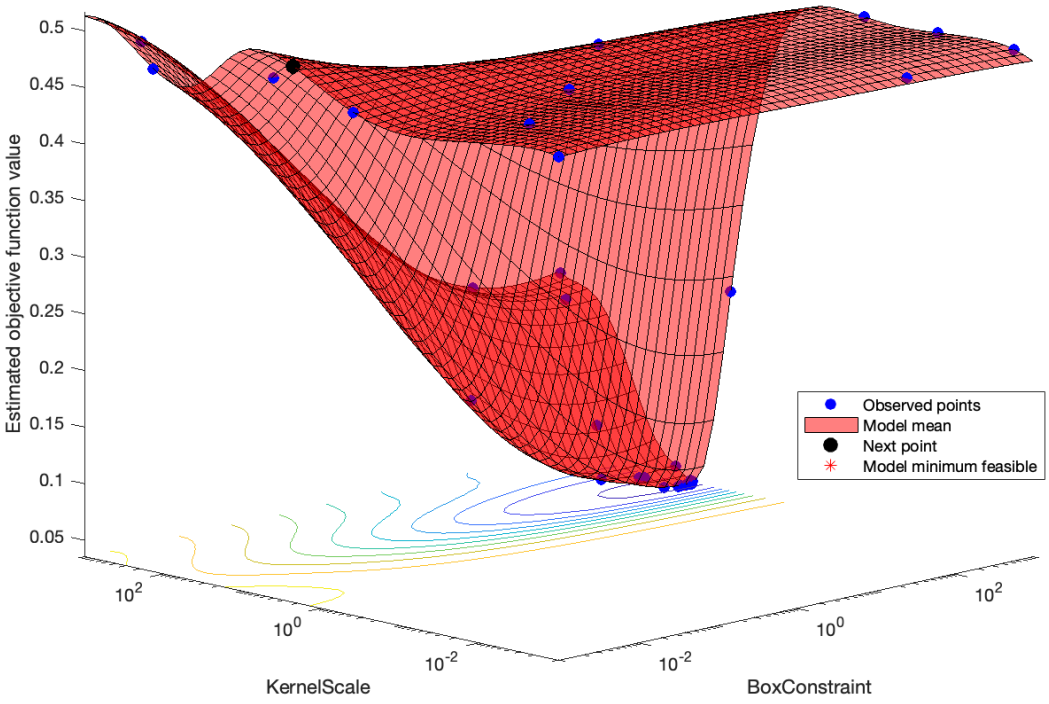



Figure 3.9: Plot showing optimum values of the `BoxConstraint` and `KernelScale` parameters when fitting a RBF SVM. The optimum values are located where the objective function is at a minimum.

was seen to plateau during training then this was considered grounds for early stopping of training.

### CNN

Unlike SVMs, which expect a training sample to be a row vector, CNNs can work directly on image data e.g. a training sample in matrix format of dimensions $w \times h \times c$ where $w$ is the width, $h$ is the height, and $c$ is the number of channels, 1 for greyscale images and 3 for colour (RGB) images. The features used in this work therefore require no transformation to work with CNN since e.g. the MFCC representation of a signal can be interpreted as a greyscale image of dimensions $p \times n \times 1$ where $p$ is the number of frames and $n$ is the number of coefficients (including $\Delta$ and $\Delta\Delta$). For the fixed sequence length described in

Section 3.3, this equates to an input of dimensions $28 \times 39 \times 1$ for the MFCC representation.

The key hyperparameters regarding NNs relate to model architecture, such as number of hidden layers and activation function. CNNs have further hyperparameters to tune, such as kernel size and stride, padding type, and pooling size. Optimising hyperparameters for CNNs can take a considerable amount of time and computational resources, so we look to the literature to best inform our model architecture. The following architectures are shown to have promising results and/or are related to the problem at hand. They provide the motivation for the CNN architectures tested in this work.

1. Ruff et al. [60] proposed a CNN, for their birdsong classification research with 7 classes. Their model contained 6 layers, of which 4 were convolutional. They used the rectified linear unit (ReLU) as an activation function. To avoid overfitting, they used dropout layers [66] with $p = 0.2$ for convolutional layers and $p = 0.5$ for the fully connected layers.

2. Kahl et al. [32] proposed a CNN for their work focused on a very large number of classes (1500). Although the difference in numbers of classes between their work and this work is significant, the concepts should apply to binary classification problems too. In their work they tested 3 models with varying numbers of hidden layers and kernel sizes. Their model number 3, comprising 8 layers of which 5 were convolutional, was shown to have superior results for smaller numbers of classes, indeed they mention that shallow models perform better on small class selections. All of their proposed models use batch normalisation [31], exponential linear units (ELU) [14] as the activation function, and He-initialization [25] for their weighted layers. Dropout layers with $p = 0.5$ were added to each weighted layer before the final dense layer. An initial learning rate of 0.1 is set, which decreases linearly after each epoch to a final value of 0.0001.

To test against a baseline, a highly simplistic CNN architecture will be implemented and tested against these more complex designs. The proposed CNN architectures are listed in Table 3.4.

### RNN

Unlike CNNs, RNNs can work with variable sized inputs. This makes them ideal for working with sequential data such as videos (consisting of sequential frames) and potentially birdsong (consisting of sequential syllables). Since we don't need fixed length inputs

| Model 0 | Model 1 (Ruff [60]) | Model 2 (Kahl [32]) |
|---|---|---|
| 2 weighted layers | 6 weighted layers | 8 weighted layers |
| ReLU activation | ReLU activation | ELU activation |
| Glorot initialization | Glorot initialization | He initialization |
| 32.5k parameters | 3.3m parameters | 23.6m parameters |
| **Conv1,** $20 \times 5 \times 5$ | **Conv1,** $32 \times 3 \times 3$<br>MaxPooling, Size 2<br>Dropout, $p = 0.2$ | **Conv1,** $32 \times 7 \times 7$**, Stride 2**<br>MaxPooling, Size 2 |
| **DenseLayer, 2 Units**<br>Softmax Output | **Conv2,** $32 \times 3 \times 3$<br>MaxPooling, Size 2<br>Dropout, $p = 0.2$ | **Conv2,** $128 \times 5 \times 5$**, Stride 1**<br>MaxPooling, Size 2 |
| | **Conv3,** $64 \times 3 \times 3$<br>MaxPooling, Size 2<br>Dropout, $p = 0.2$ | **Conv3,** $256 \times 3 \times 3$**, Stride 1**<br>MaxPooling, Size 2 |
| | **Conv4,** $64 \times 3 \times 3$<br>MaxPooling, Size 2<br>Dropout, $p = 0.2$ | **Conv4,** $512 \times 3 \times 3$**, Stride 1**<br>MaxPooling, Size 2 |
| | **DenseLayer, 64 Units**<br>Dropout, $p = 0.5$ | **Conv4,** $512 \times 3 \times 3$**, Stride 1**<br>MaxPooling, Size 2 |
| | **DenseLayer, 2 Units**<br>Softmax output | **DenseLayer, 512 Units**<br>Dropout, $p = 0.5$ |
| | | **DenseLayer, 512 Units**<br>Dropout, $p = 0.5$ |
| | | **DenseLayer, 2 Units**<br>Softmax output |

Table 3.4: Candidate CNN architectures. All models use batch normalisation after each weighted layer and all convolutional layers use shape preserving padding. All models use the 'Adam' optimiser with an $l_2$ regularisation of 0.0001.

as training samples, we can utilise a simplified form of the sequence generation method described in Section 3.3.2 for fixed length sequences (FLS) by simply concatenating $n$ syllables together to form a variable length sequence (VLS). We select $n$ by choosing a value such that a VLS has a 50% chance of being shorter than the FLS. Figure 3.10 shows a histogram of the sample lengths for the VLS for $n = 5$. The proportion of VLS with a length less than 14,000 is 0.5044, i.e. the probability that a VLS is shorter than the FLS is 50%, as desired.

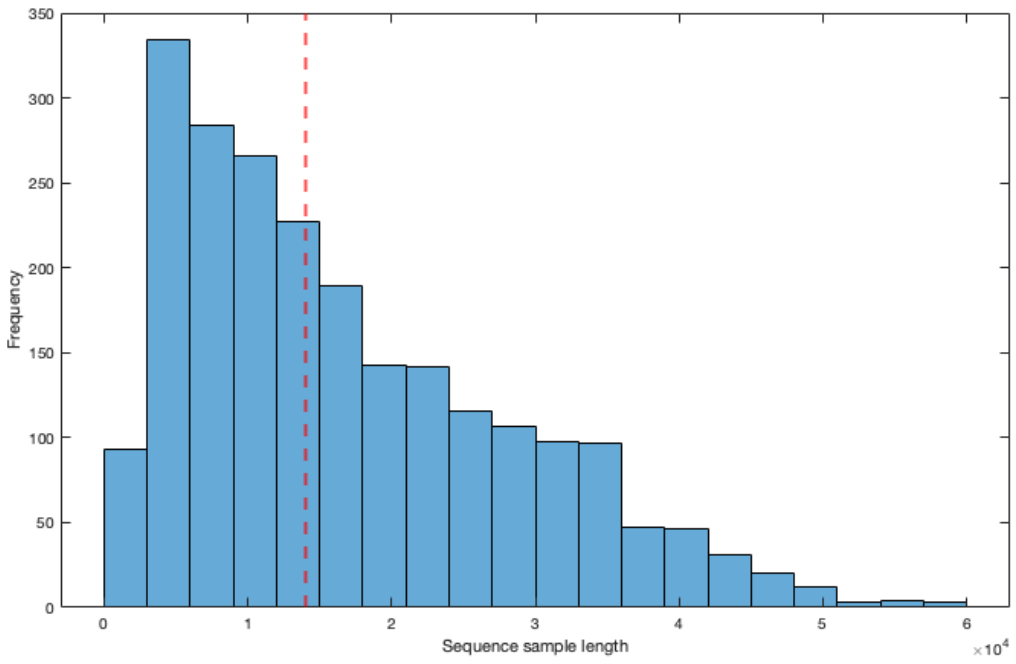



Figure 3.10: Histogram showing the frequency of different sample lengths for VLS with $n = 5$. The red dashed line shows the length of the FLS.

With this form of sequence generation, the number of training samples for each class is shown in Table 3.5. Note that these samples are only used for RNN.

| Class | Individuals | Sequences |
|---|---|---|
| Common blackbird (*TURMER*) | 28 | 1062 |
| Common nightingale (*LUSMEG*) | 28 | 1200 |

Table 3.5: Number of training sequences used for both classes for VLS used for RNN.

MATLAB can receive sequential data in different input formats, such as vector sequences and 2D image sequences. To trigger a sequence input, the first layer of the network must be a `sequenceInputLayer`. The layer accepts an `inputSize` argument, which determines the sequence format. For a vector sequence, `inputSize` is a scalar corresponding to the number of features. For 2D image sequences, `inputSize` is a vector of three elements $\begin{bmatrix} h & w & c \end{bmatrix}$, where $h$ is the image height, $w$ is the image width, and $c$ is the number of channels of the image. The formulation of the VLS lends itself well to the vector sequence input format. To illustrate this, we will concern ourselves only with MFCC, but the concepts can easily be applied to other representations in the frequency domain where each element can be thought of as a T-F unit, such as GTCC and MRCG.

The MFCC for an entire sequence is first extracted. Supposing we extract 13 MFCC coefficients (ignoring $\Delta$ and $\Delta\Delta$ values for now), then the entire sequence can then be thought of as a sequence of 13 dimensional vectors, evolving through time. This can be visualised in Figure 3.11. An alternative way to visualise the MFCC as sequential vectors is presented in Figure 3.12.

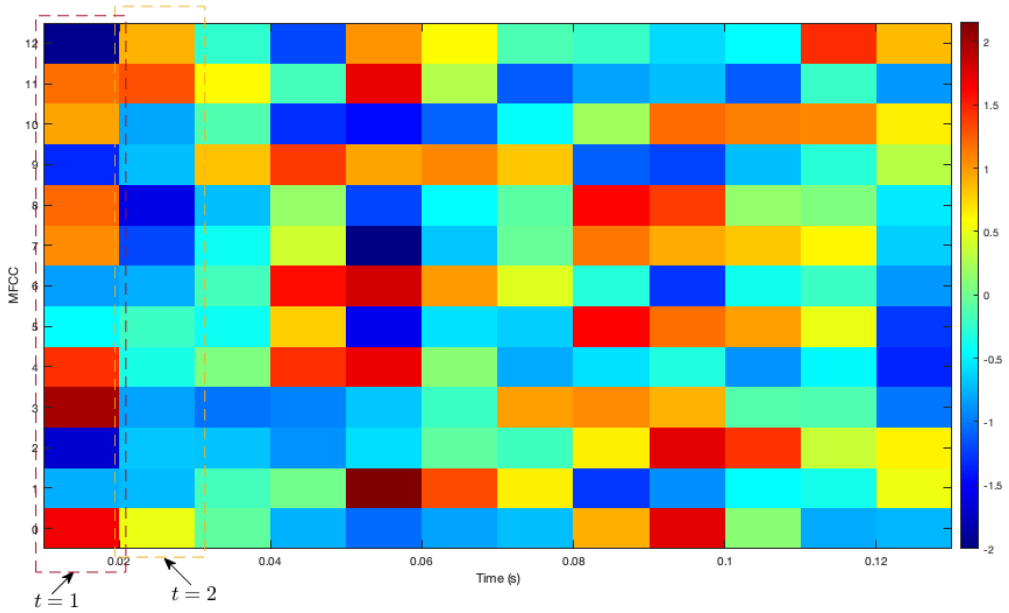



Figure 3.11: MFCC representation of a VLS showing sequential vectors. The $13 \times 12$ matrix can be thought of as a sequence of length 12 of 13 dimensional vectors. The first two vectors are shown in the figure using dashed lines.

This approach has the benefit that the training cell needs little manipulation to be compatible with MATLAB's `trainNetwork` function used to train a NN. The function

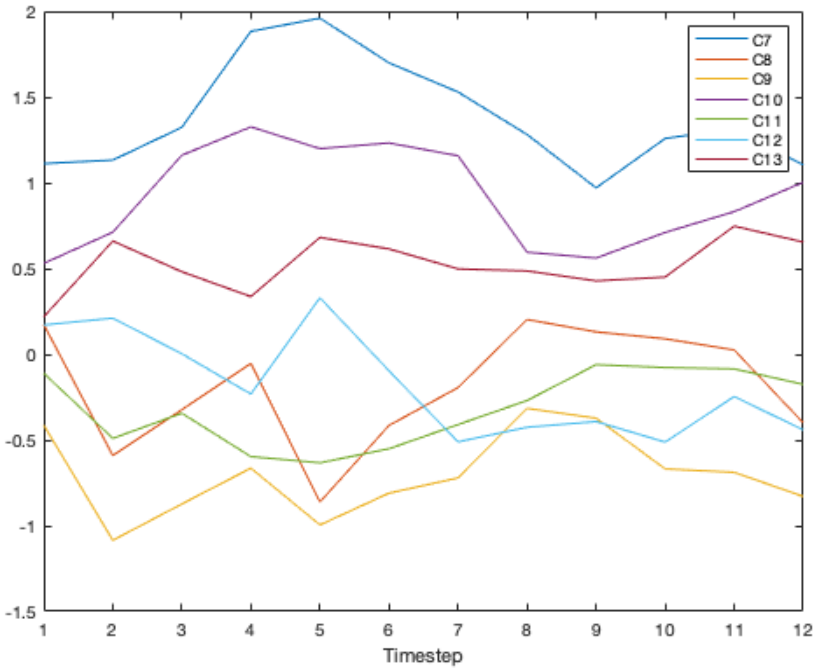



Figure 3.12: Plot of the last 6 MFCC coefficients of a VLS changing with each timestep.

can accept a training cell array where each cell is a $p \times s$ numerical matrix, where $s$ is the number of timesteps in the sequence, and $p$ is the number of features. In the example shown in Figure 3.11, $s = 12$ and $p = 13$. However, this method will treat the dimensions as independent of each other which will mean that the network isn't able to learn cross dimensional features. In other words, only 1 dimensional convolutions may be applied. This may mean that the network misses out on learning from potentially useful patterns in the signal.

An alternative formulation would be to consider sequences of 2D images. Here, the MFCC representation is extracted for each of $n$ syllables in a sequence separately. They are then stacked together to form a sequence of $n\times$ 2D 'images' (MFCC representations). This sequence can then be fed as input to the network. For this approach, the network expects a training input as a $h \times w \times c \times s$ numerical array where $h$, $w$, and $c$ correspond to the height, width, and number of channels of the images, respectively, and $s$ is the sequence length. Here, $s = n$, $c = 1$ since the images can be thought of as greyscale and $h$ will be the number of MFCC coefficients. The big drawback of this method is that $w$ will need to be consistent across all images in the training input. In other words, all syllables will

need to be the same length. This can be achieved with rudimentary processes like padding or looping a syllable until it is of a desired length, but to achieve less noisy results it's likely that this would require more advanced techniques like dynamic time warping. The large scope of this approach means that it will be left to be tackled in future work. A big benefit of this approach is that the network will be able to learn cross dimensional features by applying 2D convolutions. This would complement the temporal features learned by recurrent layers of the network well.

#### Bidirectionality

If the full sequences are known at prediction time, as is the case in this work, then the network can benefit from employing bidirectional RNN layers. This means that information can flow backward and forward through the network, meaning that predictions at a certain timestep can be influenced by future predictions. At the time of writing this work, bidirectional LSTM layers exist in MATLAB (`bilstmLayer`), but bidirectional GRU layers do not.

To test RNN architectures for sequential vectors, we propose 3 models. Given that the literature is reasonably thin in terms of implementing RNN for birdsong classification problems, we propose 3 simple models to establish a baseline. The models can be seen in Table 3.6.

One thing to note with RNN in MATLAB is that although the sequences can be of varying length, the lengths of each sequence in each mini-batch provided to the training routine need to be the same length. Padding is automatically added by the software to each sequence in a mini-batch so that they are all of the same length, usually by padding sequences so that they are the same length as the longest sequence in the mini-batch. To that end, it's recommended to sort the sequences by length before passing them to the training routine and ensuring that the software doesn't shuffle the samples. This ensures that the amount of padding is kept to a minimum which is preferable since padding introduces noise to the network. A mini-batch size of 100 was used for all RNN models.

## 3.6 Evaluating models

After the recordings from each class reserved for testing have undergone pre-processing and segmentation, the segmented syllables undergo the same routine described in Section 3.3.2 to generate the testing sequences. The number of testing sequences used for both classes

| Model 1 | Model 2 | Model 3 |
|---|---|---|
| 2 weighted layers | 4 weighted layers | 4 weighted layers |
| 112.4k parameters | 142.5k parameters | 59.8k parameters |
| **BiLSTM, 100 Units** | **Conv1**, $32 \times 3$ | **Conv1**, $32 \times 3$ |
| Dropout, $p = 0.2$ | MaxPooling, Size 2 | MaxPooling, Size 2 |
| | Dropout, $p = 0.2$ | Dropout, $p = 0.2$ |
| **DenseLayer, 2 Units** | **Conv2**, $64 \times 3$ | **Conv2**, $64 \times 3$ |
| Softmax Output | Global Average Pooling | Global Average Pooling |
| | Dropout, $p = 0.2$ | Dropout, $p = 0.2$ |
| | **BiLSTM, 100 Units** | **GRU, 100 Units** |
| | Dropout, $p = 0.5$ | Dropout, $p = 0.5$ |
| | **DenseLayer, 2 Units** | **DenseLayer, 2 Units** |
| | Softmax Output | Softmax Output |

Table 3.6: Candidate RNN and CRNN architectures for sequential vector sequences. All models use batch normalisation after each weighted layer and all convolutional layers use shape preserving padding. All models use the 'Adam' optimiser with an $l_2$ regularisation of 0.0001, with Glorot initialization and ReLU activation.

can be seen in Table 3.7.

| Class | Individuals | Sequences |
|---|---|---|
| Common blackbird (*TURMER*) | 9 | 360 |
| Common nightingale (*LUSMEG*) | 10 | 400 |

Table 3.7: Number of testing sequences used for both classes.

A testing cell is created in the same way as the training cell and is used for evaluating all models in this thesis.

### 3.6.1 AUC

As mentioned in Section 2.1, the AUC metric is commonly used for binary classification problems and has been frequently used as a metric in birdsong classification problems. Further to the advantages listed previously, the AUC benefits from being unaffected from class imbalances. Although the positive and negative classes are only slightly imbalanced

in this work (see Table 3.3), it makes sense to use a metric that is unaffected by class imbalance all the same.

The AUC is calculated using the `perfcurve` function which checks the class prediction scores returned by the model against the true labels.

### 3.6.2 Classification accuracy

Since the classes are only slightly imbalanced, the classification accuracy provides a reasonably unbiased, quick, and easily interpretable snapshot as to how well a model has performed. It is calculated as

$$A = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[\bar{y}_i = y_i] \tag{3.3}$$

where $N$ is the number of test samples, $\bar{y}$ is a predicted label, $y$ is the groundtruth label, and $\mathbb{I}$ is the indicator function.

## 3.7 Chapter summary

In this chapter we started by describing ways in which audio recordings to be used for experiments in this thesis were collected and pre-processed. We described in detail the steps involved in segmenting the processed audio to be used as training inputs. The following section lists the experiments that are performed in this work related to feature extraction. Lastly, we explore the different models used herein and discuss how the models will be evaluated.

# Chapter 4

# Results and discussion

In this chapter we first lay out the results obtained in the experiments across all 3 hypotheses, including some quick takeaways from the results. We then compare the results across all experiments and explore some deeper discussion points. We conclude the chapter by talking about future work and extensions to the current methodology.

## 4.1 Hypothesis 1

Hypothesis 1 (H1) is stated as follows:

> Explore some of the hyperparameters available during the feature extraction process. The hypothesis is that using hyperparameter values more suited to birdsong classification problems will yield a higher classification accuracy.

H1 was applied to two feature extraction processes that appear in some form widely in the literature related to birdsong classification: MFCC and GTCC.

### 4.1.1 MFCC

The key argument under test in this work as mentioned in Section 3.4.1 is the `BandEdges` argument. The results for the AUC and classification accuracy for the 7 experiments listed in Table 3.1 for MFCC can be seen in Table 4.1 and can be visualised in Figure 4.1.

As shown in the table and chart, both evaluation metrics are around their maximum in the control experiment (Experiment 1). Experiment 5 returns comparable metrics to the control experiment, with slight improvements made to the classification accuracy for the

| | AUC | | Accuracy | |
|---|---|---|---|---|
| Experiment | Linear | RBF | Linear | RBF |
| 1 | 0.887 | 0.958 | 0.825 | 0.872 |
| 2 | 0.879 | 0.830 | 0.828 | 0.801 |
| 3 | 0.868 | 0.860 | 0.802 | 0.789 |
| 4 | 0.845 | 0.836 | 0.765 | 0.785 |
| 5 | 0.873 | 0.953 | 0.813 | 0.880 |
| 6 | 0.863 | 0.929 | 0.809 | 0.878 |
| 7 | 0.879 | 0.905 | 0.827 | 0.823 |

Table 4.1: The AUC and classification accuracy scores for linear and RBF models for different experiments for MFCC. The highest evaluation scores for each metric and for each model are highlighted in grey.

RBF model, but a slight reduction in the AUC. The RBF model shows a clear reduction in performance for wider frequency ranges across both evaluation metrics.

### 4.1.2 GTCC

The key argument under test in this work as mentioned in Section 3.4.1 is the `FrequencyRange` argument. The results for the AUC and classification accuracy for GTCC can be seen in Table 4.2 and can be visualised in Figure 4.2.

| | AUC | | Accuracy | |
|---|---|---|---|---|
| Experiment | Linear | RBF | Linear | RBF |
| 1 | 0.884 | 0.895 | 0.844 | 0.802 |
| 2 | 0.887 | 0.946 | 0.831 | 0.856 |
| 3 | 0.891 | 0.933 | 0.852 | 0.850 |
| 4 | 0.883 | 0.948 | 0.846 | 0.859 |

Table 4.2: The AUC and classification accuracy scores for linear and RBF models for different experiments for GTCC. The highest evaluation scores for each metric and for each model are highlighted in grey.

As shown in the table and chart, there is a significant increase in both metrics for the RBF model when the frequency range is narrowed to be closer to the human and birdsong vocalisation range. The evaluation metrics for the linear model remain largely similar across different frequency ranges.

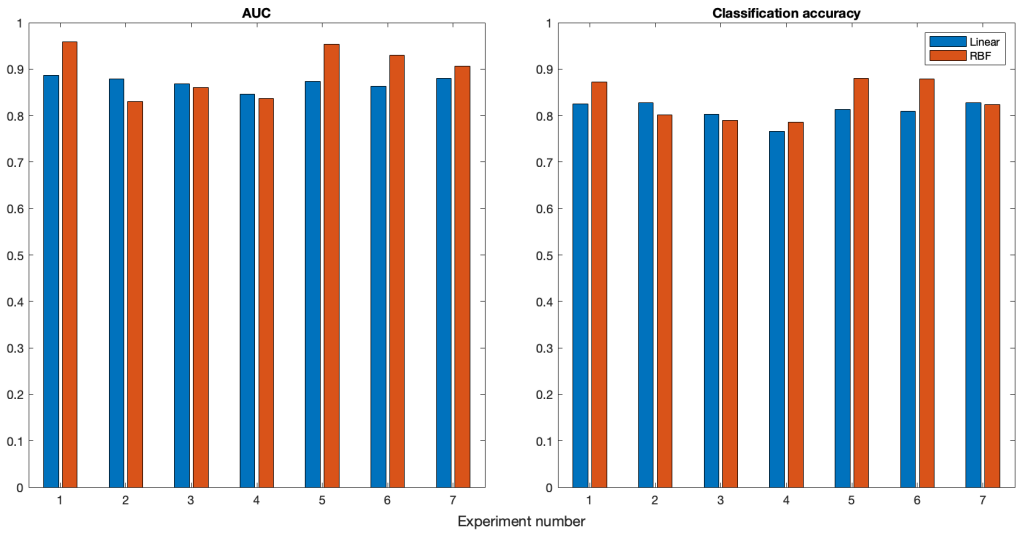



Figure 4.1: The AUC and classification accuracy scores for linear and RBF models for different experiments for MFCC.

## 4.2 Hypothesis 2

Hypothesis 2 (H2) is stated as follows:

> Explore the performance of deep learning architectures such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) and compare the results with simpler statistical models. The hypothesis is that more complex and flexible architectures such as these will have superior performance when compared to simpler statistical models, such as support vector machines (SVM).

For all experiments related to H2 we use the highest performing feature representations from H1: an MFCC representation with the `BandEdges` argument set to the range from Experiment 1 and GTCC representation with the `FrequencyRange` set to the range from Experiment 4.

### 4.2.1 CNN

The CNN architectures listed in Table 3.4 were tested and their performance evaluated. The following section details the results.

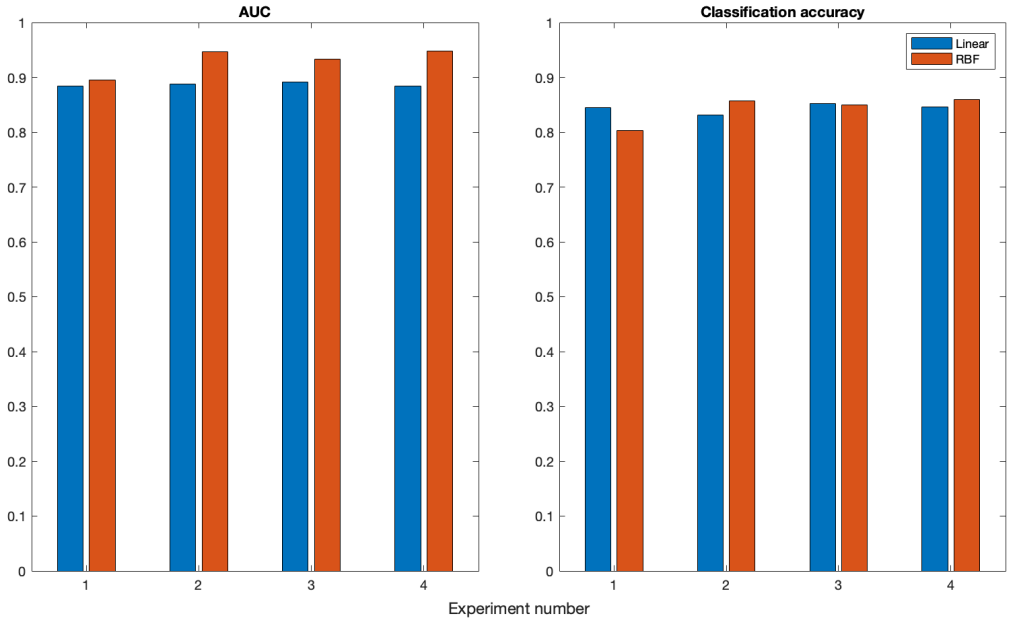



Figure 4.2: The AUC and classification accuracy scores for linear and RBF models for different experiments for GTCC.

### Results

The results for the AUC and classification accuracy for a CNN trained on a MFCC and GTCC representation of training sequences can be seen in Table 4.3 and can be visualised in Figure 4.3.

| | AUC | | Accuracy | |
|---|---|---|---|---|
| Model | MFCC | GTCC | MFCC | GTCC |
| 0 | 0.976 | 0.976 | 0.912 | 0.900 |
| 1 | 0.971 | 0.952 | 0.865 | 0.882 |
| 2 | 0.988 | 0.984 | 0.929 | 0.928 |

Table 4.3: The AUC and classification accuracy for different CNN architectures trained on MFCC and GTCC feature representations. The highest evaluation scores for each metric are highlighted in grey.

We can see that the CNN models all deliver very promising results, especially models 0 and 2. The highest performing of the CNN models is model 2, but it should be noted that it takes significantly longer to train model 2 compared to model 0 and its evaluation

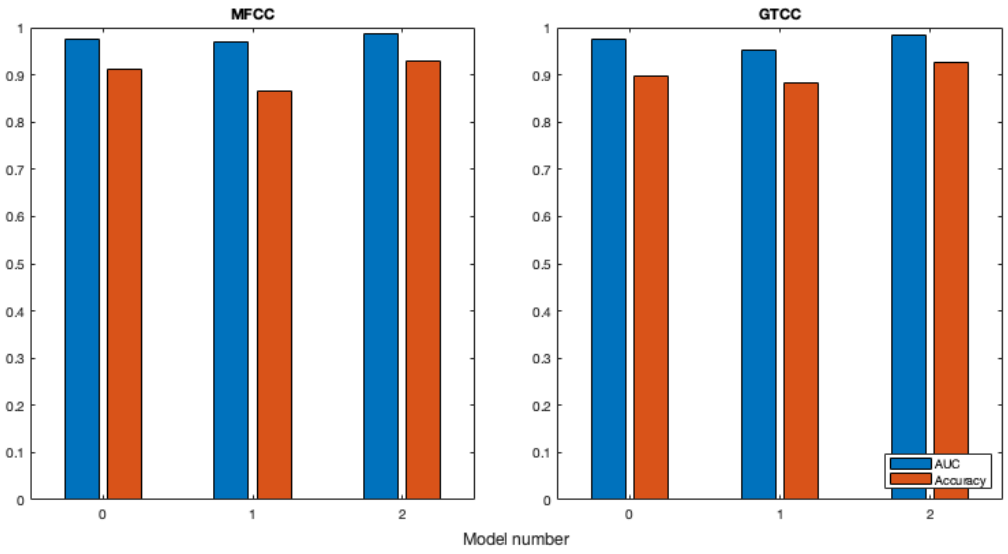



Figure 4.3: The AUC and classification accuracy for different CNN architectures trained on MFCC and GTCC feature representations.

metrics are only slightly better.

### 4.2.2 RNN & CRNN

The RNN and CRNN architectures listed in Table 3.6 were tested and their performance evaluated. The following section details the results.

#### Results

The results for the AUC and classification accuracy for a RNN and CRNN trained on a MFCC representation of training sequences can be seen in Table 4.4 and can be visualised in Figure 4.4.

From the results we can see that model 1 performs the best across both metrics. It should be noted though that model 1 takes the longest to train, about 5 times longer per epoch compared to models 2 and 3. We can also see a clear drop in performance comparing model 3 (a GRU model) to model 2 (a bidirectional LSTM model). This may be from the fact that the GRU layer cannot learn in both directions, or it may be that the decrease in parameters that the GRU offers comes at the cost of limited capacity to learn.

| Model | AUC | Accuracy |
|---|---|---|
| 1 | 0.957 | 0.892 |
| 2 | 0.912 | 0.860 |
| 3 | 0.887 | 0.840 |

Table 4.4: The AUC and classification accuracy for different RNN and CRNN architectures trained on MFCC. The highest evaluation scores for each metric and are highlighted in grey.

## 4.3 Hypothesis 3

Hypothesis 3 (H3) is stated as follows:

> Explore the birdsong classification performance of feature representations shown to have promising results for non-birdsong related audio classification problems. The hypothesis is that feature representations shown to have good results in audio classification problems will have good results for birdsong identification.

H3 was applied to a relatively novel feature representation, the MRCG, using CNN, RNN and CRNN. To the author's knowledge, this experiment has not been performed before with regards to birdsong classification.

Based on the results from the GTCC experiments shown in Table 3.2, we can see that a gammatone frequency bank range of about 50 — 10000 Hz works well for this problem. This range is used in calculating the MRCG representation for all experiments.

Due to the high dimensionality of the MRCG representation and considering the limited computational resources of the machine upon which training is performed, we select simple NN models to evaluate the MRCG feature. The models chosen here are Model 0 from Table 3.4, and Models 1 and 2 from Table 3.6.

#### Results

The AUC and accuracy results for NN models trained on MRCG representation are shown in Table 4.5.

The CNN model performs well but the RNN based models perform very poorly. The training curve for the RNN models was still converging by the time that the maximum number of epochs (100) was reached so it's possible that the model may just need a longer

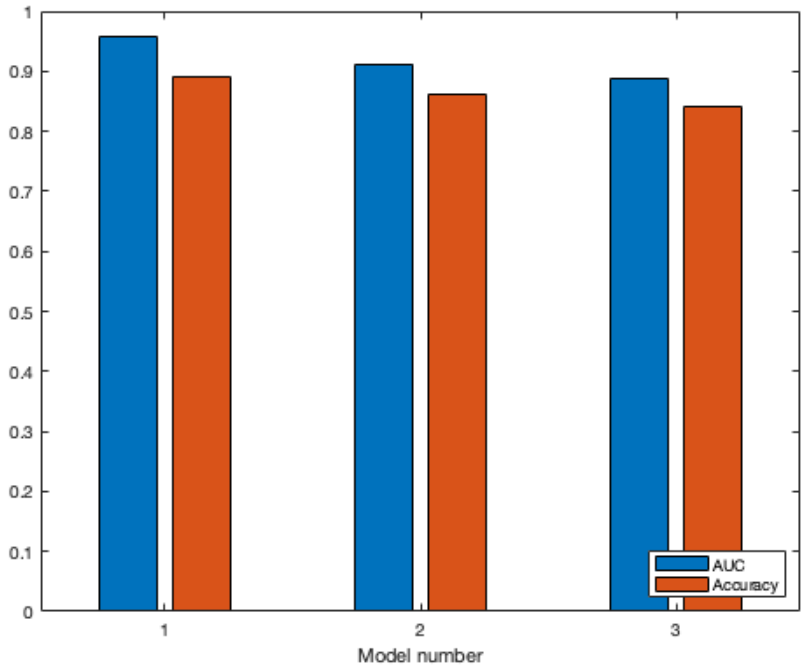



Figure 4.4: The AUC and classification accuracy for different RNN and CRNN architectures trained on MFCC.

time to train. However, during training the validation curve fluctuated around 50% and so it's also possible that the sequential vector representation is inappropriate for MRCG and so RNN with this type of sequence input for MRCG will never post good results. Since the CNN results are much better, it may be worthwhile exploring more complicated CNN model architectures with MRCG. Due to the high dimensionality of MRCG I was not able to perform training with more complicated models on my machine.

## 4.4 Comparison of models

In this section we collect the results from all experiments and compare them to see the highest performing models. This may help to identify patterns or trends in model structure or feature representation which boost birdsong binary classification performance.

We start by listing all models used in this work. The SVM models used in this work are listed in Table 4.6 and the NN models are listed in Table 4.7.

A combined table of evaluation metrics can be seen in Table 4.8 and visualised in Figure 4.5.

| Model | AUC | Accuracy |
|---|---|---|
| 1 | 0.968 | 0.889 |
| 2 | 0.188 | 0.283 |
| 3 | 0.138 | 0.205 |

Table 4.5: The AUC and classification accuracy for different RNN and CRNN architectures trained on MFCC. The highest evaluation scores for each metric and are highlighted in grey.

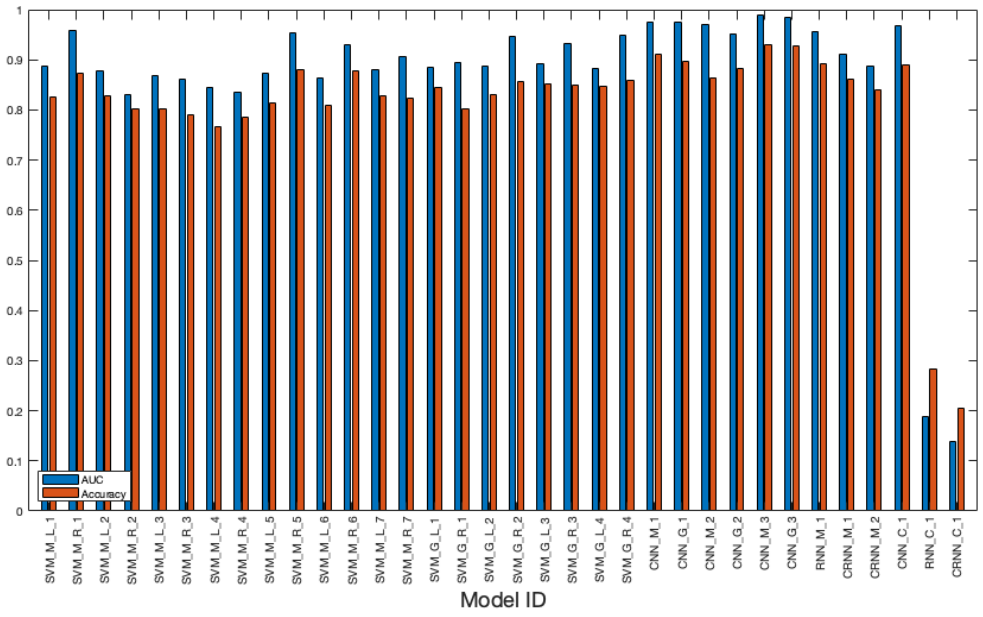



Figure 4.5: Figure showing the AUC and accuracy for all models tested in this thesis.

| Model ID | Feature | Description |
|---|---|---|
| SVM_M_L_1 | MFCC | Linear SVM with 40 mel bands, default range |
| SVM_M_R_1 | MFCC | As above, but RBF SVM |
| SVM_M_L_2 | MFCC | Linear SVM with 40 mel bands, extended range |
| SVM_M_R_2 | MFCC | As above, but RBF SVM |
| SVM_M_L_3 | MFCC | Linear SVM with 80 mel bands, extended range |
| SVM_M_R_3 | MFCC | As above, but RBF SVM |
| SVM_M_L_4 | MFCC | Linear SVM with 40 linear bands, extended range |
| SVM_M_R_4 | MFCC | As above, but RBF SVM |
| SVM_M_L_5 | MFCC | Linear SVM with 40 linear bands, default range |
| SVM_M_R_5 | MFCC | As above, but RBF SVM |
| SVM_M_L_6 | MFCC | Linear SVM with 40 anti-mel bands, default range |
| SVM_M_R_6 | MFCC | As above, but RBF SVM |
| SVM_M_L_7 | MFCC | Linear SVM with 40 mel bands, slightly extended range |
| SVM_M_R_7 | MFCC | As above, but RBF SVM |
| SVM_G_L_1 | GTCC | Linear SVM with default (widest) gammatone range |
| SVM_G_R_1 | GTCC | As above, but RBF SVM |
| SVM_G_L_2 | GTCC | Linear SVM with narrowed gammatone range |
| SVM_G_R_2 | GTCC | As above, but RBF SVM |
| SVM_G_L_3 | GTCC | Linear SVM with less narrow gammatone range |
| SVM_G_R_3 | GTCC | As above, but RBF SVM |
| SVM_G_L_4 | GTCC | Linear SVM with more narrow gammatone range |
| SVM_G_R_4 | GTCC | As above, but RBF SVM |

Table 4.6: Table listing the different SVM models used in this work. More details for the MFCC models can be seen in Table 3.1 and Table 3.2 for GTCC models.

| Model ID | Feature | Description | # Learnable parameters | Training time/epoch (s) |
|---|---|---|---|---|
| CNN_M_1 | MFCC | Simple 2 layer CNN | 32.5k | 3 |
| CNN_G_1 | GTCC | As above | 32.5k | 3 |
| CNN_M_2 | MFCC | 6 layer CNN based on [60] | 3.3m | 60 |
| CNN_G_2 | GTCC | As above | 3.3m | 60 |
| CNN_M_3 | MFCC | 8 layer CNN based on [32] | 23.6m | 165 |
| CNN_G_3 | GTCC | As above | 23.6m | 165 |
| RNN_M_1 | MFCC | Simple 2 layer RNN | 112.4k | 16 |
| CRNN_M_1 | MFCC | 4 layer CRNN with BiLSTM layer | 142.5k | 3 |
| CRNN_M_2 | MFCC | 4 layer CRNN with GRU layer | 59.8k | 3 |
| CNN_C_1 | MRCG | Simple 2 layer CNN | 795.1k | 3 |
| RNN_C_1 | MRCG | Simple 2 layer RNN | 695.6k | 30 |
| CRNN_C_1 | MRCG | 4 layer CRNN with BiLSTM layer | 212.5k | 9 |

Table 4.7: Table listing the different NN models used in this work. More details for the CNN models can be seen in Table 3.4 and Table 3.6 for RNN and CRNN models.

| Model ID | AUC | Accuracy | Model ID | AUC | Accuracy |
|---|---|---|---|---|---|
| SVM_M_L_1 | 0.887 | 0.825 | SVM_G_R_2 | 0.946 | 0.856 |
| SVM_M_R_1 | 0.958 | 0.872 | SVM_G_L_3 | 0.891 | 0.852 |
| SVM_M_L_2 | 0.879 | 0.828 | SVM_G_R_3 | 0.933 | 0.850 |
| SVM_M_R_2 | 0.830 | 0.801 | SVM_G_L_4 | 0.883 | 0.846 |
| SVM_M_L_3 | 0.868 | 0.802 | SVM_G_R_4 | 0.948 | 0.859 |
| SVM_M_R_3 | 0.860 | 0.789 | CNN_M_1 | 0.976 | 0.912 |
| SVM_M_L_4 | 0.845 | 0.765 | CNN_G_1 | 0.976 | 0.900 |
| SVM_M_R_4 | 0.836 | 0.785 | CNN_M_2 | 0.971 | 0.865 |
| SVM_M_L_5 | 0.873 | 0.813 | CNN_G_2 | 0.952 | 0.882 |
| SVM_M_R_5 | 0.953 | 0.880 | CNN_M_3 | 0.988 | 0.929 |
| SVM_M_L_6 | 0.863 | 0.809 | CNN_G_3 | 0.984 | 0.928 |
| SVM_M_R_6 | 0.929 | 0.878 | RNN_M_1 | 0.957 | 0.892 |
| SVM_M_L_7 | 0.879 | 0.827 | CRNN_M_1 | 0.912 | 0.860 |
| SVM_M_R_7 | 0.905 | 0.823 | CRNN_M_2 | 0.887 | 0.840 |
| SVM_G_L_1 | 0.884 | 0.844 | CNN_C_1 | 0.968 | 0.889 |
| SVM_G_R_1 | 0.895 | 0.802 | RNN_C_1 | 0.188 | 0.283 |
| SVM_G_L_2 | 0.887 | 0.831 | CRNN_C_1 | 0.138 | 0.205 |

Table 4.8: The AUC and classification accuracy for all models. The highest evaluation scores for each metric across all models are highlighted in grey.

## 4.5 Discussion

From the evaluation results obtained across the different experiments, the following discussion points arise:

1. A filterbank frequency range for feature representation that fits tightly around the dominant frequencies produced by birdsong performs better than one that is wider. This is strengthened by the results obtained from the experiments in H1. The initial assumption was that since birds produce higher frequency harmonics above the normal birdsong frequency range ($\backsim$1 kHz — 10 kHz), then having a filterbank range that responds to those harmonics may mean that potentially useful features could be extracted from the harmonics and utilised by the model. In practice this doesn't seem to be the case, with both mel and gammatone filterbanks that range from $\backsim$1 kHz — 10 kHz yielding the highest evaluation metrics. The assumption may have been naive since when comparing birdsong to human speech, as can be seen in Figure 4.6, we can see that both bird and human vocalisations produce strong harmonics. Given that MFCC with the default filterbank spacing has been shown to have excellent results with human speech recognition [48], it stands to reason that the default filter bank should have good results with birdsong too.

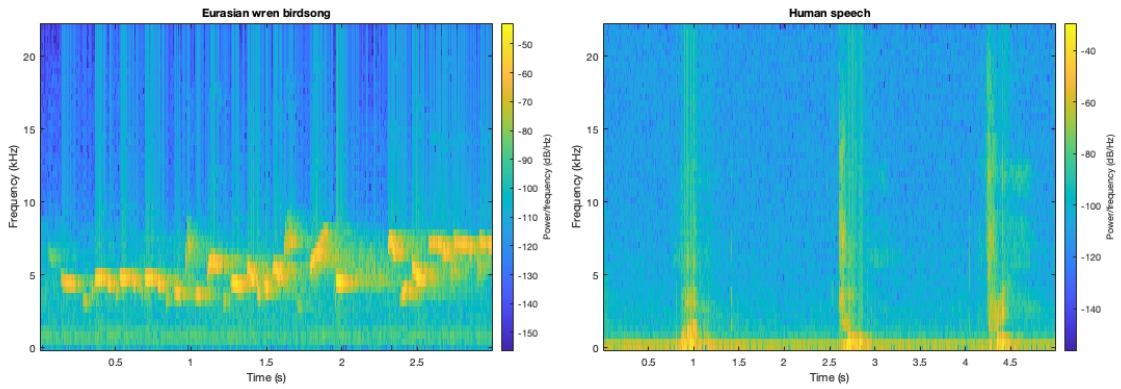



Figure 4.6: Spectrograms of a sample from a Eurasian wren birdsong and human speech.

The other assumption with regards to filter bank range and spacing is that, since birdsong typically resides at higher frequencies than human speech (as can been seen in Figure 4.6), filterbanks that are spaced closer together at higher frequencies (such as anti-mel filter banks), should offer more discriminatory power at higher frequencies and thus yield better results at discriminating birdsong. However, the results from

the experiments performed in this work did not reinforce this claim and it remains unclear why.

2. NN models can offer significantly improved metrics when compared to simpler statistical models such as SVM. Even the lowest performing NN model in this work (CRNN_M_2) trained on MFCC or GTCC posts metrics that are superior to the majority of SVM models tested here. This is reasonably unsurprising since NN models have considerably more learnable parameters and are able to fit more complex decision boundaries. This of course comes at a cost of higher computational demands and training time. While the RNN and CRNN model results are encouraging, it's important to note though that the models implemented here are quite simplistic. With some thorough tuning and testing I'm confident that the RNN models, in particular CRNN models, can achieve better evaluation metrics than plain CNN models. RNN models offer the enticing prospect of making use of the temporal evolution of birdsong and even learning from the gaps in between phrases, which may offer further information which can aid with classification.

3. MRCG feature representations tested in this work have not performed as well as expected. This is disappointing since the literature has shown promising results when MRCG was used as a feature in audio classification problems. One thing to note though is that the literature has often tested MRCG at different SNR levels to determine its performance at classifying audio in noisy situations, where it has often outperformed more traditional feature representations [7, 12]. The samples tested in this work have all been very clean samples with little or no background noise, so it's possible that MRCG could start to become more effective when different SNR levels are tested with birdsong classification.

## 4.6 Further work and extensions of methodology

The work carried out in this thesis is merely scratching the surface of birdsong binary classification problems. There is enormous scope for extending the methodologies used here in pre-processing, training, and classification in order to improve results and broaden the problem. The following list is certainly not exhaustive but is intended to provide a quick overview of some key techniques and concepts that could be incorporated in future work.

### Pre-processing

Pre-processing is a key stage in the process. A thorough and considered pre-processing pipeline can significantly improve classification accuracy and generalisation performance. An easily accessible way to improve generalisation is through data augmentation. This has been done in many research papers with promising results [32]. MATLAB provides inbuilt ways to augment audio data through the `audioDataAugmenter` pipeline, which can synthetically enlarge datasets through techniques like pitch shifting, time-scale modification, and volume control. It would be interesting to apply these augmentations and see how it affects the performance of these models. The pipeline also offers noise addition. In practice, working with clean recordings of birdsong would be very unlikely since the recordings would likely be corrupted with all manner of background noise, such as urban noise, rainforest noise, or ambient noise e.g. wind, and so adding random noise through the pipeline could be useful. It might also be informative to manually add noise at precise SNR levels and evaluate the performance of the different feature representations and models in this work.

To help with managing training and testing audio data, it might be worthwhile making use of MATLAB's `audioDatastore`. The function needs one argument: a path to the location of the audio data to be used as training and testing samples. It allows for labels to be read from directory names which would mean not having to manage a separate structure of labels for the samples. It also helps with reducing memory load since samples from the collection are referenced by their location on disk rather than loading all samples into memory. Furthermore, the datastore object comes with builtin functions to split the store into a training and validation set and perform transformations on the audio data, amongst other things.

Another interesting extension of the methodology would be to consider a birdsong segment as a variable length sequence of images as mentioned in Section 3.5.2 where each syllable is transformed to some image representation. This would allow for a system whereby syllables extracted from birdsong could be fed into a RNN or CRNN with 2D convolutions. This could then be used for realtime classification of birdsong, as is done in the excellent Merlin[1] app developed by Cornell University Lab of Ornithology.

Lastly, a potential simplification of the pipeline could be to make more use of the `detectSpeech` function. This is used in this work to strip leading and trailing bits of background noise from recordings. With some experimentation of the arguments it could

[1] `https://merlin.allaboutbirds.org/`

potentially be used to perform the syllable segmentation of the recording too.

#### Feature extraction & training

An important concept missing in this thesis is combining features into stacks. Instead of looking at feature representations in isolation, combinations of features can be created and the model can be left to determine where the important patterns lie in the dimensions of the data. This has been done in many research papers [73, 57, 65] to great effect. Chen et al. [12] conducted thorough research into a large number of feature representations in their work on speech separation. This could be used as a basis for feature stacking extension to this work.

The RNN and CRNN models implemented in this work are fairly naive and were implemented to generate a baseline of the RNN method. With some experimenting of the hyperparameters available, such as kernel size, pooling size and number of convolutional layers, they could be shown to have improved performance.

#### Classification

Binary classification is a highly simplified version of the problem. In reality, birdsong classification systems would likely need to classify birdsong into hundreds or even thousands of classes. This could take the form of one sample being assigned to one of many classes, or one sample which contains vocalisations from different birds being able to accurately identify all the classes contained therein. The former can be tackled using SVMs by turning the problem into a one-versus-all classification, as mentioned in Section 2.3.1. For NN models the modification is more simple. The final dense layer can be modified to have the same number of output neurons as classes and the softmax output layer will take care of outputting a distribution over the classes. Identifying multiple birds within one sample poses more challenges but there has already been some interesting research conducted into it using encoder-decoder architectures [51]. Indeed, in their paper they mention that future work includes adding recurrent layers to their network in order to take into account the temporal relation between bird syllables.

## 4.7 Chapter summary

In this chapter we started displaying the evaluation metrics obtained for the three main hypotheses of this work. We then compare the metrics across all hypotheses to gain a

high level view of how the hypotheses relate to each other. We conclude the chapter by exploring the discussion points that arise from the results and examine extensions of the methodology used here that could be incorporated into future work.

# Chapter 5

# Conclusion

The main contribution of this study is the investigation of important hyperparameters used to extract classic features like MFCC and GTCC, and the application of MRCG to birdsong classification. We have shown that the values selected for hyperparameters can have a significant effect on the overall classification accuracy and that the hyperparameter values should be carefully considered for the problem at hand. In terms of birdsong, we have shown that designing filterbanks that span a range of $\backsim$1 kHz — 10 kHz to be used for MFCC and GTCC extraction yields higher model performance. During this investigation we have shown that MFCC usually outperforms GTCC when the sample recordings have minimal background noise.

While the MRCG results achieved here were less than desirable, it's unclear whether this is from the model design or the suitability of MRCG with birdsong as some concessions had to be made to allow for MRCG's significantly higher dimensionality compared to classic features such as MFCC. This could be verified by testing MRCG with some more complex CNN or CRNN model architectures on more powerful machines.

We have also investigated some more flexible and modern architectures such as CNN and RNN and have confirmed that they can easily outperform more classical models like SVM. In doing so we have also proposed a novel method of generating training and testing samples by combining syllables, the elemental blocks of birdsong, into sequences. We have also laid the theoretical groundwork for an alternative form of sequence generation that can be used with CRNN in future work.

# Bibliography


[1] Miguel A Acevedo, Carlos J Corrada-Bravo, Héctor Corrada-Bravo, Luis J Villanueva-Rivera, and T Mitchell Aide. Automated classification of bird and amphibian calls using machine learning: A comparison of methods. *Ecological Informatics*, 4(4):206–214, 2009.

[2] Sharath Adavanne, Konstantinos Drossos, Emre Çakir, and Tuomas Virtanen. Stacked convolutional and recurrent neural networks for bird audio detection. In *2017 25th European signal processing conference (EUSIPCO)*, pages 1729–1733. IEEE, 2017.

[3] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. The conversation: Deep audio-visual speech enhancement. *arXiv preprint arXiv:1804.04121*, 2018.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[5] Myron Charles Baker and Michael A Cunningham. The biology of bird-song dialects. *Behavioral and Brain Sciences*, 8(1):85–100, 1985.

[6] Franz Berger, William Freillinger, Paul Primus, and Wolfgang Reisinger. Bird audio detection-dcase 2018. *DCASE2018 Challenge, Tech. Rep.*, 2018.

[7] S Binti Abdullah, Andreas Demosthenous, and Ifat Yasin. Comparison of auditory-inspired models using machine-learning for noise classification. In *International Journal of Simulation Systems, Science & Technology Special Issue: Conference Procedings UKSim2020, 25 to 27 March 2020*. United Kingdom Simulation Society, 2020.

[8] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*, 2017.

[9] Richard K Broughton, James M Bullock, Charles George, Ross A Hill, Shelley A Hinsley, Marta Maziarz, Markus Melin, J Owen Mountford, Tim H Sparks, and Richard F Pywell. Long-term woodland restoration on lowland farmland through passive rewilding. *PloS one*, 16(6):e0252466, 2021.

[10] Clive K Catchpole and Peter JB Slater. *Bird song: biological themes and variations*. Cambridge university press, 2003.

[11] Deep Chakraborty, Paawan Mukker, Padmanabhan Rajan, and Aroor Dinesh Dileep. Bird call identification using dynamic kernel based support vector machines and deep neural networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 280–285. IEEE, 2016.

[12] Jitong Chen, Yuxuan Wang, and DeLiang Wang. A feature study for classification-based speech separation at low signal-to-noise ratios. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1993–2002, 2014.

[13] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[14] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[15] Maximilian Crous. Polyphonic bird sound event detection with convolutional recurrent neural networks. *10.13140/RG. 2.2*, 11943, 2019.

[16] Michał Daniluk. Back-to-the-future networks: Referring to the past to predict the future. Master's thesis, University College, London, 2016.

[17] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

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

[19] Simone Disabato, Giuseppe Canonaco, Paul G Flikkema, Manuel Roveri, and Cesare Alippi. Birdsong detection at the edge with deep learning. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 9–16. IEEE, 2021.

[20] Allison J. Doupe and Patricia K. Kuhl. Birdsong and human speech: Common themes and mechanisms. *Annual Review of Neuroscience*, 22(1):567–631, 1999. PMID: 10202549.

[21] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *arXiv preprint arXiv:1804.03619*, 2018.

[22] Seppo Fagerlund. Automatic recognition of bird species by their sounds. *Finlandia: Helsinki University Of Technology*, 2004.

[23] Seppo Fagerlund. Bird species recognition using support vector machines. *EURASIP Journal on Advances in Signal Processing*, 2007:1–8, 2007.

[24] Theodoros Giannakopoulos. A method for silence removal and segmentation of speech signals, implemented in matlab. *University of Athens, Athens*, 2, 2009.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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

[27] Tianxing He and Jasha Droppo. Exploiting lstm structure in deep neural networks for speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5445–5449. IEEE, 2016.

[28] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[32] Stefan Kahl, Thomas Wilhelm-Stein, Hussein Hussein, Holger Klinck, Danny Kowerko, Marc Ritter, and Maximilian Eibl. Large-scale bird sound classification using convolutional neural networks. *CLEF (working notes)*, 1866, 2017.

[33] Juntae Kim and Minsoo Hahn. Voice activity detection using an adaptive context attention model. *IEEE Signal Processing Letters*, 25(8):1181–1185, 2018.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[35] Mineichi Kudo, Jun Toyama, and Masaru Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11):1103–1111, 1999.

[36] Chiman Kwan, KC Ho, Gang Mei, Yunhong Li, Zhubing Ren, Roger Xu, Y Zhang, Debang Lao, M Stevenson, Vincent Stanford, et al. An automated acoustic system to monitor and classify birds. *EURASIP Journal on Advances in Signal Processing*, 2006:1–19, 2006.

[37] Mario Lasseck. Improved automatic bird identification through decision tree based feature selection and bagging. *CLEF (working notes)*, 1391, 2015.

[38] Mario Lasseck. Acoustic bird detection with deep convolutional neural networks. In *DCASE*, pages 143–147, 2018.

[39] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[40] Howard Lei and Eduardo Lopez. Mel, linear, and antimel frequency cepstral coefficients in broad phonetic regions for telephone speaker recognition. In *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[41] Yi Ren Leng and Huy Dat Tran. Multi-label bird classification using an ensemble classifier with simple features. In *Signal and information processing association annual summit and conference (APSIPA), 2014 Asia-Pacific*, pages 1–5. IEEE, 2014.

[42] Dongge Li, Ishwar K Sethi, Nevenka Dimitrova, and Tom McGee. Classification of general audio data for content-based retrieval. *Pattern recognition letters*, 22(5):533–544, 2001.

[43] MS Likitha, Sri Raksha R Gupta, K Hasitha, and A Upendra Raju. Speech based human emotion recognition using mfcc. In *2017 international conference on wireless communications, signal processing and networking (WiSPNET)*, pages 2257–2260. IEEE, 2017.

[44] David Luther and Luis Baptista. Urban noise and the cultural evolution of bird songs. *Proceedings of the Royal Society B: Biological Sciences*, 277(1680):469–473, 2010.

[45] PETER MARLER. Chapter 5 - bird calls: a cornucopia for communication. In Peter Marler and Hans Slabbekoorn, editors, *Nature's Music*, pages 132–177. Academic Press, San Diego, 2004.

[46] Alex L McIlraith and Howard C Card. Birdsong recognition using backpropagation and multivariate statistics. *IEEE Transactions on Signal Processing*, 45(11):2740–2748, 1997.

[47] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[48] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

[49] Rajdeep Mukherjee, Dipyaman Banerjee, Kuntal Dey, and Niloy Ganguly. Convolutional recurrent neural network based bird audio detection. *DCASE challenge*, 2018.

[50] Rafael Hernández Murcia and Vıctor Suárez Paniagua. Bird identification from continuous audio recordings the icml 2013 bird challenge. *Email list of participants*, page 96, 2013.

[51] Revathy Narasimhan, Xiaoli Z Fern, and Raviv Raich. Simultaneous segmentation and classification of bird song using cnn. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 146–150. IEEE, 2017.

[52] Alberto García Arroba Parrilla and Dan Stowell. Polyphonic sound event detection for highly dense birdsong scenes. *arXiv preprint arXiv:2207.06349*, 2022.

[53] Roy D Patterson, KEN Robinson, John Holdsworth, Denis McKeown, C Zhang, and Michael Allerhand. Complex sounds and auditory images. In *Auditory physiology and perception*, pages 429–446. Elsevier, 1992.

[54] Ilyas Potamitis, Stavros Ntalampiras, Olaf Jahn, and Klaus Riede. Automatic bird sound detection in long real-field recordings: Applications and tools. *Applied Acoustics*, 80:1–9, 2014.

[55] Zhongdi Qu, Parisa Haghani, Eugene Weinstein, and Pedro Moreno. Syllable-based acoustic modeling with ctc-smbr-lstm. In *2017 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 173–177. IEEE, 2017.

[56] Murugaiya Ramashini, Pg Emeroylariffion Abas, Kusuma Mohanchandra, and Liyanage C De Silva. Robust cepstral feature for bird sound classification. *International Journal of Electrical and Computer Engineering*, 12(2):1477, 2022.

[57] Murugiaya Ramashini, Pg Emeroylariffion Abas, Ulmar Grafe, and Liyanage C De Silva. Bird sounds classification using linear discriminant analysis. In *2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, pages 1–6. IEEE, 2019.

[58] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[59] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[60] Zachary J Ruff, Damon B Lesmeister, Leila S Duchac, Bharath K Padmaraju, and Christopher M Sullivan. Automated identification of avian vocalizations with deep convolutional neural networks. *Remote Sensing in Ecology and Conservation*, 6(1):79–92, 2020.

[61] DR Sarvamangala and Raghavendra V Kulkarni. Convolutional neural networks in medical image understanding: a survey. *Evolutionary intelligence*, 15(1):1–22, 2022.

[62] Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for lvcsr. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4955–4959. IEEE, 2016.

[63] John Shore and Rodney Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on information theory*, 26(1):26–37, 1980.

[64] Malcolm Slaney. Auditory toolbox. *Interval Research Corporation, Tech. Rep*, 10(1998):1194, 1998.

[65] Panu Somervuo, Aki Harma, and Seppo Fagerlund. Parametric representations of bird sounds for automatic species recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):2252–2263, 2006.

[66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[67] Naoya Takahashi, Michael Gygli, and Luc Van Gool. Aenet: Learning deep audio features for video analysis. *IEEE Transactions on Multimedia*, 20(3):513–524, 2017.

[68] Vlad M Trifa, Alexander NG Kirschel, Charles E Taylor, and Edgar E Vallejo. Automated species recognition of antbirds in a mexican rainforest using hidden markov models. *The Journal of the Acoustical Society of America*, 123(4):2424–2431, 2008.

[69] Xavier Valero and Francesc Alias. Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification. *IEEE transactions on multimedia*, 14(6):1684–1689, 2012.

[70] Willem-Pier Vellinga and Robert Planqué. The xeno-canto collection and its relation to sound recognition and classification. In *CLEF (Working Notes)*, 2015.

[71] Yuxuan Wang, Kun Han, and DeLiang Wang. Exploring monaural features for classification-based speech segregation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(2):270–279, 2012.

[72] Zhong-Qiu Wang and DeLiang Wang. A joint training framework for robust automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):796–806, 2016.

[73] Na Yan, Aibin Chen, Guoxiong Zhou, Zhiqiang Zhang, Xiangyong Liu, Jianwu Wang, Zhihua Liu, and Wenjie Chen. Birdsong classification based on multi-feature fusion. *Multimedia Tools and Applications*, 80:36529–36547, 2021.

[74] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.