University of York

School of Electronic Engineering

Identification of Insect Pollinator Species using Bioacoustics and Artificial Intelligence

Author:

Hafed Khalil

Supervisor:

Dr. Martin Trefzer

February 16, 2021

Abstract

Reliable identification of pollinator species is vital in order to monitor these species and reduce their decline. The reduction of pollinator species has occurred due to several factors, such as a change in land use, agriculture intensification, and climate change. Nowadays, typical methods used to monitor pollinators cause additional deaths of the specimen as they are required to be examined under a microscope or the presence of human activity such as using mobile applications. This thesis presents the development of an automated bioacoustics identification system of pollinator species. This is achieved through a combination of an environmental microphone and a Raspberry Pi. The latter uses a novel Time Domain Signal Coding (TDSC) based technique called Statistical Time Domain Signal Coding (S-TDSC) and the application of artificial neural networks are implemented. Statistical Time Domain signal Coding is compared to other time-domain based feature extraction techniques in principle, and the produced features are compared to features resulting from another frequency-domain based feature extraction technique. Statistical Time Domain Signal Coding is shown to be more efficient than other techniques as it eliminates the need for codebooks dependence and limits any audio signal fed from the microphone to only 122 features. These features are then processed by four different classifiers: Support Vector machines (SVM), Random forests, Extreme Learning Machines (ELM) and back-propagation. The classifier that provided the best accuracy was found to be ELM, where the best accuracy achieved was

82.14% in an experiment to identify bumblebees against three other species. This combination of bio-acoustics technique, enhanced TDSC and ELM is used to identify pollinator species. Finally, an embedded system consisting of a microphone and a Raspberry Pi is implemented and evaluated. The sound of flight is played through a speaker and then captured through the microphone to emulate field environment, this recorded sound is then fed to the Raspberry Pi in order for the feature extraction and classification to be performed.

Dedication

To my parents and my grandfather......

Declaration

This thesis is submitted for the degree of Doctor of Philosophy at the University of York. The research described herein was conducted under the supervision of Dr David Chesmore, and Dr. Martin Trefzer in the Department of Electronic Engineering, University of York between November 2015 and June 2019. This work is to the best of my knowledge original, except where acknowledgment and references are made to previous work. Neither this, nor any substantially similar dissertations has been or is being submitted for any other degree, diploma or other qualification at any other university.
This thesis contains 37677 words.
Hafed Khalil
Augest 2020

Acknowledgements

I would like to thank my supervisors Dr David Chesmore, who I wish a full recovery from his medical condition, and Dr Gianluca Tempesti for their excellent supervision, continued support throughout my PhD programme. A big thanks to Dr Martin Trefzer for his excellent supervision and believing at a very crucial time in my PhD. Many thanks to all of my friends here in England, who I had the pleasure to meet after I came and started my academic journey 9 years ago for the support on the personal and academic levels . Jorge, Elizavetta, Tamer, Lloyed, Amara, Ameera, Abhilasha, Shefali, Kaboos, Alia, Safwan, Jose, Raffael – god I'm goanna be in trouble if I miss somebody – Yaya, Ibrahim, Osama and everyone in Leeds, thank you all. Many thanks to my number one role model in life as well inspiration to get business done, my grandfather who showed a great example of hard work and dedication towards his goals, I love you and I pray that you rest in peace. I wold like to also thank all of the students and researchers who I shared the office with, you guys have made it always fun ! Last but not least I would like to thank my parents, brothers and sister who have supported me, and showed nothing but unconditional love. I love you all.

Contents

List of Figures

List of abbreviations	
Abbreviation	**Explanation**
TES	Time encoded speech
TESPAR	Time encoded signal processing and recognition
TDSC	Time domain signal coding
S-TDSC	Statistical time domain signal coding
CBD	Convention on Biological Diversity
FAO	Food and Agriculture Organisation
PoMS	Pollinator Monitoring Scheme
DEFRA	Department of Environment, Food, and Rural Affairs
NPPMF	National Pollinator and Pollination Monitoring Framework
kNN	k-th Nearest Neighbour
GMM	Gaussian mixture model
MFCCs	Mel- Frequency Cepstrum Coefficients
D	Duration
S	Shape
SVM	Support vector machines
RF	Random forests
IW	Input weights
LW	Output weights
ANN	Artificial neural networks
ELM	Extreme learning machines

SLFN	Single layered feed forward neural networks
RNNs	recurrent neural networks
CNNs	Convolutional neural networks
MLP	Multi-layer perceptron
RBF	Radial basis function network
LS-SVM	Least square support vector machines
FFT	Fast Fourier Transform
DFT	discrete Fourier transform
STFT	short-time Fourier transform
FWT	Fast Wavelet transform
CWT	Continuous wavelet transforms
DWT	discrete wavelet transform
FSA	Frequency stability assessment
LPC	linear predictive coding
GUI	Graphical user interface

Chapter 1

Introduction

Every day, there has been a global decline in insect pollinators, and this issue has been overly evident worldwide [1]. This decline has increased concerns over the effects of the continued loss of the benefits provided by pollination on agriculture and ecosystems. Therefore, the international recognition of the decline of pollinators has increased through the Convention on Biological Diversity [2]. Additionally, more research was done to discover more causes and impacts. Furthermore, 78% of the species living on our planet depend on services provided by animals and mainly insects, such as pollen transfer and reproduction, as well as maintaining genetic diversity within populations [3]. Additionally, various crops worldwide utilize pollinators for increasing the yield, for example, *Apis mellifera*, which is a honey bee [4, 5]. However, recent studies suggest that bumblebees and solitary bee species also significantly contribute more to crop pollination in the UK than *Apis mellifera* [6, 7]. Hence, pollinators play a vital role in natural and semi-natural habitats [8], and native wildflowers [9, 10]. Furthermore, insects-mediated pollination provides an ecosystem service to agriculture that is estimated to be 215 billion dollars in 2005 [5], which represents about 9.5% of the global food production economy.

Uniform standards for pollination in ordinary ecosystems are not available but would likely exceed these considering the significant number of ecosystem processes depend on a sufficient and healthy flora and insect environments [5]. Correspondingly, there is an extension of efforts to protect the pollinators from the dangers of further potential decline [1].

1.1 Humanity's current knowledge on pollinator decline

The importance of insect pollinators, both from an ecological and economic standpoint, has been highlighted by a growing body of evidence concerning population declines within many insect pollinator taxa [1, 11–14]. This issue is prominent within the international research community, and has captured the public consciousness, having been addressed by the United Nations under both the Convention on Biological Diversity (CBD) [15] and the Food and Agriculture Organisation (FAO) [16], as well as by several individual national governments [17, 18]. There are various known causes of pollinator decline, including changes in how agriculture operates in utilizing land, the application of chemicals, and climate change. A brief discussion is given in the following sections.

1.1.1 Changes in land-use and agriculture intensification

Decreased floral resources and nesting sites for wild pollinators caused by the degradation and fragmentation of natural habitats, resulted in the fall of pollinator diversity [1, 9]. According to a report by Nature Today, there were considerable losses to nectar resources in the United Kingdom, particularly in England and Wales, starting from the 1930s until the 1970s, a period significantly linked to agricultural intensification. Furthermore, according to Professor Bill Kunin: "wild bees and other insect pollinators are vital to the success of many important food crops and wild plants. Therefore, the relationships between floral resources and pollinating insect populations must be understood. Despite the stabilization seen recently, our research shows significant long-term declines in the diversity of nectar sources mirrored in a fall in the diversity of pollinator species. We are at a point where only four plant species account for more than half the nectar in Britain." [19]. The loss of field margins and hedgerow habitats associated with the more use of monoculture has caused a loss of spatially and temporally diverse floral resources used by insect pollinators [1, 20, 21]. Also, pollinator health is directly affected by the increased use of pesticides, and the increased use of fertilizer that has been linked with the decrease of natural wildflower communities, which is a source of forage for wild bees [5].

1.1.2 Climate change

Natural ranges of pollinator species are affected by the change in regional temperatures [1]. The latter can lead to increased invasions of non native species [21]. Additionally, change in regional temperatures can also lead to a mismatch in the spatiotemporal links between pollinators and the plants pollinated, resulting in a potential lack of both forage and pollination service. In addition, Intergovernmental Panel on Climate Change stated that pollinator species including bees, and butterflies are under an increasing rate of extinction due to global warming which has caused alterations in the seasonal behavior of species. In other words, bees emerging in different periods of the year when flowering plants were not available [22].

1.1.3 Invasive species

The invasion of none native pollinators, which naturally occurs, is a cause of the decreases in native pollinator fitness and fecundity [1], leading to the alteration of community dynamics and increasing inter-specific competition for floral resources, at a place that has not existed previously [1, 5]. Additionally, plants that are none native cause a decrease in local wildflowers through competition for resources; this can have a knock-on effect on specialist pollinator taxa [1].

1.1.4 Pests and pathogens

Parasitism by the Varroa Mite, also known as Varroa Destructor, has caused a significant decrease in European honey bee colonies. Diseases such as deformed wing disease, which severely affects the bee fitness, are caused by parasites as it is considered a viral vector [1]. Viruses of this kind may endanger the *Apis* colonies as they are highly infectious due to the proximity, which also may be detrimental to the health of surrounding wild bee communities [1].

1.2 The economic and ecological importance of insect pollinators

The importance of insect pollinators is often quantified in terms of their value as ecosystem service providers, specifically, their contributions to global agriculture and food production. In 2009, Gallai, Salles, Settele, & Vaissiee [4] estimated that global pollination services were worth 153 billion. A more recent estimate by the Food and Agriculture Organisation placed between 5% and 8% of global agricultural production by volume, worth an estimated 235-577$ billion, as directly attributable to animal-mediated pollination; while around 75% of our most important food crops, accounting for a 35% of global agricultural production, are at least partially dependant on pollinators to increase yield [23, 24]. The majority of the world's staple crops may be wind-pollinated (anemophilous), i.e., wheat, maize, barley, oats, and rice, but there is also evidence to suggest that insect pollinators are common visitors to many species presumed to be entirely anemophilous and that their visits may enhance crop yield [25]. Aside from crop production, domesticated pollinator species like the Western honeybee (*A*pis mellifera) provide additional sources of income to the people and communities that keep them [24]. Honey is a valuable commercial product [26], and the income gained from hiring out honeybee hives for agricultural pollination services can be considerable. A classic example of this is the hire and transportation of over two million honeybee hives from across the United States for almond crop pollination in California [27]; beekeeping is also an important poverty-alleviation tool in rural and developing communities [24]. Hence, the economic value provided by animal-mediated pollination in terms of employment within the agricultural sector [24]. Insect pollinators also contribute to human wellbeing in a more qualitative sense. Current research shows that human wellbeing, including our mental health, can be positively influenced by contact with nature and green spaces [28]. Since nearly 90% of all flowering plant species rely on pollination that is animal mediated in order to reproduce, maintaining diverse insect pollinator communities, especially within urban centers, is only likely to enhance these benefits.

1.3 History of pollinator decline

Since the 1950s European honey bees have faced documented regional declines [29], this is believed to have occurred due to the causes mentioned sub-chapter 1.1 and a fall in beekeeping in recent years [29]. Besides, honey bee colonies have been monitored by organizations such as BeeBase [30] and the British Beekeepers Association [31]. One of the most challenging aspects of measuring pollinators is the scope of their wild communities. However, in the UK, 13 species of bee have gone extinct since 1890, 8 of which are believed to have gone extinct between 1930 and 1950. The latter is a time associated with changes in land use, which was resulted by the outbreak of World War 2 [32]. This spread of agriculture caused the decline of several natural meadows, areas of unimproved grassland and hedgerows [9]. Consequently, fall in ranges of some bumblebee species over Europe and the USA [9,20,33], similarily, the extinction of bumblebee species, for instance *Bombus cullumanus* and *Bombus subterraneus* in the UK since 1940 [32]. The richness in native bee species declined in Britain and the Netherlands by comparing values in 1980 and post 1980. In comparison, hoverflies richness had an increasing trend in the UK and the Netherlands during the same period [34]. Also, there is evidence of a degree of homogenization in pollinator communities with 30% fewer species accounting for 50% of all the records post 1980 [34]. A more in-depth study utilizing similar data throughout more specific time frames agrees that after 1970 significant decrease in bee species richness happened in the Netherland. In comparison, bumblebee species richness falls in the UK. This increased spatial homogeneity of bee communities that occurred in the UK and the Netherland. On the other hand, after 1990, decreases in the richness in species began to lessen, as an increase in bumblebee richness at a national scale in the UK and the rate of homogenization within pollinator communities both the UK and the Netherlands slowed [35]. There is a significant issue with the data used to generate results in pollinator decline, which is that most of the past data compared to the modern approaches, which was collected in an unstandardized manner that has no application of systematic methods [36]. This data can still result in beneficial outcomes. However, the ability to obtain interpretable results from such data depends on consistent sampling efforts over time, space, and change [37] in motivation for making the recording might have also been biased results. Also, biological monitoring in the past was mainly done for the interest of a local scale [36], and it is viable to assume that the surveyors were targeting rare species. Therefore, the ratio of rare to common species might have high value throughout their records [38]. On the other hand, modern systematic monitoring is possibly less focused on recording full species inventories than on representative sampling, and so maybe recording a lower ratio of rare to common species [38]. This change in emphasis might be the reason behind results illustrating increased biotic homogenization over time. For instance, in case fewer species are being recorded because of difference in historical recorder motivations as opposed to actual decreases in rare species [38].

In addition, there are substantial problems associated with the evidence supporting insect pollinator declines [39]. Primarily, the lack of data concerning the abundance of individual insect pollinator populations [39]. The vast majority of studies exploring pollinator decline do so in terms of species occurrence, species richness, and range sizes [7, 11, 14, 40, 41], but none of these measures, although valid, allows scientists to make any judgments regarding how much an individual species has declined by over a given period of time within a given area. The main reason for this absence of abundant data is a global lack of centralized, systematic insect pollinator monitoring schemes [42]. Monitoring schemes exist in several countries, such as the Great Sunflower Project in the United States and the Wild Pollinator Count in Australia. Nevertheless, the aforementioned schemes are not aimed at the collection of species-level data, which are critical to the investigation of population trends. The UK's national Pollinator Monitoring Scheme (PoMS) is the only example of a nationwide, systematic pollinator monitoring project collecting species-level data in the world and they state that "whilst the distribution of some species of pollinator has become more restricted the extent of the declines in overall pollinator abundance are largely unquantified. The UK PoMS (Pollinator Monitoring Scheme) aims to better understand how insect pollinator populations are changing across Great Britain.". Hence comes the need to provide a method that can solve the problem of monitoring bees by quantifying them over a period of time over an area to collect data on their redundancy to help reduce the decline of pollinators.

Finally, the link between identifying species and reducing their decline can be found through the Prestonian shortfall [43]. The latter means that there is not enough information available regarding the abundance of most invertebrate species that include pollinators such as bees, wasps, and hoverflies. Additionally, this shortfall is the main reason behind the academic and industrial efforts toward insect pollinator conservation, which have resulted in an increased need to monitor insect pollinator populations. In other words, it is essential to observe how pollinator species populations are changing with space and time to target our conservation efforts. As a result, when it is found that the population of a particular species has decreased in abundance over time, it is then possible to start assessing why that has occurred. For example, was there an increase in pesticide use in nearby farmland, or has the habitat that a species relies upon becoming significantly fragmented in a specific part of a country?

1.4 Pollinator monitoring

Current methods of monitoring such as pan traps and which are described in detail in section (2.2.1) rely on visible morphological characteristics. Additionally, such methods require the killing of the species in order for it to be identified under the microscope, which is considered an inhumane method of monitoring and defies the purpose of monitoring pollinators, which is protecting them from decline. Several smartphone applications already exist to provide a method of visual identification [44]. However, such classification methods need human presence at the time of identification to carry out the decision process. Hence, these methods consume a considerable amount of time and require skilled individuals who obtain visual integrity since several bumblebee species seem similar to the untrained eye. Furthermore, classification techniques that are based on visual features (image recognition) are an option that would be beneficial. However, it is challenging to implement due to many factors, such as the movement of the pollinator, image resolution, and light. On the other hand, the buzzing sound of pollinators is relatively not as challenging to obtain as sound can be recorded remotely and continuously. Hence sound-based classification techniques are more practical than traditional surveying methods. Furthermore, Sound-based classification techniques are easy for the public to use and do not require a trained eye like image recognition techniques in the process of identification. Finally, the most important advantage is that bees are kept alive and are not required to be killed in order for them to be identified.

1.5 Project aims and hypothesis

The need for pollinator monitoring is getting more crucial every day, and widely-used techniques for species identification to date are still microscopy-based and require the manual collection of individuals, resulting in the individuals' deaths. There still has not been devised as a method of identifying pollinators automatically using bioacoustics and machine learning. Therefore, this project aims to develop an embedded system capable of acquiring the sound of pollinators from a field, then a novel technique for extracting the features from the acquired audio signals is implemented and compared with an existing feature extraction method. The features are then fed to four different classification algorithms, and an analysis as to which algorithm performed better is shown.

Hence the following hypothesis can be formulated:

"Bio-acoustics combined with machine learning techniques can remove the need for human intervention in species identification and avoid sacrificing insect species."

Having proposed this, will this technique help improve monitor pollinators and hence lead to the reduction in pollinator decline. Lastly and most importantly, will the help of machine learning and algorithms perform better than existing pollinator monitoring methods and provide better results in terms of quality and overcome the disadvantages of typical pollinator monitoring techniques?

Contribution 1: statistical time-domain signal coding

In this research, a novel technique called statistical time-domain signal coding (S- TDSC) is developed to extract the features from audio signals. This technique is entirely based on the time domain. Therefore it is not computationally expensive when compared to frequency domain-based feature extraction techniques. Additionally, S-TDSC does not require looking up tables as in previously designed time-domain feature extraction methods, and what makes S-TDSC unique and powerful in terms of audio feature extraction is the fact that it codes any audio signal fed to it to 122 features only regardless of the length of the audio file.

Contribution 2: Comparison between STDSC and a frequency domain-based feature extraction.

In chapter 3, a comparison between STDSC and a frequency domain-based feature extraction technique is shown as well as an in-depth discussion of why statistical time-domain signal coding has been chosen as the primary feature extraction method for this project.

Contribution 3: Classification methods for species identification

Moreover, in chapter 4, an example of 4 species classification is shown using the features extracted from the audio recording of the species using S-TDSC. The features are fed to four different classification models, which are support vector machines (SVM), Random Forests (RF), Extreme Learning Machines (ELM), and a Back Propagation. Finally, all of the classification models are explained in detail, and the performances of the four classifiers are discussed and compared. The classifier with the highest accuracy is set to be implemented in the embedded system shown in chapter 5.

Chapter 2

Background

2.1 National pollinator monitoring in the UK

Currently, the pollinator monitoring scheme in the UK is essential [17], and due to the recent concern over pollinator decline in the UK, (DEFRA) The Department of Environment, Food, and Rural Affairs has authorised the creation of a National Pollinator and Pollination Monitoring Framework (NPPMF). The latter sets out the design for a standardised monitoring programme that aims to gather long term data on the national pollinator populations in the UK [18,45]. This plan is to test the efficiency of monitoring methods nowadays and how they are applied to different types of participants, so they can enhance the understanding of the current trends of pollinator taxa in the UK and the place as to where they occurred. Therefore, allowing more targeted solutions to the causes of the decline in the future.

2.2 Common monitoring methods

There are many methods used to survey and monitor insect pollinator populations. These can be separated into two categories: active methods, where surveyors are involved in the capture of data or samples, and passive methods, where a range of traps are employed to collect data without the involvement of the surveyor [45]. Examples of both methods, as well as advantages and disadvantages, are illustrated in table 2.1. In recent years, there have been numerous studies that have explored the sampling biases of different insect pollinator survey methods, as well as their performance in relation to other methods. However, there is still no consensus as to which method or combination of methods constitutes the most effective approach to pollinator monitoring. In addition, the two most commonly used survey methods for insect pollinator communities are pan trapping and transect surveys [46–49]; they are also the two most commonly compared survey methods, in terms of their performance.

2.2.1 Trapping and transect surveys

Pan trapping, also called Moericke traps [50], is a passive sampling method that uses brightly-coloured bowls, filled with water, as surrogate flowers to attract foraging pollinator species, which then drown in the water [46,47]; different insect groups have evolved specific colour preferences in relation to their preferred source of forage [51]. Research suggests that oligolectic bee species, which are defined as bees that exhibit a narrow, specialized preference for pollen sources are captured more often in pan traps whose colours are similar to these of their preferred forage [52]. While Saunders and Luck indicate that colour preferences are context-driven, and may change depending on habitat or background floral colour [53,54]. Research suggest that pan traps may catch fewer insects in florally-rich habitats due to competition between flowers and the bowls for insects [55–57]. Pan trapping is often considered to be more cost-effective than more active survey methods since it requires less in-person time within the field [47]. However, time is still required to sort through samples and maintain the equipment [49].

There are also other types of trappings such as malaise trapping, vane trapping, and nesting trapping. Malaise traps consist of an open-sided tent-like structure and function by intercepting flying insects using a central fabric wall. Insects are funnelled up towards the upper-front corner of the tent, where they are captured within a plastic bottle that is sometimes filled with water, in a manner similar to pan trapping. Malaise traps have been compared with pan trapping, in terms of their performance, with Bartholomew showing that pan trapping and Malaise trapping catch approximately similar species, even though pan traps caught a higher overall abundance [58]. Campbell and Hanula found that pan traps performed better than Malaise traps in terms of both species richness and abundance while showing that the addition of coloured panels to Malaise traps also increases the abundance of insects captured [59]. The advantage of these traps lies in the fact that surveyors can leave them unattended for long periods with little reduction in efficacy [58]; however, regular trips must be made to empty the traps to the plastic bottle in which the insects are captured [58, 59]

On the other hand, vane traps are becoming more prevalent within the literature concerning insect pollinator survey methods. Vane traps consist of a plain plastic bottle, filled with water, to which two brightly-coloured, vertical "vanes" are attached above [60]. Multiple colours could be used, although, so far, only yellow and blue have been tested [60–62]. Vane traps function in a similar way to pan traps, using bright colours to attract foraging insects. On their own, they have been shown capable of catching a diverse selection of bee species in a range of habitats [60, 61], with blue vane traps being the most effective [62, 63]. In comparison to other methods, namely pan trapping, blue vane traps have been shown to catch significantly more insects in terms of abundance, than pan trapping [63]. In fact, Kimoto et al. suggest that, due to their effectiveness in initial studies, fewer vane traps may be needed to carry out a monitoring survey than other passive methods like pan traps [61]

Finally, trap nests are a much more specialist survey tool than other trapping methods. This is becuase they are only useful for monitoring the diversity of cavity-nesting bee species such as the Megachilidae in the UK [46–48]. However, this method may also provide data on other cavity-nesting insect species and their parasites. These sampling abilities for these traps are reliant upon their design. Specifically, the type of materials used to construct the nesting tubes, the size of the tubes themselves, and only collects data on a subset of the overall insect pollinator community [46, 47]. However, they are a useful survey tool when used in conjunction with other methods [47, 64].

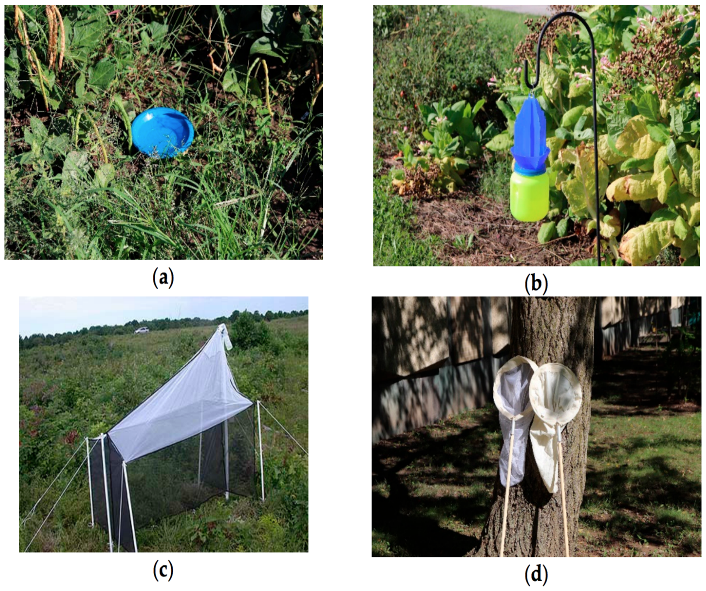


Figure 2.1: Common sampling methods: (a) Pan traps (b) Blue vane traps (c) Malaise traps (d) Sweep nets [64]

Transect surveys are an active sampling method that involves walking along pre-set routes at a slow pace and recording the number of insects observed. If the study aims to record species-level data, it is also common for individuals to be captured using nets. As with all active methods, transect surveys are open to collector bias and may have additional biases relating to the size and flight speed of individual taxa. For example, Potts, Evan, Boone (2005) list transect surveys as less likely to sample smaller, faster-flying insect pollinator taxa. Unlike passive methods such as pan trapping, transect surveys do allow observations to be made regarding insect pollinator behaviour [49]. This method is quite labour-intensive, requiring extensive periods to be spent in the field, together with taxonomic skills and high levels of concentration. However, little equipment is needed beyond a net, also since the captured samples are not drowned, they can be easier to identify [46, 49].

Direct comparisons between pan trapping and transect surveys, in terms of their performance, are common within the literature [47–50, 55, 56, 65]. Most of the studies encourage using pan trapping over net sampling in terms of better monitoring results. however Westphal found that pan traps better represented bee species richness than either variable or standardised transect surveys [47].

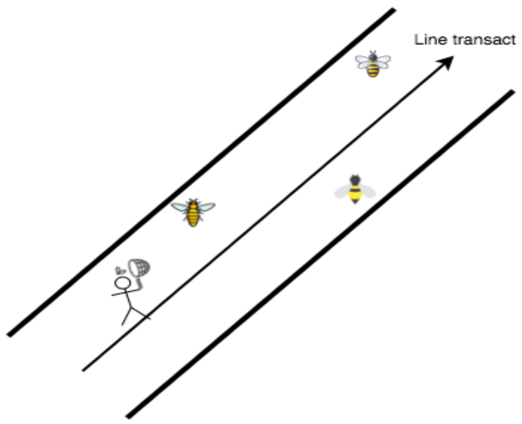


Figure 2.2: Transact survey

Criteria	**Passive methods**	**Active methods**
Example	Trap nests, pan traps and vane traps	Hand netting along a set distance over a set amount of time.
Advantages	Does not require experience in collecting samples	Measures population per density unit area
Disadvantages	Does not allow observation of pollinator-flower interaction or measure population density	More experienced collectors will get more collections than those with less experience

Table 2.1: Examples of passive and active methods as well as advantages and disadvantages

2.2.2 Bio-acoustics

Since pan traps and net sampling both depend on visible characteristics as species have different appearances in terms of their body size, hair colour and pattern. Within the same species, workers are smaller than queens. While males are typically different from females in colour scale as well as physical characteristics as males have different length of antenna compared to females, and they also lack pollen baskets in their legs [66]. Furthermore, there are many web and smartphone applications that can be used for image identification [44, 67].

However, as mentioned in the introduction, such classification techniques have several drawbacks such as requiring humans to carry them out in the decision process, and technical issues arising from the movement of the species and resolution of the image. On the other hand, the sound of the flight of pollinators is relatively easy to acquire remotely and continuously; hence, pollinator monitoring is conducted easier when compared to the old scientific surveys that collect species individually. Moreover, sound-based animal identifications have been made previously; these animals included frogs and birds. For instance k-th nearest neighbours and support vector machines have been implemented as an audio feature extraction method to classify frog species. The result of this classification method reached 90% accuracy for six species [68]. Additionally, audio features extracted from four types of passerine birds using the Gaussian mixture model (GMM), and Mel- Frequency Cepstrum Coefficients (MFCCs), which is observed in section 3.4.3 resulted in 90% identification accuracy [69]. In terms of bird species, there was another approach used to classify 28 bird species using probabilistic models, which has resulted in around 84% classification accuracy [70]. While the latter experiment reached 100% in some instances, it also decreased to under 10% as well, which was justified in the research paper with the shortage of data available. Furthermore, different groups compared accuracies of other machine learning algorithms for the dataset of birds and frogs [71] and bird species [72]. SVM performed well when implemented on both datasets to reach more than 90% identification accuracy. Furthermore, there have not previously been substantial insect sound identification applications, especially for monitoring pollinators. Nevertheless, the minor amount of work that has been done in this field was either to monitor biodiversity or to reveal pests, such as larvae [33]. Additionally, several artificial neural network structures have been designed for sound-based classification of insects such as beetle species when they bite on fibre. These studies included multilayer perceptron (MLP), self-organising map, and learning vector quantisation implemented on 3 to 4 audio files of each beetle species. The resulted classification accuracy reached 80%. Similarly, using the same neural network structures, 25 British

Orthoptera species were classified with an accuracy of 99%. However, the authors have justified such high accuracy with the high-quality audio recordings that included minor noise interference [73]. Furthermore, a technique close to the human recognition system was implemented on a sample of 313 species of crickets, katydids, and cicadas [74]. The scientists achieved the feature vectors through a cascaded linear frequency cepstral coefficients, probabilistic neural networks and generalised method of moment (GMM). As a result, the achieved classification accuracy reached 86%. The scientists stated that the classification accuracy would increase if these experiments were applied on the subfamily level [74].

Various websites and mobile applications for identifying animal species from sounds exist. Generally, animal sound identification depends on detecting structured sounds like a dog barking or a horse neighing. However, insects that fly make sounds that are biologically defined as monotonic buzzing, which is very different in nature than other animals as it is unstructured. Hence insect classification could easily be mistaken because of the nature of the input audio signal. Moreover, sounds produced by pollinator species such as bumblebees can also vary depending on different circumstances, for instance, buzzing during flight, hissing, and sonication. Additionally, sonication is the sound produced when bees extract pollen from anthers. These different sounds occur as a result of oscillations of the flight muscles inside the metathorax. In case the pollinator species are flying the frequency at which an undamped system will vibrate is called the natural frequency [75]. Furthermore, in order to get the pollen out of individual flowers, bumblebees make sonication sound, which occurs at significantly high frequencies [75]. Furthermore, through placing their thorax near anthers, bumblebees make a sound at a noticeably high frequency of around 400 Hz [9, 76]. Furthermore, when disturbed, bumblebees hiss [77], the hissing was discovered to happen at the presence of potential intruders. It was evident that hissing can occur by simulating an intruder though vibrating the nest or raising the density of CO2 in air [77]. Higher hissing and sonication frequencies have been noticed to be made when the species move their flight muscles without moving its wings [75, 77–79]. Finally, researches have proven that when bumblebees are placed in a large space or when the temperature goes down, they make audibly distinctive sounds [78].

2.3 Observation of the used species

The audio recordings to be classified in this project are produced from bumblebees, honey bees, common wasps, and hoverflies. Also, bumblebees and honeybees are both under the bees family, in which there are around 20,000 species that have almost identical body shapes, which can only be distinguished by a trained eye [80]. Most of the flying insect pollinators including bees have five shared physical characteristics, which are two sets of wings, the outer shell, also known as the exoskeleton, abdomen, thorax and head as shown in figure 2.3 [80].

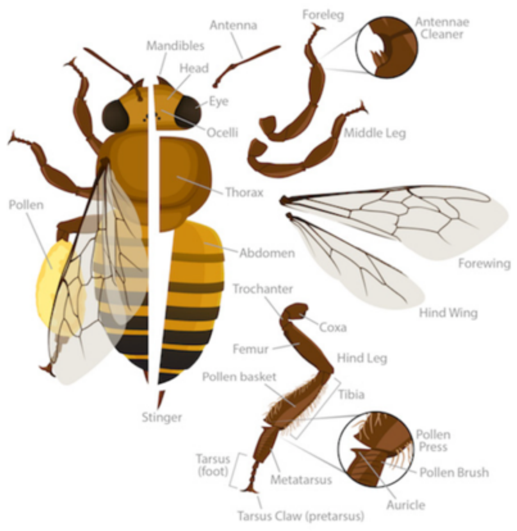


Figure 2.3: Honey bee [79].

When bees fly, their wings make a 230 beats per seconds flap which results in an air vibration and therefore sound is generated. The faster the wings beat, the higher the pitch of the sound [81].

Bees make buzzing sounds for several reasons such as communications and pollination; the frequency seems to vary based on the reason for flight. According to laboratory experiments done by Hassall, the buzzing frequency bumblebees produce when roam flying for no purpose varies between 140.7 Hz to 218.7 Hz, Whereas under the same circumstances honeybees produce around 171.2 Hz [82].

Furthermore, when pollination occurs, bees create sound by vibrating their bodies, and for that, they use the same muscles used to fly but do not fly. The resultant rapid vibrations can reach 440 beats per second; therefore, pollen gets extracted from flowers [81]. However, some flowers are harder to extract pollen from than others, so pollinators require extra work. For instance, tomatoes and blueberries have their pollen trapped inside the pore. Therefore, bees hold on into the flower, bite inside any small hole they find and vibrate quickly until the pollen is unlocked [81].

Common wasps, though, have some standard features similar to bees, such as their slender, segmented bodies, their stings, and living habitat. However, what sets them apart from bees are their pointed lower abdomen, slender waists, and legs with fewer hairs [83]. There is a significant number of around thirty thousand wasp species today which all vary in terms of the colour scale and dimensions. In terms of habits, wasps are considerably different, and their connection to other creatures can significantly vary [83]. Based on a study, the frequency of the flight of wasps varies between 157.5Hz and 175.4Hz; however, the frequency of the post-attack buzzing sound of a common wasp varies between 159.8Hz and 171.6Hz. Besides, no studies show the frequency of pollination buzzing frequency of common wasps [82].

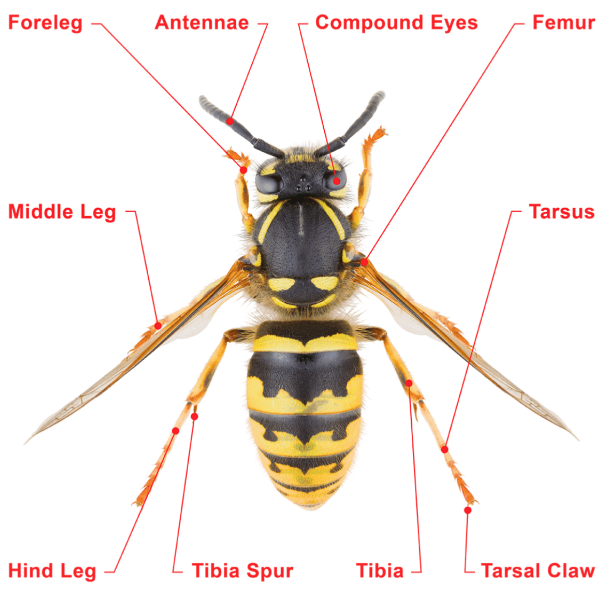


Figure 2.4: Common wasp [82]

Furthermore, There are about 6000 species of hoverflies which often referred to as flower fly from their behaviour of hovering around flowers [84]. Hoverflies morphological appearance has black and yellow stripes, like bees and wasps, which act as a form of protection. Due to their appearance, they can often be confused for a bee or a wasp. These stripes act as a form of camouflage to help the hoverfly avoid potential predators who think they can sting [84]. They do, however, only have two wings, while bees and wasps have four. They are also crucial in pollinating flowers and are natural enemies of pests. Because of this, farmers have been using them for biological control, a form of pest management [84]. Based on a study, the frequency of the sound of the flight of flies varies between 159.4Hz and 171.3 Hz. However, the frequency of the post-attack buzzing sound of a common wasp varies between 152.2Hz and 177.1 Hz [82]. Finally, there are no studies found to show the pollination buzzing frequency of hoverflies.

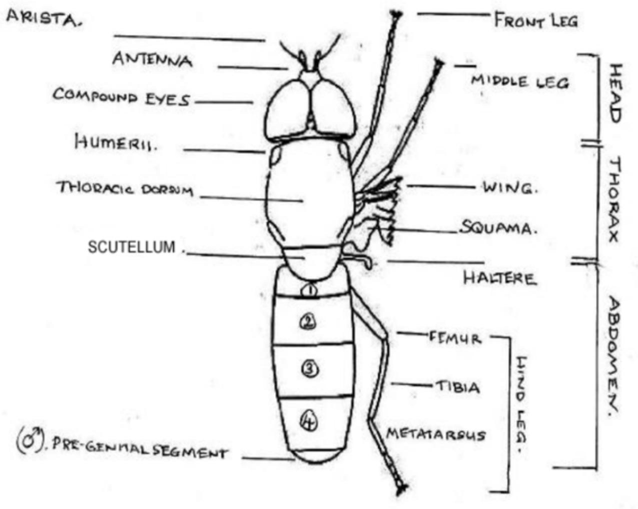


Figure 2.5: Hover fly [83].

2.4 Summary

This chapter reviews three related areas to this project, the national pollinator monitoring in the UK, the common monitoring methods, and bioacoustics combined with a brief overview of machine learning techniques for wildlife classification. In particular, the advantages and disadvantages of common methods used for pollinator monitoring are discussed. Similarly, the benefit of using the combination of bioacoustics and machine learning technique is illustrated.

Chapter 3

Feature extraction

Feature extraction has posed a challenge to scientists for many years, due to the need to isolate the optimum set of features for the best solutions to recognition problems. The features of the audio files of the bee species are extracted using time-domain signal coding (TDSC). Thus, this chapter begins with a description of the audio files dataset that feature extraction will be performed on, which will be fed to the classification stage. Then the chapter provides an in-depth observation of TDSC as a feature extraction method, then covers the principles, history, and development of TDSC since the 1970s, when it was first developed as time-encoded speech (TES) [85], up until its current format. Additionally, a novel approach based on TDSC and statistical time-domain signal coding (S-TDSC) is introduced and discussed. This is followed by an illustration of its classification parameters. Moreover, a frequency domain-based algorithm implemented in smartphone applications used to identify songs is produced to extract features from the audio file sets. Finally, a comparison between the time domain and frequency domain-based techniques is made, and the use of S-TDSC to extract the features is justified.

3.1 Data collection

The acoustic data were collected by Thomas Dally as a part of his PhD research at the environmental science school at the University of Leeds. These acoustic recordings of the species occurred between June-September 2016 and June-August 2018 in multiple sites around Leeds, West Yorkshire, Wimborne Minster, and Dorset. Sites varied in terms of the diversity of insect and plant species present, including wildflower meadows, brownfield sites, and urban parks and gardens. Acoustic data were collected using an omnidirectional Sony microphone attached to a Sony ICD- PX312 Dictaphone. The microphone was placed near flower-visiting insects, no-more than 5cm away, while they were foraging from or visiting flowers, as illustrated in figure 3.1. Once airborne, at least ten seconds of flight sounds from each individual was recorded. Each individual was identified by eye during this recording period. Once flight sounds from an individual had been recorded, Tom has immediately moved to another individual, preferably from a different pollinator species, in an attempt to reduce the likelihood of recording the same individual multiple times. Audio files were recorded at 32 kbps/44.1 kHz in the MP3 format.

Figure 3.1: Acoustic recording of flying insects taking place in the field by Thomas Dally.

Finally, the data set consists of audio files that correspond to 4 different species: bumblebees, common wasps, hoverflies, and solitary bees, the number of audio file recording for the species is 69, 21, 42 and 39 respectively.

3.2 Time domain signal coding (TDSC)

Time encoded speech (TES) consists of novel signal processing and pattern recognition techniques that were developed to represent and classify band-limited signals. TES relies on the delivery of codes that describe successive segments of speech waveforms. In other words, the speech waveform is divided into segments between the sequential real zeros of the function. For every segment a code of 1 word is derived from two parameters of the segment, which are the time duration and its shape, but quantised. Duration indicates the number of elements between two successive zeros, and shape is in principle the number of negative maxima and positive minima during the same period. King and Gosling used TES to encode speech signal waveforms for a small bit rate over poor quality channels. This method has subsequently been used in several applications, including acoustic condition monitoring of machinery [86] and heart sound analysis and defect identification [87].

In 1995, King developed time-encoded signal processing and recognition (TESPAR) [85], although this technique is based on TES schemes, it incorporates signal recognition and classification as matrix data structures are developed. The basis of TESPAR was developed according to a specific mathematical description of waveforms that involves polynomial theory. The latter exhibits how a band-limited signal can be expressed entirely through the locations of the real and complex zeros. Using these locations, a vector quantisation process was implemented to code these data into a vector that consists of about 30 discrete numerical descriptors [87]. The TESPAR coder outputs a simple numerical symbol stream that could be transformed into a numerous progressively informative structure of matrices. For instance, the S- matrix, which is a single-dimension vector, that represents a histogram which shows how many times each TESPAR coded symbol occurred in the data flow. Another distinctive data set is the two-dimensional histogram or A-matrix, produced from the frequency of symbol pairs, which do not have to adjacent. These developed matrices are fed into an artificial neural network (ANN) for classification, and pattern recognition are performed on the signal. This technique was used in the verification of the identity of individuals, through them speaking a simple common phrase [85].

Similarly, the TES technique that the novelty of this thesis is based on is called Time-domain signal coding (TDSC). this technique was developed through further development in a time encoded signal processing and recognition (TESPAR) [85]. TDSC is used to observe the audio signal in the time domain, where the signal is divided into epochs between each successive zero crossings.

Each segment is then analysed alone, and the number of positive minima and negative maxima for each of the epochs are determined and referred to as the shape (S). Moreover, the number of samples within each segment of the signal is determined and referred to as duration (D).

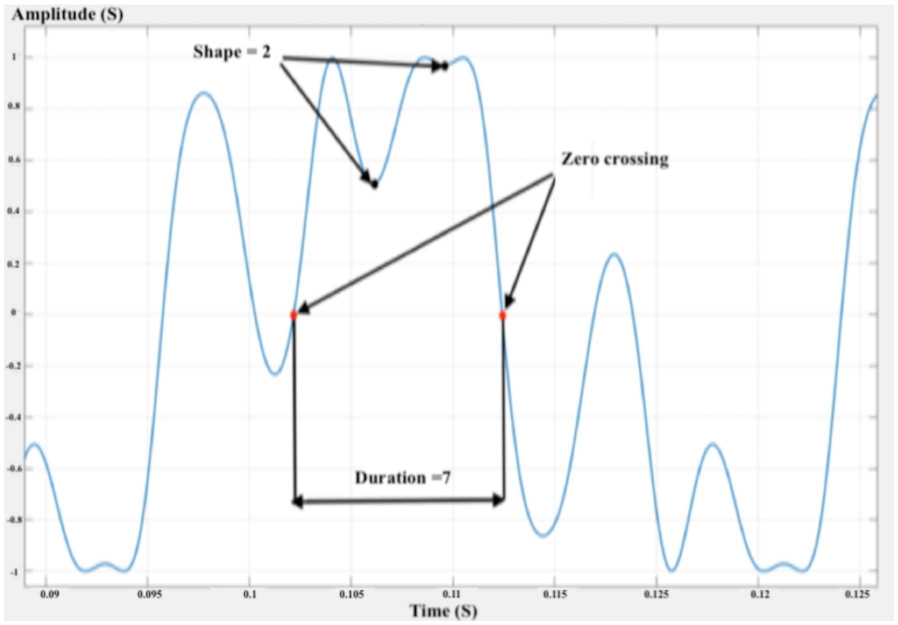
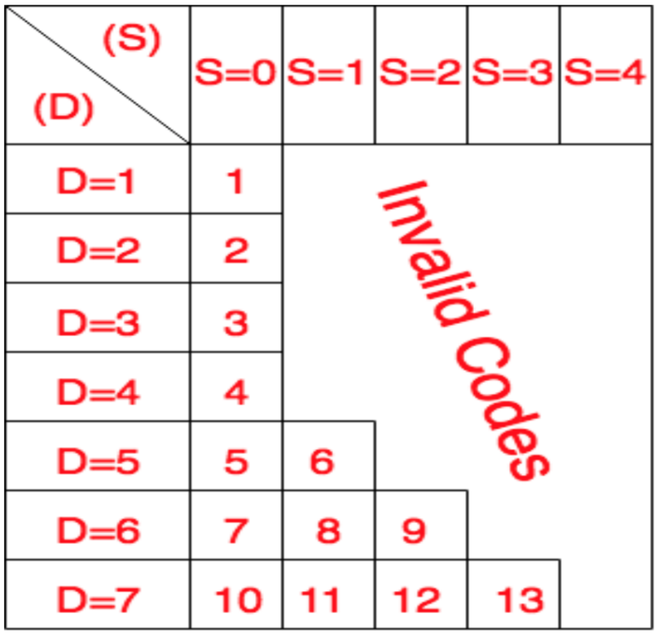


Figure 3.2: shape, duration and zero crossing on a waveform

These results are then given a code obtained from a lookup table (codebook). The above figure presents a graph on which an epoch is indicated between the two zero crossings shown by the red points. Between these zero crossings, there is a shape of 2 and a duration of 7. Therefore, based on the following codebook, the D-S pair give a code of 12.



(S) / (D)	S=0	S=1	S=2	S=3	S=4
D=1	1				
D=2	2				
D=3	3				
D=4	4				
D=5	5	6			
D=6	7	8	9		
D=7	10	11	12	13	

Invalid Codes

Figure 3.3: Example of codebook

When all of the waveform is encoded using this method, the result is a unique representation of the signal, through a series of codes. However, when TDSC is used as a feature extraction technique, the codebook has to be defined first. The other features outside the scope of the codebook will lead to errors in the representation of the derived waveform. For instance, suppose there is an epoch that has a shape of 5 and a duration of 6. Therefore, there will be no code associated with it based on this pair of duration and shape, according to the codebook illustrated in figure 3.3. Therefore the code that would be assigned to it is the maximum duration and the maximum shape in the codebook, which is 13. Hence an error has occurred. At this point, the codes are obtained and are available for manipulation to make the feature vector, which is then fed to the ANN for the classification stage. One of the pre-existing manipulations is the production of a histogram of the frequency of occurrence of codes. Therefore, the S-matrix (1-dimensional) shown in the subfigure(b) of figure 3.4 is produced.

Furthermore, another technique is to examine the occurrence of pairs of symbols over time to produce a histogram, which illustrates the quantity of symbols i and j of the codes that have occurred after each other. In other words, how many times i is followed by j by a lag L [69]. A 2-dimensional histogram, the A-matrix, can be formed, and illustrated in subfigure C of figure 3.4 and expressed mathematically as:

$$a_{ij} = \frac{1}{(N-L)} \sum_{n=L+1}^{m=N} X_{ij}(n) \text{ [79]} \tag{3.1}$$

Where:

- a_{ij}=element (i, j) of matrix A.
- $L = lag.$
- $x_{ij}(n) = 1$, if t(n)=i and t(n-l)=j (0 otherwise) and $t(n) = nth$ symbol

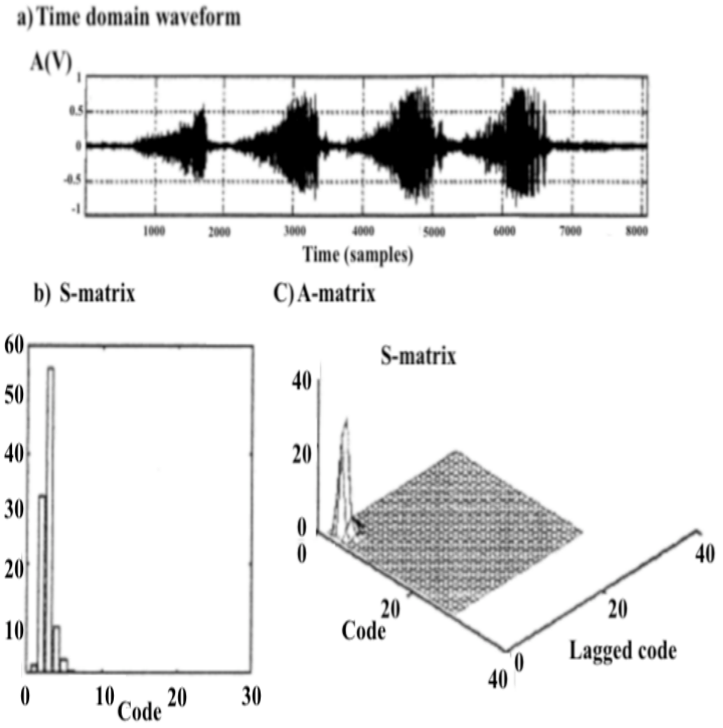


Figure 3.4: Waveform, A matrix and S-matrix of Grey Bush Cricket [69]

3.2.1 The mathematical basis of TES, TESPAR, and TDSC

In 1948, Licklider and Pollock proved that a decreased speech wave to a square wave with a variable period through infinite peak clipping, while still being intelligible is possible [88]. This proof led to a square wave that is distorted but 90% intelligible. The resulting wave was found to have only one common feature with the original un-clipped wave, the location of the zero-crossing points. Bond and Cahn then improved Licklider and Pollock's work by showing how a band-limited signal is fully described by the locations of its complex and real zero [89, 90]. This is as follows: Assuming f(t) is a band-limited signal in (o,w), and V(f) is its double-ended Fourier transform, the function f(z) is described as follows:

$$f(z) = \int_{-w}^{+w} V(f).e^{jw\pi f z} df \tag{3.2}$$

$z = t + ju$ is a complex variable whose real axis coincides with the real time axis, whereas $f(t)$ is a real entire function described by the location of zeros that may occur in real or complex conjugate pairs. Moreover, $f(z)$ is also expressed by the infinite product:

$$f(z) = f(o) \prod_{n=1}^{\infty} (1 - \frac{z}{z_n}) \tag{3.3}$$

Voelcker then proved the TES theory, which has led to the mathematical descriptions of the locations of all zeros in the complex signal. These locations are mapped to a sequence of discrete locations in a vector.

As mentioned earlier, TES analyses an input signal by dividing it into segments between successive real zeros. Each of these segments are called an epoch, and these epochs are allocated a code in the form of a single digital word. This code is given by the two parameters of each epoch, which are the quantised time duration (D) given by the number of samples and the shape (S), given by the number of positive minima and negative maxima between zero crossings. As a result, an accurate approximation of the complex zeros within the epoch is obtained.

Gosling developed TES further with the addition of signal recognition and classification to the set of codes initially obtained by TES, naming this new technique TESPAR. The latter was then expanded to TDSC by applying matrix scaling, matrix normalisation, and other methods to automatically select code books [91]. TESPAR and TDSC allow the development of matrix data structures by manipulating TES codes in the form of either A-matrix or S-matrix. These matrix data structures characterise the signal, given that it is stationary over time intervals. Moreover, A-matrix and S-matrix can be used as inputs for ANNs, allowing for recognition and classification techniques to be implemented.

3.2.2 TDSC for insect recognition

Swarbrick raised the question of whether TDSC could be applied in other areas, after using it in his work on acoustic diagnoses of heart defects [87,91]. Chesmore then used this technique to classify different species of Orthoptera (grasshoppers, crickets, and bush crickets) [92]. Chesmore was able to classify 13 species of Orthoptera with 100% accuracy under low noise conditions [33].

3.2.3 D-matrix

D-matrix is another feature extraction technique based on TDSC. This method was developed to replace the code book [93]. Therefore, rather than mapping the D-S pair combination onto a codebook, the code consists of the shape and duration. For instance, using the codebook in figure (3.3), if an epoch has D=7 and S=2, it will be given the code 9. However, using the D-matrix method, the assigned code will be calculated using the following equation:

$$Code = ((S \times S_f) + D) \tag{3.4}$$

Therefore, if the scaling factor $S_F = 100$, which is an experimental value that produced the best coding result, which led to an improvement in classification accuracy when compared to other scaling factors. The result of the equation will be a code of 209 for a duration of 7 and a shape of 2. Similarly, when an epoch has D=5 and S=2, the code will be 205. As a result, the D-matrix gives a solution for the pre-generated codebooks and the related issues of optimisation and determination. However, the main disadvantage of this technique is that it is not suitable for an audio signal such as pollinator species recordings with a considerable amount of zero crossings and a large number of duration and shape values and hence, a significant feature vector length. As a result, this technique does not fit the aim of this project, which is implementing an embedded system that can be deployed in a field, whilst keeping the computational complexity such as memory and processing power to a minimum.

3.3 Observation of statistical time-domain signal coding (S-TDSC)

A common issue with these TDSC methods in the feature extraction stage of this project was the length of the feature vector. When the feature space vector is passed to any classification algorithm (discussed in the following chapter), it uses a vast amount of memory and computational resources. The result is that either the compiler of the software (Matlab and Python) takes a long time to manage all matrix manipulations or the consumption of considerable memory space. Hence, the software crashes. S-TDSC solves this problem by applying statistical manipulations to the duration and shape values so that every audio file signal is coded in a feature vector that consists of just 25 samples.

Firstly, in S-TDSC, the duration of each epoch indicating the number of samples between zero crossings is extracted and arranged in the order of occurrence into a vector labelled 'D'. Similarly, the number of positive minima and negative maxima in each epoch is determined and stored in a vector labelled 'S'. Other vectors are also initialised and labelled 'maxima' and 'minima', where the amplitude values of the negative maxima and positive minima are stored, respectively. For instance, figure 3.5 illustrates the plot of an array x where x = [2,1,4,2.5,6,4,8,-6,-4,-5,-4,-5,-2,-5,-4,-5,2,1,4,2.5,6,4,8,0].

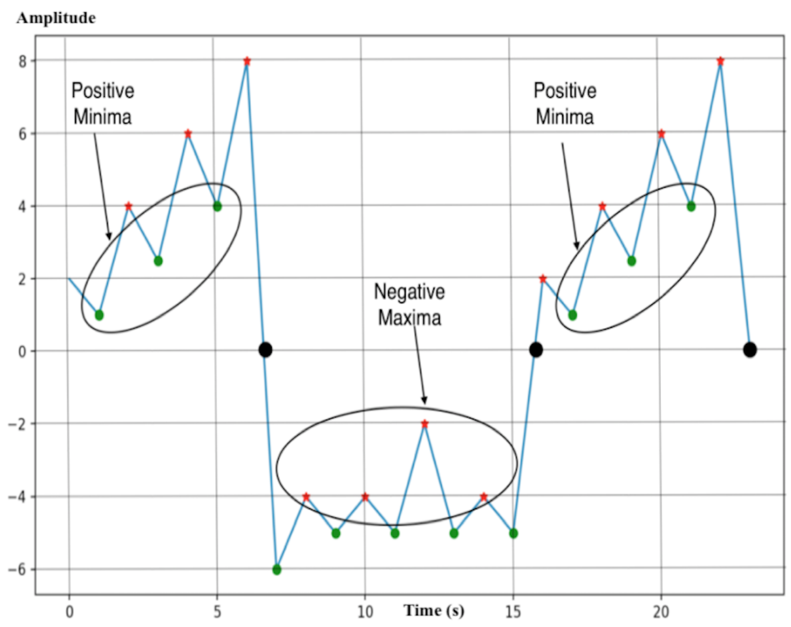


Figure 3.5: Positive minimas, negative maximas and zero crossing on a waveform

Therefore, the vectors D, S, maxima, and minima are as follows:

- D = [7., 9., 8.]
- S = [3., 4., 3.].
- Maxima = [-4.0, -4.0, -2.0, -4.0].
- Minima = [[1.0, 2.5, 4.0], [1.0, 2.5, 4.0]].

The reason for choosing D and S is because it has been shown by the work implemented by Chesmore [92], that these two parameters best describe the signal. Also,experiments showed that when the vectors maxima and minima were chosen as parameters for the feature extraction technique to be applied, the result increased the classification accuracy. The latter resulted from passing the feature vector to the classification algorithms described in chapter 4.

Additionally, more parameters were initially taken into account for the feature extraction stage. However, after experimenting with these parameters, the result was a decrease in classification accuracy. Hence these parameters were removed. Additionally, the removed parameters included the number of negative minimas and positive maximas stored in a vectore named L, the magnitude of samples in vector L, the number of the zero crossings along the signal, the skewness, and the signal variance.

Furthermore, the conducted experiments included using these removed parameters independently and combined with the other removed or chosen parameters for the feature extraction technique to be implemented, then using the result for the classification stage.

The following flowchart illustrates the S-TDSC feature extraction algorithm:

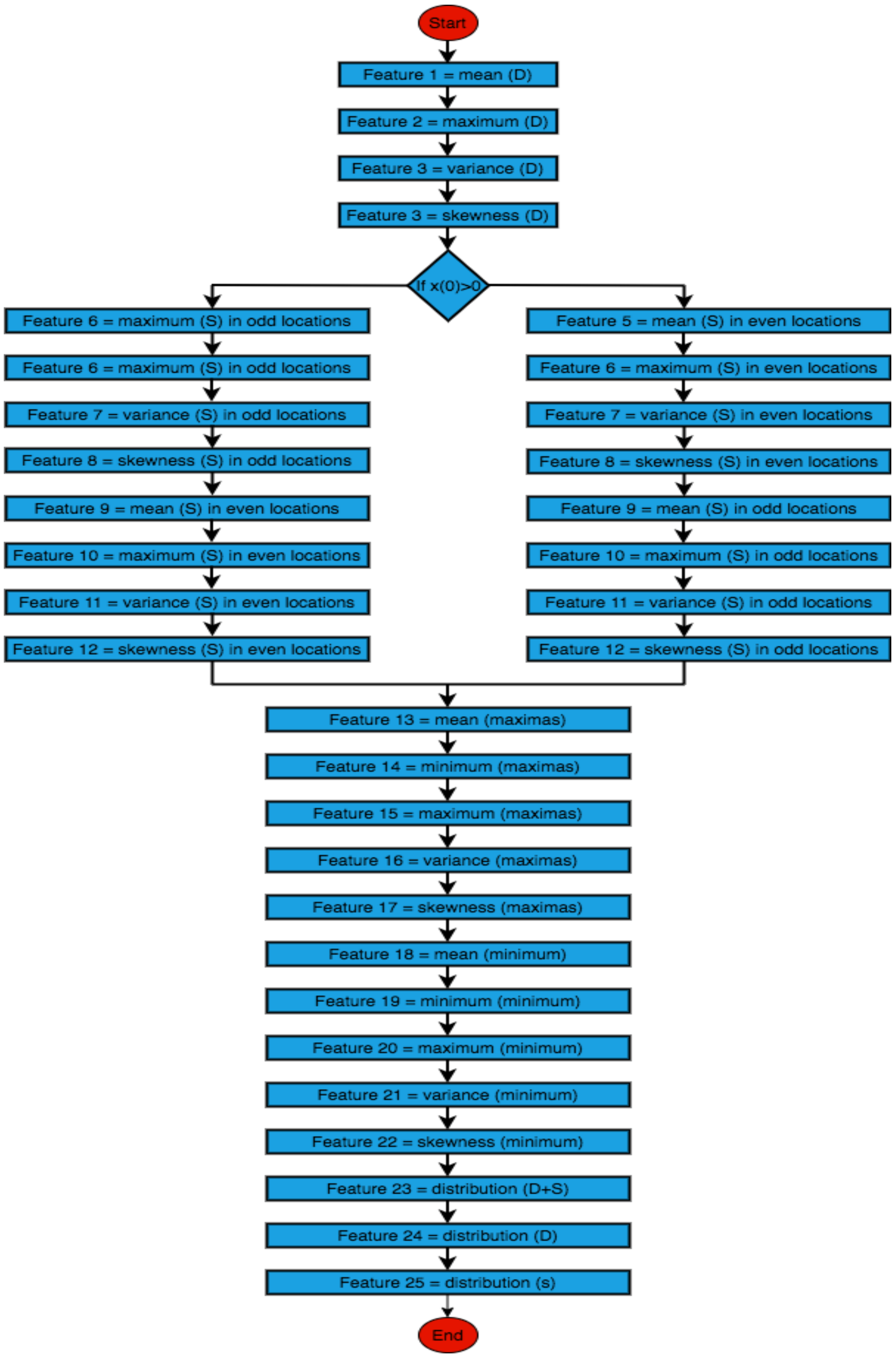

Start
Feature 1 = mean (D)
Feature 2 = maximum (D)
Feature 3 = variance (D)
Feature 3 = skewness (D)
If x(0)>0
Feature 6 = maximum (S) in odd locations
Feature 6 = maximum (S) in odd locations
Feature 7 = variance (S) in odd locations
Feature 8 = skewness (S) in odd locations
Feature 9 = mean (S) in even locations
Feature 10 = maximum (S) in even locations
Feature 11 = variance (S) in even locations
Feature 12 = skewness (S) in even locations
Feature 5 = mean (S) in even locations
Feature 6 = maximum (S) in even locations
Feature 7 = variance (S) in even locations
Feature 8 = skewness (S) in even locations
Feature 9 = mean (S) in odd locations
Feature 10 = maximum (S) in odd locations
Feature 11 = variance (S) in odd locations
Feature 12 = skewness (S) in odd locations
Feature 13 = mean (maximas)
Feature 14 = minimum (maximas)
Feature 15 = maximum (maximas)
Feature 16 = variance (maximas)
Feature 17 = skewness (maximas)
Feature 18 = mean (minimum)
Feature 19 = minimum (minimum)
Feature 20 = maximum (minimum)
Feature 21 = variance (minimum)
Feature 22 = skewness (minimum)
Feature 23 = distribution (D+S)
Feature 24 = distribution (D)
Feature 25 = distribution (s)
End

Figure 3.6: The S-TDSC algorithm

The feature vector consists of 25 elements. Firstly, the mean, maximum, variance, and skewness operations are applied to the vector D to determine features 1-4. Features 5-8 depend on the sign of the first sample in the audio signal; for example, suppose the signal is X, shown in the above graph, the first value of x is a positive number, which indicates that the first epoch is above the horizontal axis. Hence, elements 5-8 in the feature vector are the mean, maximum, variance, and skewness of the positive minimas, which are placed in the even number positions in the vector S. Similarly, features 9-12 are the mean, maximum, variance, and skewness of the positive maximas located in the odd number positions in the vector S. In contrast, if the first element of the signal is a negative number, then elements 5-8 of the feature vector are the mean, maximum, variance, and skewness of the negative maximas, followed by the features 9-12, which are the same statistical operations but for the positive minimas.

Additionally, features 13-17 and 18-22 are the mean, minimum, maximum, variance, and skewness of the vectors maxima and minima. There is an additional statistical operation applied to maximas and minimas, which is the minimum operation. The latter is added because when the minimum operation is applied to the D and S vectors, it results in zero improvements in the classification accuracy, as shown in the following chapter.

Feature 23 consists of 40 samples. In other words, feature 23 is a vector on its own that consists of 40 samples added to the original features vector. Figure 3.7 illustrates the original vector added to it feature 23 highlighted in red as follows:

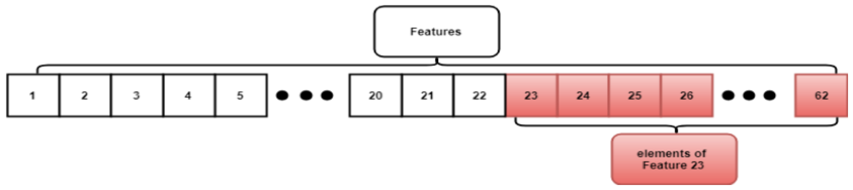


Figure 3.7: first 23 elements of the feature vector, where feature 23 consists of 40 elements.

The flowchart below illustrates how the elements comprising Feature 23 are obtained:

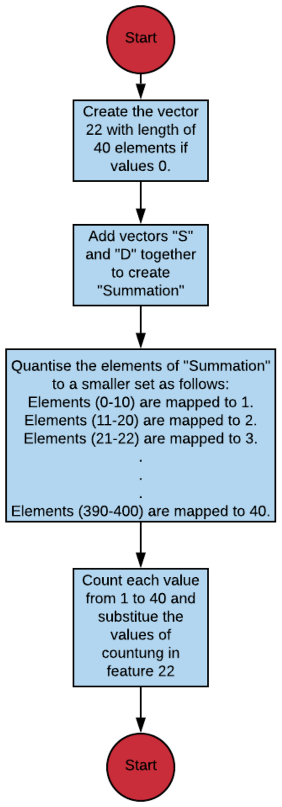

Start
Create the vector 22 with length of 40 elements if values 0.
Add vectors "S" and "D" together to create "Summation"
Quantise the elements of "Summation" to a smaller set as follows:
Elements (0-10) are mapped to 1.
Elements (11-20) are mapped to 2.
Elements (21-22) are mapped to 3.
.
.
.
Elements (390-400) are mapped to 40.
Count each value from 1 to 40 and substitue the values of countung in feature 22
Start

Figure 3.8: Obtaining the elements of feature 23.

According to figure 3.8, a new vector labelled features 23 consisting of 40 elements of values zero is created. Moreover, through vector summation, D and S are summed, which results in a new vector called summation. The reason for selecting the distribution of the summed values of vectors D and S for feature number 23 is because after intensive experimental work of choosing the most optimal feature, this feature specifically was found to increase the accuracy of the classification in the next stage. The elements values of summation are quantised as shown in the above figure. Each of the quantised values are counted, and the result is positioned at each element in Feature 23. As mentioned earlier, the elements of Feature 23 are added to the vector of features. Feature 24 consists of the same number of elements as Feature 23, with these feature vectors nearly identical in element acquisition. The only difference is that, rather than quantising the summation of D and S, only the D vector is quantised in Feature 24. Furthermore, the 40 new elements of Feature 24 are added to the features vector, which is, at this point, comprises 102 elements. Lastly, Feature 25 consists of just 25 elements, where the maximum value of the elements is 100, and the rest are identical to Features 23 and 24. Figure 3.9 illustrates the whole feature vector in its final form:

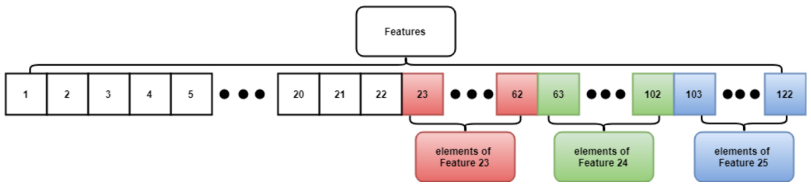


Figure 3.9: Final feature vector

3.4 Frequency domain

The frequency domain is used as a feature extraction method in the field of acoustic identification of species. For instance, Cheng et al. were able to identify a diverse dataset that consist of four species of passerine birds using probabilistic models, notably Mel-Frequency Cepstrum Coefficients (MFCCs) and the Gaussian mixture model (GMM) as audio features [68]. They achieved approximately 90% accuracy in classification. Similarly, Lee et al. implemented a technique on a broader dataset of birds, reaching an overall 84% classification accuracy on a 28 bird species [69]. The classification achieved 100% accuracy for several species, while it was significantly lower (less than 10%) in some cases. Such low accuracies were due to the similar frequency range of certain birds within the experimented data set. The following are a variety of the most commonly implemented techniques for extracting features based on the frequency domain.

3.4.1 Fast fourier transform (FFT)

The discrete Fourier transform (DFT) is applied to calculate the frequency component of a signal with a computational complexity of $O(N^2)$.

FFT is used to calculate DFT for a discrete signal, which has a complexity of $O(N \log N)$ [94]. Computing the FFT for a long signal is computationally expensive, and FFT assumes that a signal is stationary. Hence it should not be used for non-stationary signals. In order to solve this issue, an adjustment was made in FFT to produce a short-time Fourier transform (STFT). The latter is obtained by splitting the signal into small chunks, with each chunk assumed to be stationary. Additionally, a smoothing window, such as Hamming or Hanning, is applied to smooth out the signal near the end of each chunk and avoid a high-frequency response when FFT is implemented. The frequency components obtained using FFT are widely used as features in the classification of signals, not only in bioacoustics classification but also in the classification of signals in general [95].

3.4.2 Fast wavelet transform (FWT)

Wavelet is similar to FFT in transforming the signal into several sinusoids with different frequencies. However, it is implemented in terms of wavelet transforms. Moreover, wavelet transformation has a clear advantage over FFT in terms of conversion from time to frequency domain. For instance, the specific points in the time domain where it is not possible to obtain the exact frequency. Therefore, when transforming the signal from the time domain to the signal domain using FFT, temporal properties are eliminated due to the assumption that the signal is stationary. Therefore, STFT is used to solve this problem. However, when processing the signals in small windows, the window remains fixed regardless of any change in the geometry of the signal, which causes a loss of resolution. This disadvantage of Fourier transforms is related to the uncertainty principle detailed by Heisenberg in 1927, which states that either the direction or the speed of a particle can be determined, but not both. Wavelets solve this issue using dynamic resolution, depending on the frequency range upon which the signal is analysed [96]. In other words, this means having a dynamic window size for FFT; but with wavelets, the term scale is used rather than frequency. Although standard wavelets give an excellent resolution, there are still some data in the signal not being captured. Therefore, a complex wavelet transform was created to solve this disadvantage, taking into account the signal phase. Morlet wavelet is commonly used and based on Gaussian modulated with a sin wave carrier. Continuous wavelet transforms (CWT) is the method of scanning the signal with various scales of the mother wavelet [97]. Additionally, discrete wavelet transform (DWT) directly corresponds to CWT, as it mainly depends on the sampling of it. FWT is an efficient method of obtaining the DWT. As a result of the above, Wavelets have been used for audio classification [98].

3.4.3 Mel-frequency spectrum coefficients (MFCC)

Stevens, Volkman, and Newman (1937) refer to the relationship between the frequencies transmitted and the pitches perceived by the human ear as the Mel-scale [99]. The dimension of the vocal tract acts as a filter of the sounds produced by humans. If this dimension is calculated precisely, it would reflect on the accuracy of the phoneme being created. Moreover, a mathematical representation of the dimension of the vocal tract is the envelope of the short-time power spectrum, which is calculated by MFCC [100, 101]. The steps taken to compute the Mel-frequency coefficients are as follows:

- **Frame the signal into short frames:** An audio signal is in a constant state of change; presumably, the audio signal is low in magnitude under short time scale conditions, in other words, statistically stationary. Evidently, the elements are continuously varying on short time scales, therefore, the frame of the signal is altered to 20-40ms frames. Because of the short time frame, there are not any sufficient samples to extract robust spectral estimate, as when the signal is longer it constantly fluctuates along the frame [102].

- **For each frame calculate the periodogram estimate of the power spectrum:** Cochlea, which is a small organ in the human ear, inspired the power spectrum that is used to calculate each frame. Cochlea is vibrated at many positions depending on the incoming sound frequencies. Based on the frequency of vibration, several nerves are activated giving signs to the brain that specific frequencies are occurring. Furthermore, the periodogram estimates simulation is similar in order to classify the frequencies occurring in the frame [103]. There is valuable information delivered by the periodogram spectral estimate, as it can distinguish between two closely spaced frequencies. The effect gets more noticeable with any rise in frequencies [103].

- **Apply the Mel-filterbank to the power spectra, and sum the energy in each filter:** In order to compute energy at different frequencies, the bins of the periodogram are summed by the Mel-filterbank. The initial filter is narrower than the others so it provides an estimation of how much energy exists near the 0 Hz region. Then, the higher the frequencies get, the wider the filter becomes. The Mel scale calculates the width of the filterbanks and by how much they should be spaced from each other [100, 101].

- **Take the logarithm of all filterbank energies:** When the filterbank energies are obtained, the logarithms of each of their corresponding values are computed. This is inspired by human hearing due to humans not being able to hear loud sounds on a linear scale [103].

- **Take the DCT of the log filterbank energies:** DCT decorrelates the overlapping output of the log filterbank energies, hence diagonal covariance matrices can be used as features [102].

Finally, The Mel scale relates the perceived pitch or tune to its specific computed frequency. Moreover, the human ear is more sufficient at differentiating small changes in pitch at low frequencies than they are at high frequencies. Proposing this scale makes the features simulate better what humans hear. The relationship between the frequencies transmitted and the pitches perceived by the human ear [100, 101].

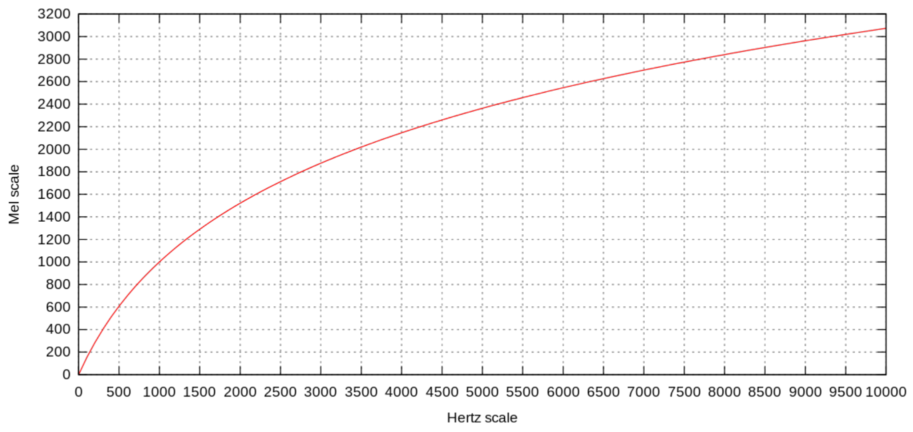


Figure 3.10: Mel-scale waveform [101]

The formula for converting from frequency to Mel scale is as follows:

$$M(f) = 1125ln(1 + \frac{f}{(700)}) \tag{3.5}$$

and from Mel scale to frequency is as follows:

$$M^-1_(m) = 700(\exp \frac{m}{1125} - 1) \tag{3.6}$$

3.4.4 Frequency domain method implemented for this project

The frequency domain-based technique to extract the features from the audio files in this project was implemented by Jaap Hatisma and Ton Kalker in 2010. The study sought to identify songs using the fingerprint of an unknown audio clip as a query on a fingerprint database. The latter contained the fingerprints of a vast library of songs, from which the audio clip was identified [104].

The following figure represents the code implemented by Jaap Hatisma and Ton Kalker to extract the features of the songs. However, it is applied here to the three-second clips of bee recordings.

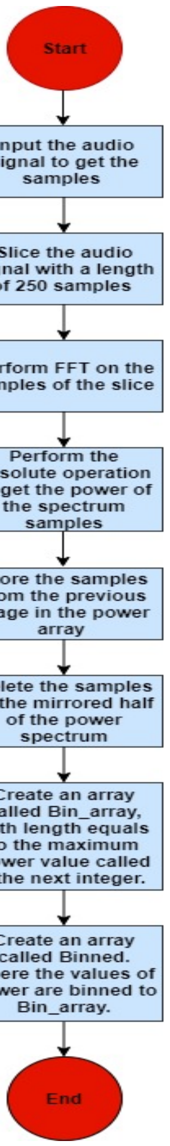

Start
Input the audio signal to get the samples
Slice the audio signal with a length of 250 samples
Perform FFT on the samples of the slice
Perform the absolute operation to get the power of the spectrum samples
Store the samples from the previous stage in the power array
Delete the samples of the mirrored half of the power spectrum
Create an array called Bin_array, with length equals to the maximum power value called to the next integer.
Create an array called Binned. where the values of power are binned to Bin_array.
End

Figure 3.11: Frequency based feature extraction algorithm

In the above flowchart, the first stage is to read the audio signal. Thus, the sample signals are obtained. Figure 3.12 illustrates an audio signal from the bumblebee species *Bombus hortorum*.

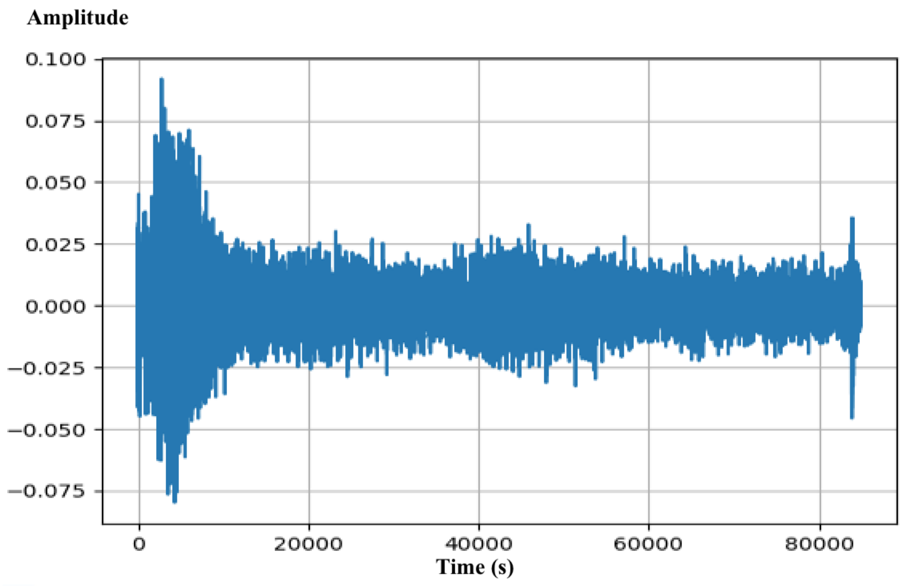


Figure 3.12: Audio signal of a bumblebee in the time domain

When the samples are obtained from the audio recording, it is sliced to a window size of 500 samples, as the figure 3.13 represents.

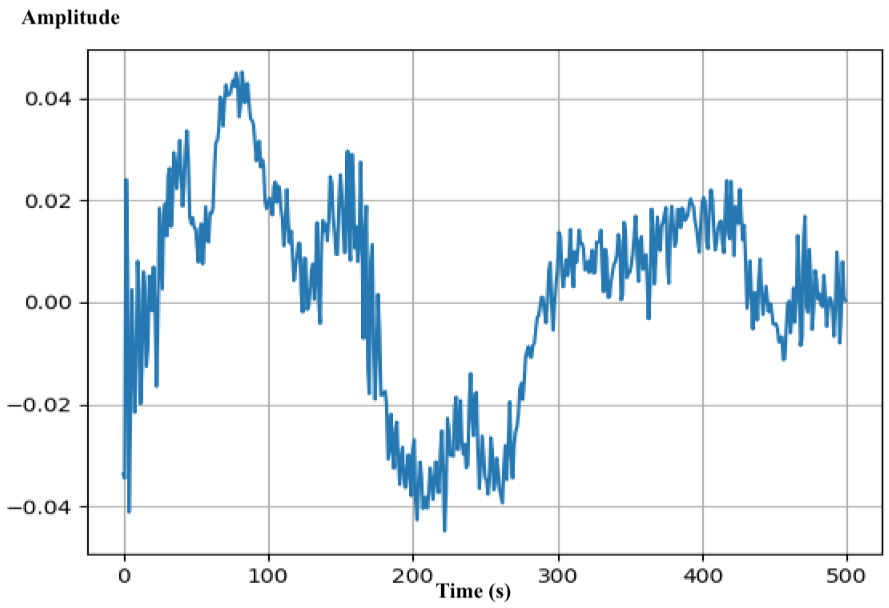


Figure 3.13: First slice of the audio signal

The FFT function is then applied to the slice to give the following graph. However, according to Jaap Hatisma and Ton Kalker, FFT is implemented on a window size of 3ms of the whole song. In this project, this is considered a significantly long window size because most of the species recordings are less than 371ms.

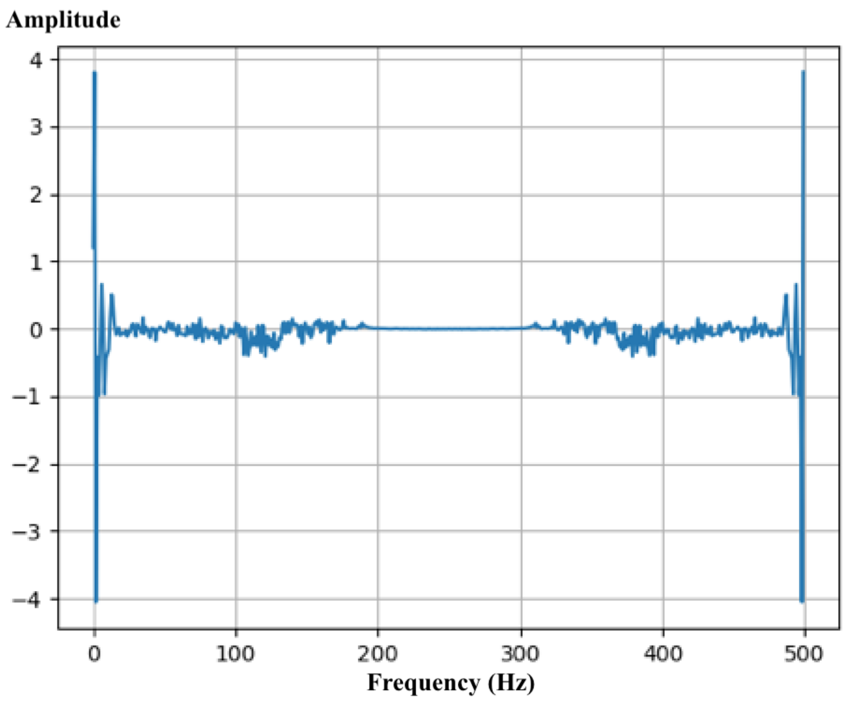


Figure 3.14: converting the audio signal to the frequency domain using FFT1.

The next stage is applying the absolute function on each of the samples obtained after applying FFT to get the power spectrum as shown in the next graph.

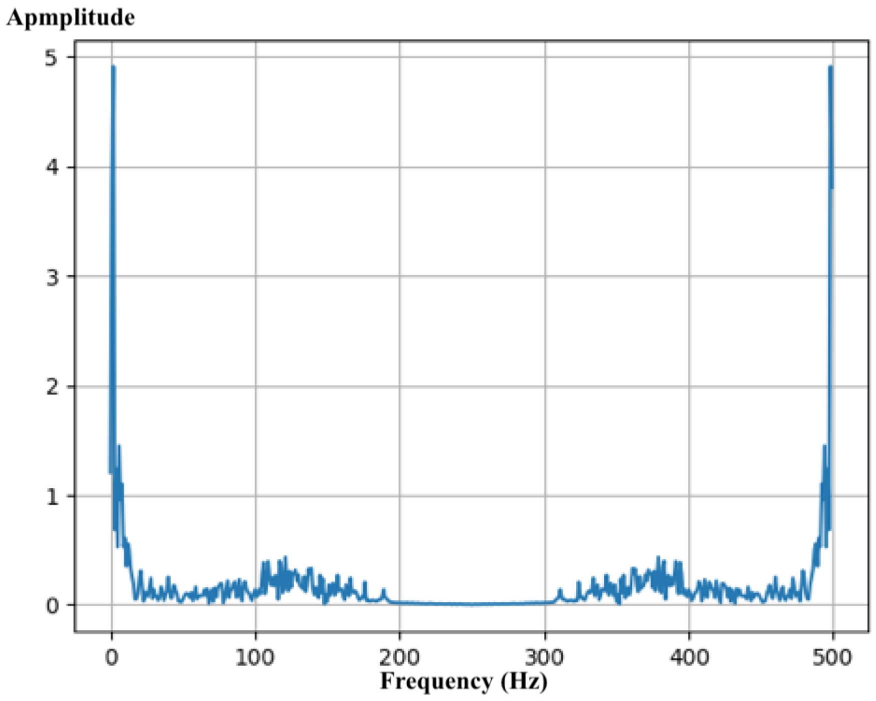


Figure 3.15: Power spectrum of the first slice of the audio signal

The samples used to represent the mirrored half of the power spectrum are deleted, and the power spectrum is binned. It is important to note that multiple power values that are closer to each other are binned together, thus obtaining some robustness to small changes (noise). Jaap Hatisma and Ton Kalker logarithmically compressed the power spectrum of their audio files between 318Hz and 2KHz to save space [104]. Because the audio files of the species are small in size, if they were to be compressed any further, this would lead to a loss of information within the files. Finally, the whole technique is repeated and applied to the next slice of the audio signal with an overlap of 50% with the previous one. This process continues until the audio signal reaches an end, and the result is 250 bins (features) for every audio recording.

3.5 Observation on the advantages of time-domain over the frequency domain

This project aims to implement a real-time embedded system to extract the features from the audio samples recorded and classifying each record. Therefore the less computational power used for the stage of feature extraction the better, as the matrices that will be developed in the classification stage are massive.

Moreover, extraction methods based on the frequency domain are generally established using fast Fourier transforms (FFT), wavelet transforms, and linear predictive coding (LPC) [33, 105–107]. However, such methods are computationally expensive. Therefore such analyses take a long time to perform, leading to high-cost system implementation [106]. However, the main advantage of frequency domain analysis, such as Fourier analysis, is that little information is lost from the signal during the transformation. Furthermore using Fourier transforms the information on amplitude, harmonics, and phase is all maintained, and it uses each part of the waveform to translate the signal into the frequency domain.

On the other hand, although time-domain signal coding uses only three parameters to describe the signal, which makes it less useful in applications, where the signal needs to be compressed while keeping as much of the information within it preserved as possible. The low computational complexity of TDSC has a significant advantage, as it is implementable on low power microcontrollers. Hence the possibility of hand-held recognizers and remotely sited long term bioacoustics monitoring. Therefore, TDSC based techniques fit the purpose of this project, as the main aim is to develop an embedded system capable of carrying out the complex calculations of the classification stage observed in the following chapter. Additionally, issues regarding the storage and battery life arising from the computational complexity of frequency domain-based techniques can be avoided using TDSC based techniques, which fits the aim of this project.

Furthermore, frequency domain methods are computationally expensive and challenging to implement on low-cost microcontroller-based systems. Indeed, in the remotely-sited PC-based system, which can be any 486 computer or better with a minimum of 4 Mbytes of RAM and a hard disc could only record for 75% of the time, the rest devoted to signal to process even though the FFT size was limited to 32 points [108]. The use of dedicated digital signal processors is also prohibitive in cost and power consumption (crucial for remotely sited systems).

Also, fast fourier transforms face issues with phase, where two different signals X1 and X2, are analysed in frequency and time domains [93]. The latter analysis illustrated that the frequency domain analysis was performed using the power spectrum of 512 points FFT, and the time domain analysis used the WASP program to encode the signal by TDSC. The power spectrum result showed that the signals had identical frequency components (50Hz, 120Hz, and 375Hz). Thus FFT analysis is incapable of identifying X1 and X2. TDSC appears to be more conclusive, as different distributions of the codes were given for the signals [93].

Furthermore, the novel approach for extracting the features from audio signals S-TDSC, which, compared to other time-domain based techniques, does not require looking up codebooks or providing a constant code for any D-S pairs when exceeding a particular value. Furthermore, statistical time-domain signal coding time takes less time to be implemented than the frequency domain-based techniques. However, what made S-TDSC the most suitable feature extraction technique for this project is that it is computationally less expensive than frequency-domain techniques. The latter is due to the simple statistical operations used in S-TDSC compared to the implementation of FFT and the binning of the power spectrum samples.

Finally, the computational complexity analysis is carried out using MacBook Pro, which has a processor of 2.6 GHz Intel Core i5, a memory of 8GB 1600 MHz DDR3 and the graphic card of Intel Iris 1536 MB. The computational complexity analysis is conducted through the Python functions of time() to carry out the time taken to execute the codes of S-TDSC and the frequency domain-based method. Additionally, the Python function memorymeasure() is used to compute the memory consumption. Furthermore, all of the subsequent computational complexity analyses in this thesis are carried out using the same computer and Python functions. Based on a computational complexity analysis for the frequency domain-based method and S-TDSC, it is concluded that the time taken to execute S-TDSC code for all the audio files in the dataset is 2 minutes and 30 seconds. On the other hand, the time taken to implement the frequency domain-based method to extract the features from the audio files in the dataset is 4 minutes and 3 seconds. Hence, there is an increase in the time taken to execute the frequency domain based method by 87% compared to the time taken to execute the S-TDSC code. Additionally, in terms of memory consumption, the resident set size memory consumed to perform S-TDSC, which is the non-swapped physical memory a process has used, is 111.84128MB compared to 114.08MB of the resident set size memory consumed to perform the frequency domain based method. Moreover, when implementing S-TDSC, the amount of virtual memory size consumed, which is space on the hard drive that is allocated by the operating system (ios) to be used as a supplemental reserve of memory when the RAM of the software has reached its maximum capacity is 419.84MB. Whereas, the amount of virtual memory size consumed by the frequency domain method is 423.02MB. As a result, for the feature extraction stage in this project, S-TDSC has proven to be less computationally expensive especially in terms of the time taken to execute the codes, than the frequency domain based method. However in terms of memory consumption both methods are roughly the same. Therefore S-TDSC is chosen as the feature extraction method used for both chapters 4 and 5.

Chapter 4

Classification

The classification problem of data has been widely investigated, and several algorithms have been devised that are mainly categorized, based on the learning method they use, into unsupervised, semi-supervised, and supervised methods.

Unsupervised methods try to cluster the data into clusters without any prior information about the data. A crucial input for some algorithms though, is the number of clusters that the data need to be split among. In supervised methods, data used for the training phase are labelled, and the training algorithms try to decrease iteratively the error between the ground truth and the output of the network until convergence. Semi-supervised methods grab the learning characteristics of both unsupervised and supervised algorithms. They are generative methods that learn the structure of the features without supervision, and then, they can use further supervised training to operate discriminatively.

Support vector machines (SVM), random forests, which are widely used classification methods for sound, in addition to recently adopted methods, such as back-propagation and extreme learning machines (ELM), are investigated in this chapter aiming for finding the optimum method for the classification problem in this research.

4.1 Support vector machine (SVM)

In the early 1960s, Vladimer Vapnik introduced support vector machines to solve binary classification problems [109]. Vapnik's approach begins with the assumption that there is a space with negative and positive samples, as shown in figure (4.1). Vapnik then poses the question of how to divide these samples by a straight line (indicated here in red), with the widest distance between the samples representing the decision boundaries. This distnace is indicated in the following figure by the space between the yellow lines, which are the 'support vectors'.

4.1.1 Derivation of binary support vector machine (SVM)

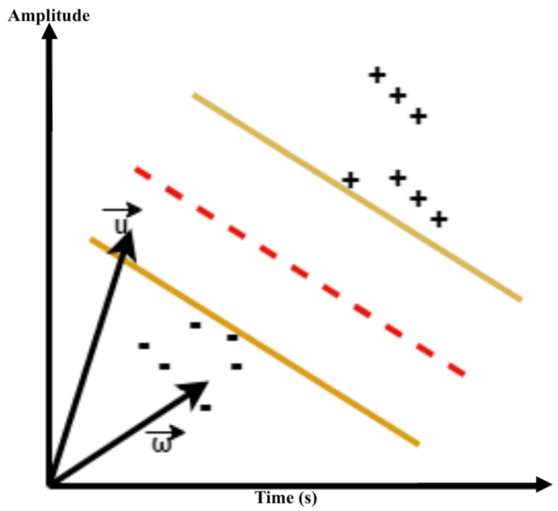


Figure 4.1: Space with positive and negative samples separated by the widest area.

Vapnik then introduced an unidentified length vector ω that is perpendicular to the median of the decision boundaries and another unknown point denoted by u, with a vector pointing to it. Here, the main concern is whether this unknown point is on the positive side of the boundaries or the left side, so vector u is projected to the vector, perpendicular to the decision boundaries ω; hence, the distance from the same direction as the latter. It is therefore represented by the vector ω, dotted with the vector u, to measure whether the result is equal to or greater than a constant C. This is represented by the following equation:

$$\vec{w}.\vec{u} \geq c \tag{4.1}$$

Alternatively, it is better represented by the decision rule:

$$\vec{w}.\vec{u} + b \geq c \tag{4.2}$$

Where:

$$c = -b \tag{4.3}$$

If the decision rule equation is true, then the sample is positive. Until this moment, the values of constant b and the vector ω remain undetermined; therefore, more constraints must be provided for ω and b to be calculated by applying the decision rule to positive and negative samples, as follows:

$$\vec{w}.\vec{X_+} + b \geq 1 \tag{4.4}$$

$$\vec{w}.\vec{X_-} + b \leq -1 \tag{4.5}$$

For mathematical convenience, the variable y_i , equals +1 for positive samples and -1 for negative samples. Therefore, when this variable is multiplied using the above two equations, the results are represented by the following equation:

$$y_i(\vec{w}.\vec{X_+}) - 1 \geq 0 \tag{4.6}$$

If the sample is within the yellow lines (in Figure 4.1), then the equation is illustrated as follows:

$$y_i(\vec{w}.\vec{X_+}) - 1 = 0 \tag{4.7}$$

The main aim of the SVM is to establish the longest distance between the different samples in the space, this being the width between the yellow lines. Figure (4.2) illustrates a vector X+, a vector X-, and a vector that represents the subtraction of X+ and X-.

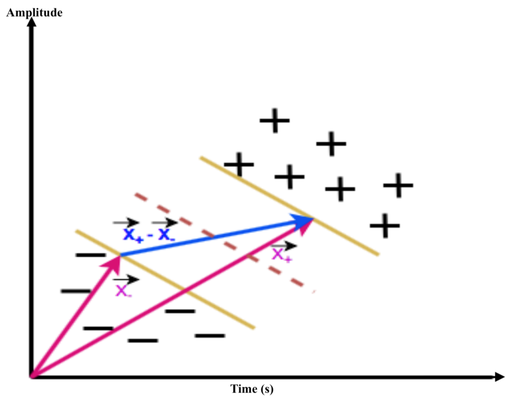


Figure 4.2: Obtaining the width of the separating space using the support vectors

To obtain the width between the yellow lines, the subtraction of vectors X+ and X- is dotted by the unit vector, which is represented as follows:

$$Width = (X_+ - X_-).\frac{\vec{w}}{|w|} \tag{4.8}$$

$\frac{\vec{w}}{|w|}$is the unit vector as $\vec{w}$ is a normal vector.

From equation (4.7), assuming that the sample is positive, then $y_i = 1$ and $X_+ = 1 - b$; and if the sample is negative, then $X_- = 1 + b$. Therefore, when X_+ and X_- are both substituted in the width equation, the result is as follows:

$$Width = \frac{2}{|w|} \tag{4.9}$$

Since the objective is to maximise $\frac{2}{|w|}$, this allows the maximisation of either $\frac{1}{|w|}$ by dropping the constant, or the minimisation of $|w|$ or $\frac{1}{2}|w|^2$ The problem now is thus maximising the width, which has a constraint represented by equation (4.7). The Lagrange multipliers are used to achieve this, as follows:

$$L = \frac{1}{2}|\vec{w}|^2 - \sum a_i[y_i(\vec{w}.\vec{x} + b) - 1] \tag{4.10}$$

Where:

- a_i is the Lagrange multiplier.

To find the extremum of the above equation, the derivative with respect to the variables is obtained and then set to 0. Firstly, derivation with respect to $\vec{w}$ is expressed as follows:

$$\frac{2L}{\partial \vec{w}} = \vec{w} - \sum a_i y_i x_i = 0 \tag{4.11}$$

Therefore:

$$\vec{w} = \sum a_i y_i \vec{x_i} \tag{4.12}$$

This means that the vector w is a linear sum of the samples a_i, y_i and $\vec{x_i}$.

$$\frac{2L}{\partial b} = -\sum a_i y_i = 0 \tag{4.13}$$

Substituting the value of $\vec{w}$ into equation (4.10), the result is:

$$L = \frac{1}{2}((\sum a_i y_i \vec{x_i})(\sum a_i y_i \vec{x_j}) - ((\sum a_i y_i \vec{x_i})(\sum a_i y_i \vec{x_j}) - b(\sum a_i y_i) + (\sum \alpha_i) \tag{4.14}$$

Since:

$$-\sum a_i y_i = 0 \tag{4.15}$$

Therefore:

$$L = \sum a_i - \frac{1}{2} \sum_i \sum_j a_i a_j y_i y_j x_i . x_j \tag{4.16}$$

Finally, substituting $\vec{w}$ into the decision rule gives the following for a positive sample:

$$\sum a_i y_i \vec{x_i} . \vec{u} + b \geq 0 \tag{4.17}$$

4.1.2 Non-linearly separable data

However, up to this point, SVM can separate the linearly separable data. However, the data are not necessarily linearly separable, as the following graph illustrates.

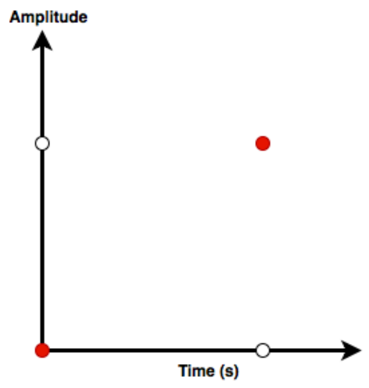


Figure 4.3: Non-linearly separable data

This is easy to resolve another axis is added in a different space, as shown in figure (4.4)

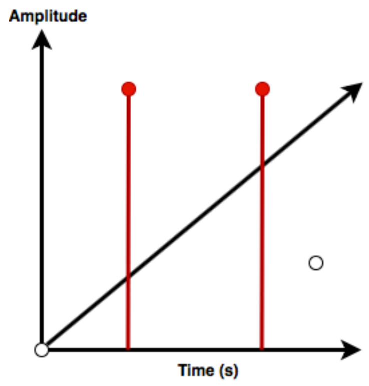


Figure 4.4: Establishment of another axis with which to project the data

Therefore, a transformation in space is required, where the points are taken from the original space to a new space, as shown in the above figure. This transformation is referred to as $\phi(\vec{X})$. Since the maximization depends solely on the dot products, all that is needed is the transformation of one vector dotted with the transformation of another. Therefore, the following must be maximised:

$$\phi(\vec{x_i}).\phi(\vec{x_j}) \tag{4.18}$$

To recognise the group a sample belongs to, the transformation of the dot product is needed:

$$\phi(\vec{x_i}).\phi(\vec{u}) \tag{4.19}$$

This is achieved through the Kernel function, which provides the dot product of these vectors in another space, without needing to know the transformation in the other space. Furthermore, the principle of the Kernel function is to map the samples into a new space, then to find the hyperplane that best separates the data, and finally to map the hyperplane back to the original space. This is represented by the following graph.

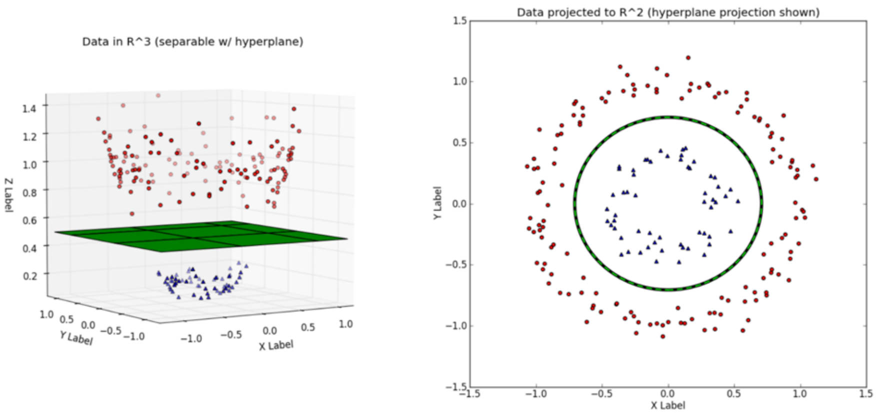


Figure 4.5: Hyperplane separating the different samples [107].

SVM has been used for sound signal classification. Low power consumption by SVM and Kernels in sound detection event monitoring is observed [110, 111]. Furthermore, SVM is a classification technique used for this project to identify four types of bees based on their flight sounds.

4.1.3 Experimental results of SVM

. Table (4.1) shows the classification accuracy of four species, as obtained using SVM. The species are bumblebees, common wasps, hoverflies, and solitary bees, with every species stored in a separate file and having 69, 21, 42, and 39 recordings, respectively. However, the following is taken into account:

Firstly the features are extracted from the audio files and stored, and then cross-validation is performed on the features dataset. Typically cross-validation is done through randomly shuffling the features dataset, and then the data is split into k groups. The overall process is as follows:

1. the dataset randomly must be shuffled randomly then split into k clusters.
2. Then for each unique cluster:
   - assign a certain cluster as testing dataset.
   - assign the cluster that are left as training dataset.
   - Implement a model on the training dataset and evaluate it based on the the testing dataset.
   - Keep the testing score.
3. Summarize the how model performed using the sample of model evaluation scores. However, k is equal to 2 in the case of the experiments carried out in this project due to the limit of the recordings available to train the models. Furthermore, one of these groups is used as training, and the other used as testing. Finally, the size of the group chosen to train the model out of the whole dataset is increased in steps of 10% to show how can more data available for training can affect the classification accuracy.

The following table shows how the proportion of the files used for training begins at 50% and increases in steps of 10 and illustrates the accuracy of the testing results at each step is an average of seven runs.

Proportion of files used for training	S-TDSC accuracy	Frequency domain method accuracy
50%	46.14%	33.5%
60%	44.7%	35.5%
70%	45.28%	34.4%
80%	50.28%	36.7%
90%	51.1%	32.42%

Table 4.1: Performance of the support vector machine (SVM) classifier after extracting the features using S-TDSC and the frequency domain methods

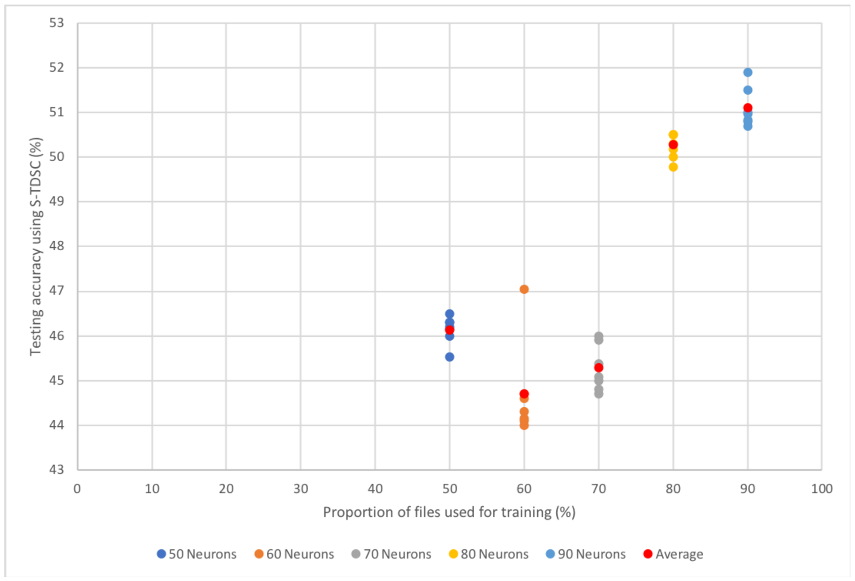


Figure 4.6: The accuracy of each of the 7 runs for each proportion of training files using S-TDSC

The following chart illustrates the dominance in terms of accuracy of S-TDSC as a feature extraction technique, compared to the frequency domain method. It is noted that accuracy sometimes decreases, even as the amount of training data increases.

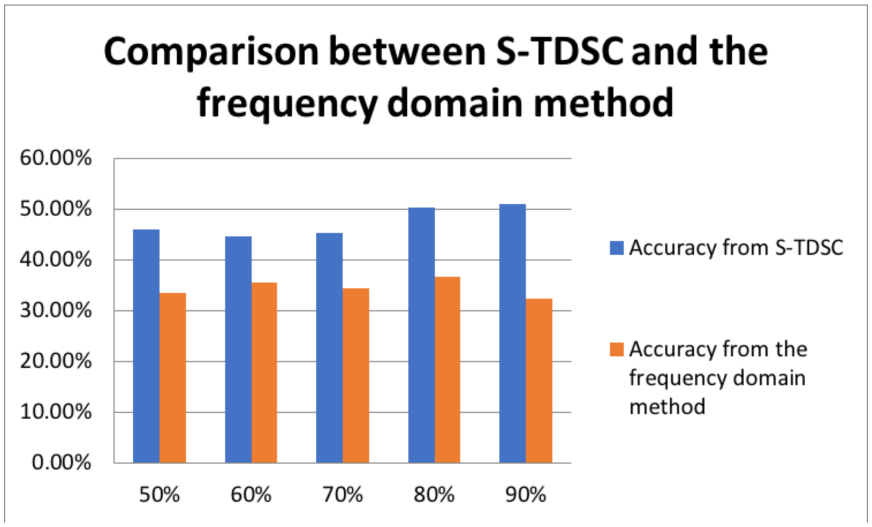


Figure 4.7: Comparing the performances of S-TDSC and the frequency domain method as feature extraction techniques

Confusion matrices are obtained using S-TDSC as a feature extraction technique. Confusion matrices describe the performance of a classification model on a set of test data for which the correct values are established and known. For instance, When 50% of the files are used for training, the run that gave the maximum accuracy out of the seven runs was 52%, and when the accuracies of the seven runs were averaged the accuracy was 46.14%, as table 4.1 illustrates. The following confusion matrix shows that common wasp was predicted correctly by the classifier two times and has been mispredicted by the classifier two times as bumblebee, seven times as a hoverfly, and three times as solitary bee. Whereas, solitary bees were predicted correctly 26 times and mispredicted five times as common wasps, four times as bumblebees, and seven times as hoverflies. It can also be noticed that the most crucial set of numbers in any confusion matrix is the set the runs diagonally through it as it is the set that represents where the classifier got all the classes correctly.

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	2	5	1	1
Solitary bees	3	26	4	2
Bumblebees	2	4	10	20
Hover flies	7	7	2	7

Table 4.2: Table 4.2

More confusion matrices of when 60%, 70%, 80%, and 90% of the audio files are used to train the support vector machine (SVM) model using both S-TDSC and the frequency domain methods are shown in Appendix A.

In order to validate the performance of SVM a dummy dataset is used for the algorithm to be implemented on. This dataset is imported from a Python library called SciketLearn, and this dataset consists of 1797 images, and the size of all the images is 8×8 pixel . Each image, as the following figure illustrates, is of a hand-written digit, the digits are from 0 to 9 and each digits is written by 180 people. This dataset is highly used within the machine learning community as a method to validate the result of self coded algorithm.

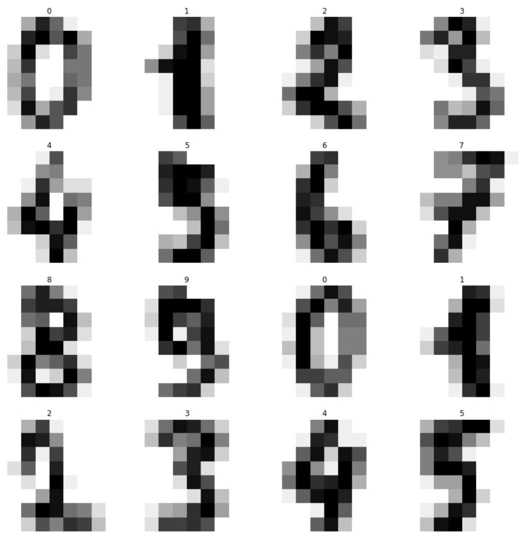


Figure 4.8: Samples of the hand-written digits dataset

Since each digit consists of 8×8 pixels, the grey scale of each pixel is considered as a feature of its own, then all of these features are put together in one feature vector that is 64 samples long. Furthermore, each vector is then fed to the SVM algorithm and the average accuracy, when 70% of this data is used for training and 30% is used for testing is 97.96% accurate. The confusion matrix of the run that gave the best accuracy is as follows:

	Target									
Predicted	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	50	0	0	0	0	0	0	0	0	0
1	0	54	0	0	0	0	0	0	1	0
2	0	1	54	0	0	0	0	0	0	0
3	0	0	0	56	0	0	0	0	0	0
4	0	1	0	0	43	0	0	0	0	0
5	0	0	0	0	0	51	0	0	0	0
6	0	0	0	0	0	0	62	0	0	0
7	0	0	0	0	0	0	0	58	0	0
8	0	2	1	2	1	0	0	0	54	0
9	0	0	1	0	1	0	0	0	0	47

Table 4.3: Confusion matrix of the hand-written digits dataset resulting from SVM algorithm

4.2 Random forests (RFs)

4.2.1 Earlier development of random forests (RFs)

In 1995, a method was proposed to resolve the complexity of decision tree classification using basic methods [112]. The proposed method uses the convenience of the oblique decision trees to optimise the training set accuracy. It builds several trees in randomly shuffled subsets of the feature space. The classification of each tree is generalised in a complementary manner, and the combined classification can be monotonically enhanced.

In 1998, Ho made another proposition to resolve the conflict between overfitting, which is the case where the overall cost is significantly tiny, but the generalization of the model is insufficient. This is because the model has learnt “too much” from the training data set [113]. This involved constructing a classifier using decision trees that maintained the highest accuracy in the training data while improving generalisation as the accuracy increases. This classifier is made up of many trees, established systematically by pseudo-randomly choosing subsets of components of the feature vector space; that is, several trees constructed in randomly chosen subsets. The next section presents random forests (RFs) as a classification technique.

4.2.2 Random forests (RF)

Random forests is an ensemble method used for classification and regression. In 1996, Breiman combined his bagging sampling approach with a random selection of features introduced by Ho [112,113] and Amit and Geman [114] to construct decision trees. In order to understand random forests, it is crucial to observe the concept of a decision tree first as random forest is number of decision trees added together. Using the observation of Zhou V. [115, 116], random forests are observed as shown in the dataset of the fallowing example.

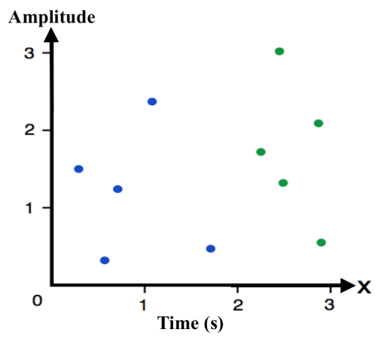


Figure 4.9: Samples of dataset [113]

Suppose testing the decision tree at a value of 2, thus the resultant tree would be as illustrated in figure 4.10.

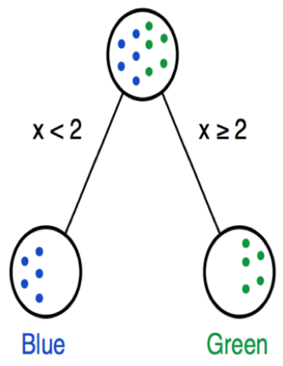


Figure 4.10: Decision tree with two branches [113]

This is a simple decision tree with one decision node that tests $x < 2$, if the test passes ($x < 2$), the decision would go down the left branch of the tree and the colour picked is blue. However, If the test fails $x \geq 2$, the decision would go down the right branch of the tree and the colour picked is green. Hence the dataset with decision split is shown as follows:

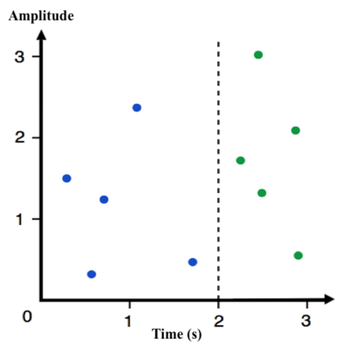


Figure 4.11: Plot of the decision boundary in the dataset [113]

On the other hand, considering another example of a dataset with three different classes as the following graph illustrates:

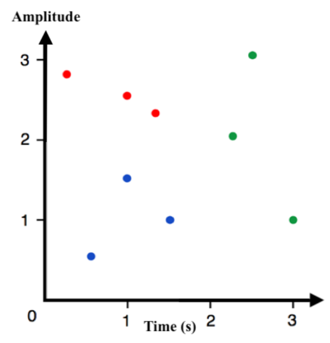


Figure 4.12: Dataset of three different classes [113]

The decision tree in figure (4.10) can not be used as the only classification option when a new point (x,y) is introduced to be classified. So if $x \geq 2$, the point is classified as green, on the other hand if $x < 2$ the data point is classified as red or blue. Therefore another decision node is added to the decision tree as follows:

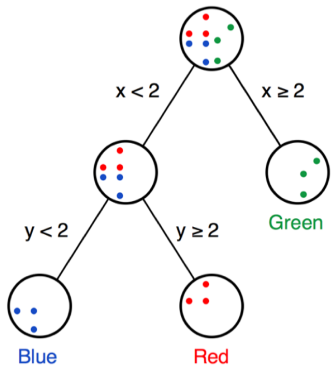


Figure 4.13: Decision tree with two cascaded splits each of value 2 [113]

Hence the dataset with the decision split is shown as follows:

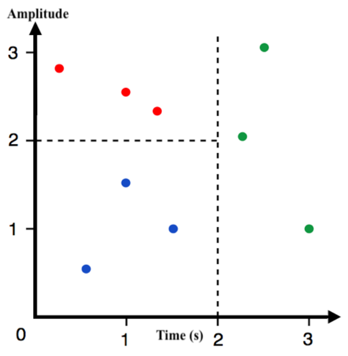


Figure 4.14: Plot of the decision boundaries in the dataset [113]

When training a decision tree, the first task is to calculate the root decision node in the tree. For instance, the root node in the following tree used the x feature with a minimum threshold of 2.

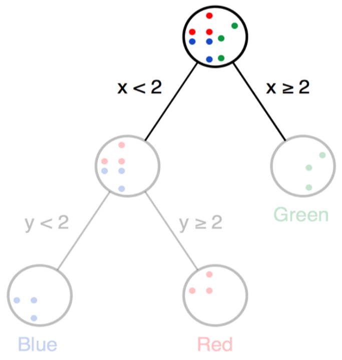


Figure 4.15: A decision tree with a root decision node of 2 [113]

The decision node that computes the most optimum split is considered a node that separates the classes as much as possible, as the graphs illustrate all the green samples are on the right and no green samples on the left. Additionally, a method called Gini Impurity is a technique that quantifies how good a split is [115, 116]. This method is observed in the following sub-section.

4.2.3 Gini impurity

Based on the dataset in figure (4.9) can be split at x = 2 and x=1.5 as figures (4.16 and 4.17)

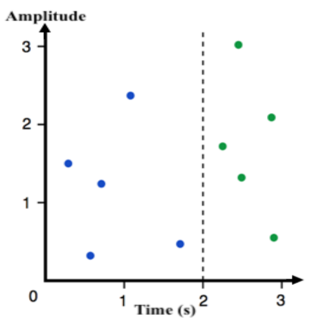


Figure 4.16: A dataset with a perfect split [113]

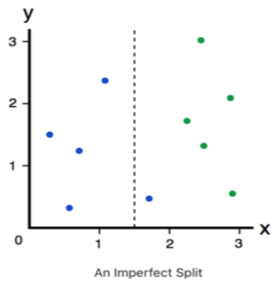


Figure 4.17: A dataset with an imperfect split [113]

The perfect split breaks the dataset into two branches, where the left branch consists of 5 blue samples and the right branch consists of 5 green samples. On the other hand, if a split is made at x = 1.5, the left branch would consist of 4 blue samples and the right branch would consist of 1 blue and 5 green samples. Therefore, it is clear that the split at x = 1.5 is worse than the split at x = 2. Furthermore, measuring the quality of the split becomes more important if a third class, which is red sample is added to the dataset. Suppose the following split:

- Branch 1, with 3 blues, 1 green, and 1 red .....
- Branch 2, with 3 greens and 1 red ....

Now compare the previous split with the next split:

- Branch 1, with 3 blues, 1 green, and 2 red ......
- Branch 2, with 3 greens ...

It's difficult to determine if these splits give the best split. Therefore, the Gini Impurity provides a qualitative approach of how good any split is and the following are examples of the Gini Impurities for the whole data set, the perfect split and the imperfect split.

- Whole data set:

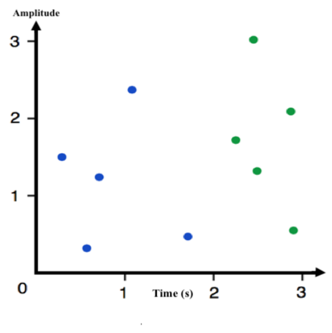


Figure 4.18: Data set of two classes [113]

According to the dataset in figure (4.18) if a random data point is picked, its either 50% blue or 50% green, now the data point is randomly classified according to the class distribution and since there is 5 data point of each colour, its classified as green 50% of the time and clue 50% of the time. Additionally, the next table shows probabilities of incorrectly classifying a data point.

Event	Probability
Pick Blue, Classify Blue (correctly)	25%
Pick Blue, Classify Green (incorrectly)	25%
Pick Green, Classify Blue (incorrectly)	25%
Pick Green, Classify Green (incorrectly)	25%

Table 4.4: The probability of incorrectly classifying a datapoint [113]

Thus, in a couple of events the data point was incorrectly classified. Therefore the total probability is 25% + 25% = 50%, so the Gini Impurity is

0.5. The formula to calculate the Gini Impurity is as follows:

$$G = \sum_{i=1}^{C} p(i) \times (1 - p(i)) \tag{4.20}$$

Where:

- C is the total number of the classes.
- $p(i)$ is the probability of picking a data point with class.

Therefore based on the above example, where $C = 2$ and $p(1) = p(2) = 0.5$, Therefore:

$$G = p(1) \times (1 - p(1)) + p(2) \times (1 - p(2)) \tag{4.21}$$

$$G = 0.5 \times (1 - 0.5) + 0.5 \times (1 - 0.5)$$

Hence, the value of the Gini Impurity from the formula and the Gini Impurity based on the result obtained from the table (4.4) match.

- Perfect data set:

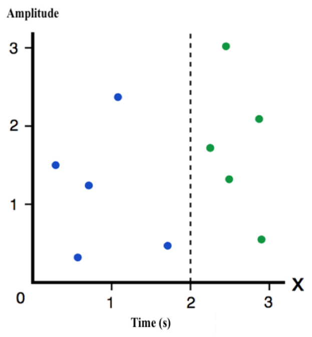


Figure 4.19: Dataset with a decision boundary of 2 [113]

Since the left branch consists of only blue samples, Gini Impurity is calculated as follows:

$$G_left = 1 \times (1 - 1) + 0 \times (1 - 0)$$

The right branch consists of green samples, therefore the Gini Impurity = 0 also. Therefore the perfect data split turned a dataset with a 0.5 impurity into 2 branches with 0 impurity.

- Imperfect data set:

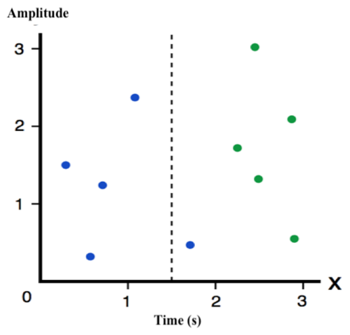


Figure 4.20: Dataset with a decision boundary of 1.5 [113]

Since the left branch of the split consists of only blue samples, the impurity value = 0. However, the right branch has five green samples and one blue. Therefore the impurity value is calculated as follows:

$$Gright = \frac{1}{6} \times (1 - \frac{1}{6}) + \frac{5}{6} \times (1 - \frac{5}{6})) = 0.278$$

So in order to quantitatively quantify how optimum the split is based on the Gini Impurity values of before the split, the left branch and the right branch, which were equal to 0.5, 0, and 0.278 respectively. Furthermore, the quality of the split is measured by weighing the impurity of each branch by how many elements it consists of. Therefore, since the left branch consists of 4 samples and the right branch consists of 6, the result is:

$$(0.4 \times 6) + (0.6 \times 0.278) = 0.167$$

Therefore the amount of impurity removed from the split is calculated through the following equation and called the Gini gain:

$$0.5 - 0.167 = 0.33$$

As a result, when training a decision tree, the best split is chosen by maximizing the Gini Gain, which is calculated by subtracting the weighted impurities of the branches from the original impurity. At this point the Gini gain is observed, back to this example illustrated in the following graph.

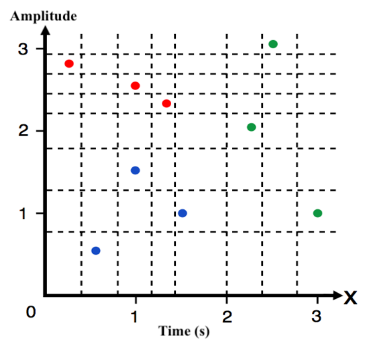
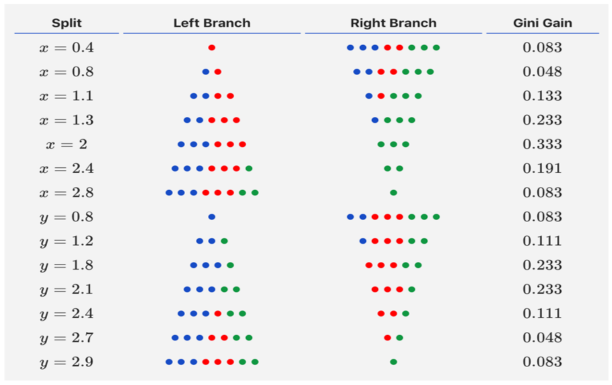


Figure 4.21: Dataset where each of the smaples is saperated based on different values of x and y [113]

And through calculating the Gini gain for every possible split as follows:



Split	Left Branch	Right Branch	Gini Gain
$x = 0.4$	🔴	🔵🔵🔵🔴🔴🟢🟢🟢	0.083
$x = 0.8$	🔵🔴	🔵🔵🔴🔴🟢🟢🟢	0.048
$x = 1.1$	🔵🔵🔴🔴	🔵🔴🟢🟢🟢	0.133
$x = 1.3$	🔵🔵🔴🔴🔴	🔵🟢🟢🟢	0.233
$x = 2$	🔵🔵🔵🔴🔴🔴	🟢🟢🟢	0.333
$x = 2.4$	🔵🔵🔵🔴🔴🔴🟢	🟢🟢	0.191
$x = 2.8$	🔵🔵🔵🔴🔴🔴🟢🟢	🟢	0.083
$y = 0.8$	🔵	🔵🔵🔴🔴🔴🟢🟢🟢	0.083
$y = 1.2$	🔵🔵🟢	🔵🔴🔴🔴🟢🟢	0.111
$y = 1.8$	🔵🔵🔵🟢	🔴🔴🔴🟢🟢	0.233
$y = 2.1$	🔵🔵🔵🟢🟢	🔴🔴🔴🟢	0.233
$y = 2.4$	🔵🔵🔵🔴🟢🟢	🔴🔴🟢	0.111
$y = 2.7$	🔵🔵🔵🔴🔴🟢🟢	🔴🟢	0.048
$y = 2.9$	🔵🔵🔵🔴🔴🔴🟢🟢	🟢	0.083

Figure 4.22: RF14 [113]

It is clear that the highest Gini Gain values is obtained when $x = 2$, therefore the decision node is chosen accordingly.

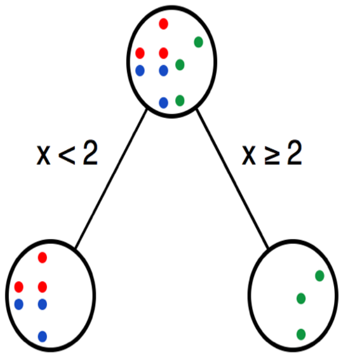


Figure 4.23: Decesion tree with a decesion bountry at a value of x = 2 [113]

However, in order for the decision tree to be completely trained, there needs to be a second node created for the left branch of the tree. Therefore, when every possible split for the 6 data points its realized that y=2 is the best split.

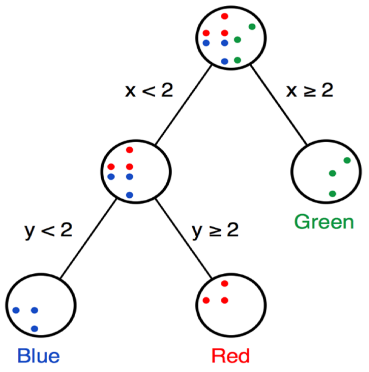


Figure 4.24: A perfect split decision tree that has a decision boundaries of x = 2 and y = 2 [113]

At this point, the decision tree can't be improved any further, therefore each final node is called a leaf node and labelled green, red and blue. To conclude random forest is a number of decision trees bundled together with the aid of bagging approach.

Bagging

In order to observe bagging, the following algorithm needs observed to train a bundle of decision trees given a data set of n points:

1. Sample, with replacement, n training examples from the dataset.
2. A decision tree needs to be trained on samples of number n .
3. loop for a number of τ.

To achieve a decision for a number of τ trees, the predictions are collected from the decision trees individually, then either:

- In case the decision trees produce labels of classes, for example colours, then the class with the majority of votes wins.
- In case the decision trees produce numerical values, for example product prices, then take the average of the trees.

This technique is called bagging as the following figure illustrates:

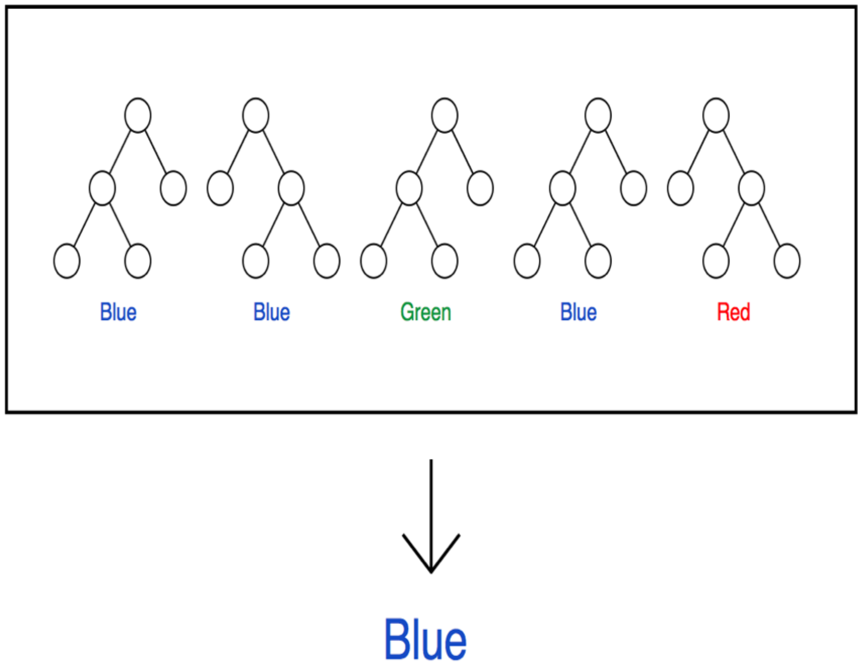


Figure 4.25: Illustration of the bagging approach [113]

Suppose there is a data set with p features. Instead of trying all features every time a new decision node is made, only try a subset of the features is tried, usually of size square root of p or $\frac{p}{3}$. This is primarily done to introduce randomness making individual trees more unique and decrease correlation between trees, which enhance the forest's performance in genral. This method is sometimes referred to in litreture as feature bagging [116].

The first paper on random forests [117] notes that the main factors affecting the error rate of RF are correlation and strength. In other words, the strength of individual trees is indirectly proportional to the error rate of random forest, and the correlation is directly proportional to the error rate. These findings seem to support the study that found the error rate is statistically decreased by simultaneously maximising strength and minimising correlation [116]

4.2.4 Experimental results of random forests

Taking the same example of the four species used in the SVM, RF is applied as classification algorithm. The accuracy results using both feature extraction techniques (S-TDSC and the frequency domain method) are illustrated in the following table:

Proportion of files used for training	S-TDSC accuracy	Frequency domain method accuracy
50%	49.7%	43.7%
60%	51.8%	41.4%
70%	53.5%	41.0%
80%	56.8%	45.4%
90%	54.5%	52.5%

Table 4.5: The accuracy of random forests (RFs) using S-TDSC and the frequency domain method as feature extraction techniques

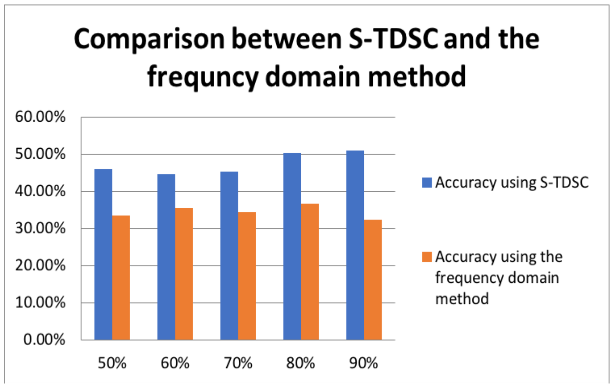


Figure 4.26: Comparison between the performances of random forests using S-TDSC and the frequency domain method

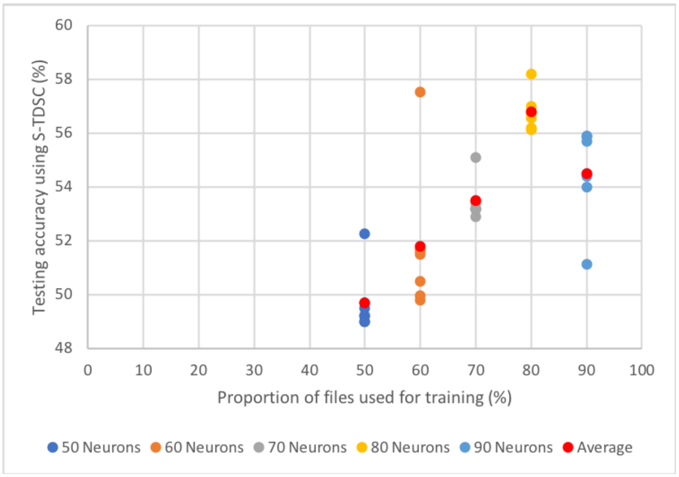


Figure 4.27: The accuracy of each of the 7 runs for each proportion of training files using S-TDSC

The above chart shows that S-TDSC has greater accuracy than the frequency domain method, and the difference is approximately 10%, which is relatively large.

The following are confusion matrixes obtained when using S-TDSC as a feature extraction technique. When 50% of the files are used for training, the maximum accuracy of the specific run is 51%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	10	0	6	3
Solitary bees	3	0	5	4
Bumblebees	8	2	22	3
Hover flies	0	0	7	12

Table 4.6: Table 4.6

More confusion matrices of when 60%, 70%, 80% and 90% of the audio files are used to train the random forest model using both S-TDSC and the frequency domain methods are shown in Appendix B.

Similarly with SVM, due to the lack of audio files to train the algorithms with, the code made for the classification stage using random forests algorithm is verified using the hand-written digits dataset discussed in 4.1.3. There when 70% of the data is used for training the average testing accuracy of the 7 runs is 93.14% and the confusion matrix that gave the best accuracy out of these 7 runs is illustrated by the following table

	Target									
Predicted	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	59	0	0	0	1	0	0	0	0	0
1	0	44	1	2	0	0	0	0	1	0
2	0	2	51	1	0	0	0	0	0	0
3	0	0	0	45	0	0	0	0	1	2
4	0	1	0	0	57	0	0	1	0	1
5	0	0	1	0	2	39	0	0	0	1
6	1	0	0	0	0	0	60	0	0	0
7	1	0	1	1	0	0	0	54	1	1
8	1	2	1	0	0	1	1	0	45	3
9	0	0	0	0	1	0	0	0	2	49

Table 4.7: Confusion matrix of the hand-written digits dataset resulting from random forests algorithm

4.3 Artificial neural networks (ANN)

ANN are mathematical models that comprise the interconnections of simple processing units (nodes), each of which respond to an input signal with predefined responses. The basic principle of the ANN is that the nodes are designed to simulate the biological neurons: in other words, when a certain input signal is applied to the neural network, a predictable output response is obtained. Moreover, such nodes can learn from errors and alter the output of the network to decrease errors through a process of 'training'. When neurons are interconnected, they acquire the processing power to solve linear and non-linear problems with better accuracy than statistical methods [93].

4.3.1 Observation on the operation of perceptron

The following figure illustrates a perceptron, which is also known as a node. The latter is considered the essential unit of the ANN and represents a model of a biological neuron. The latter has a scaler input p, which has a scaler weight w and a bias b. The bias is a scaler value in the node itself and not an extra input, typically driven by a constant input value of 1 [118].

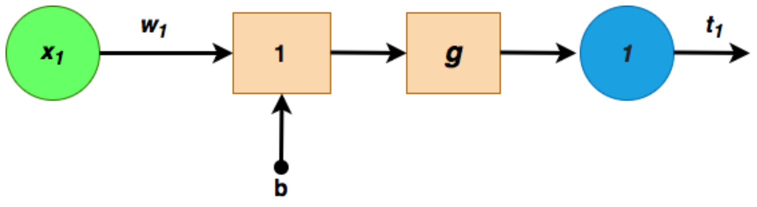


Figure 4.28: Structure of the perceptron with a single input

The following equation defines the output of the above neuron:

$$T_1 = g(w_1 x_1 + b) \tag{4.22}$$

Where:

- T_1 is the output of the neuron.
- w_1 is the weighted input.
- b is the bias.
- g is the activation function.
- x_1 is the input neuron.

This activation function of the node is usually a step function, linear function, or sigmoid, and there are many others used. Furthermore, by altering the values of the input weights and bias, this allows for the manipulation of the net input n before it is applied to the transfer function. This process of manipulating the weights and biases of the network resulted in the output being directed to the desired output of T [118].

When the number of inputs into a node is increased, this yields more complexity to the problem to be solved. The following figure shows a node with a number of inputs.

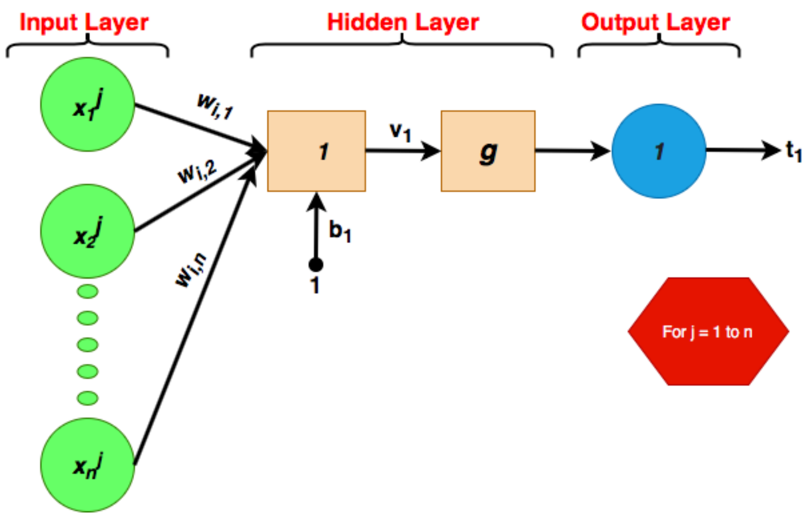


Figure 4.29: Structure of perceptron with multi-inputs

The number of inputs can be considered one dimensional vectors, with multiple elements representing the values of the inputs $[x_1, x_2, x_3, x_4, x_5......x_n]$. The weights can be similarly represented $[w_{1,1}, w_{1,2}, w_{1,3}, w_{1,4}, w_{1,5}, ...w_{1,n}]$. As a result, these input weight products are combined with the bias, as represented by the following equation:

$$v_1 = w_{1,1}x_1 + w_{1,2}x_2 + ... + w_{1,n}x_n + b \tag{4.23}$$

This node does not have sufficient processing power to solve non-linear problems; therefore, there is a need for multiple interconnected nodes to form a network with high processing power, capable of solving complex problems. The next figure illustrates a perceptron consisting of three inputs and three outputs, where each of the inputs is connected in parallel to the output through a weight factor.

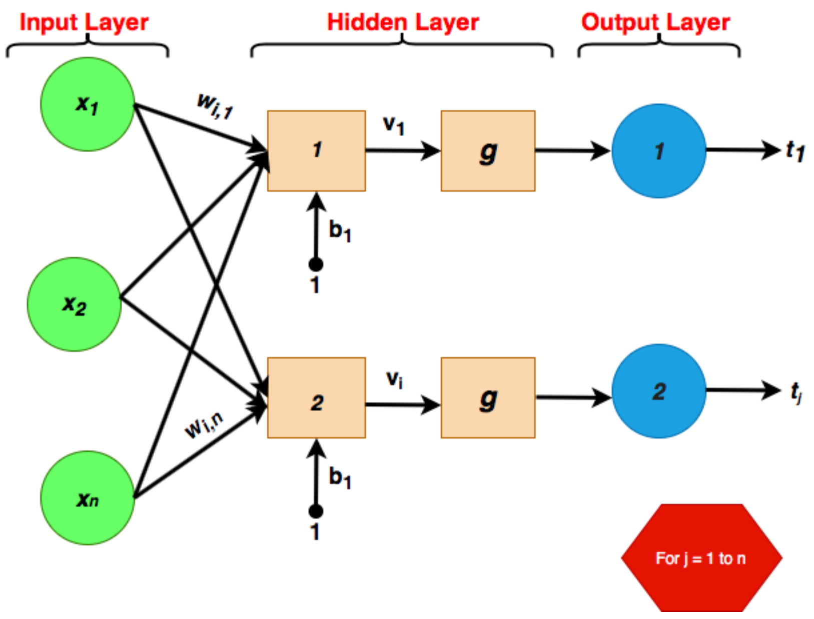


Figure 4.30: Connection of the weights between an input and an output

The transfer function of the network illustrated above can be expressed through the following equation:

$$t_i = g(\sum_n W_{in} x_n - \theta_i) \tag{4.24}$$

Where:

- t_i is the output of node i.
- w_{in} is the weight for input n to node i.
- x_n is the input value of input n.
- θ is the threshold, which is given by the transfer function of the network.

Even though perceptron networks are more superior in terms of processing power to nodes, they are still only capable of solving linear problems only because perceptron networks are not able to converge at the stage of training. On the other hand, Non-linear problems are solved through networks with three layers or more, with each layer a set of nodes. Multilayer networks consist of the typical components of a perceptron but with an additional hidden layer of nodes M. The latter also has weights and biases associated with their inputs and outputs. Therefore, there is a distinctive difference between the input weights IW and the output weights LW.

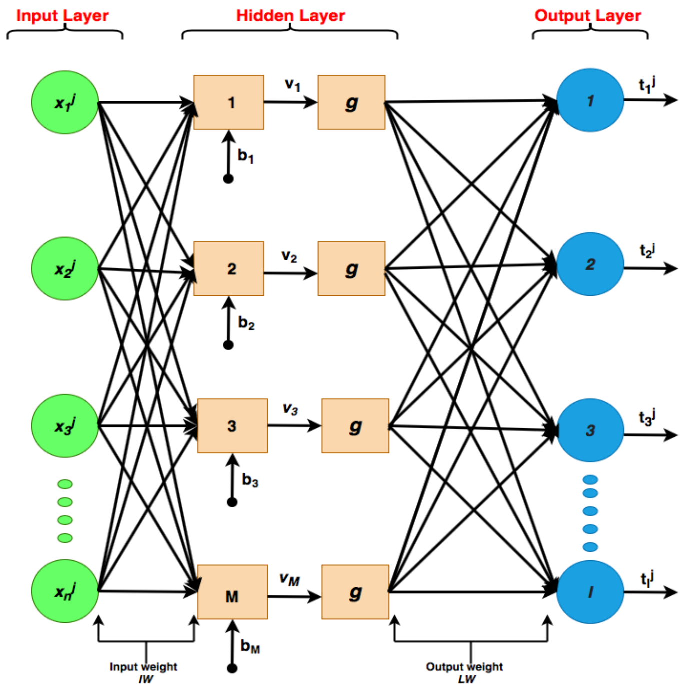


Figure 4.31: Single hidden layer feed-forward neural network (FNN)

The number of nodes in each layer need not match, and the structure of multilayer networks varies depending on the problem to be solved. In addition, the input does not necessarily need to pass through the layers of the structure and may only affect the resultant outcome through the weight factor. On the other hand, for all the networks, the direction of the prorogation of the data will always be the same: going from the input to the output.

4.3.2 Observation of deep learning neural networks

According to figure 4.31, the illustrated neural network is a single hidden layer feed-forward neural network. However, deep learning neural networks consist of many hidden layers [119], as figure 4.32 represents:

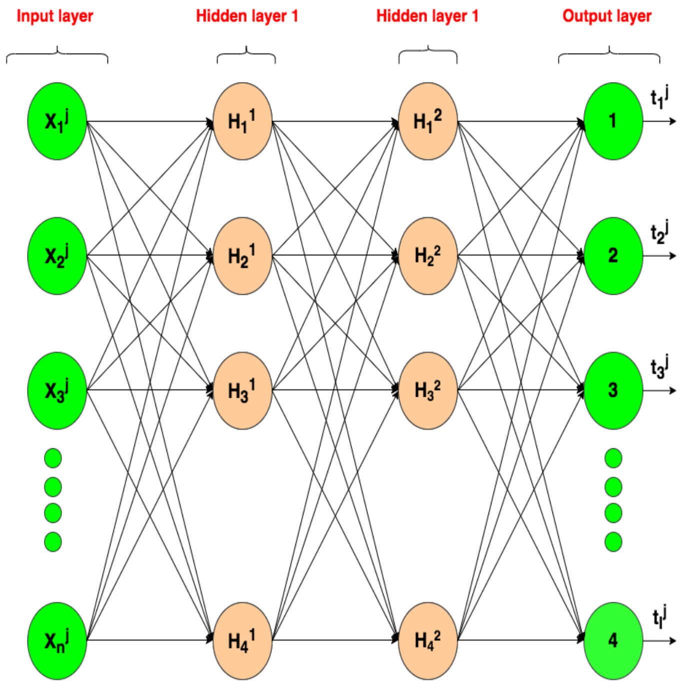


Figure 4.32: 2 hidden layer feed forward deep neural network

In feed forward deep neural networks the weights from each of the neurons in the first hidden layer to the neurons in next hidden layer follow the same rules as in the the single hidden layer feed forward neural networks. Moreover, the direction of the propagation in deep neural networks follows the same pattern as in the single hidden layer neural networks. Similarly, as shown in the following figure, the neuron in the hidden layers, for instance H_1^1, which means that this neuron is the first neuron located in the first hidden layer. This neuron as well as all the other neurons in the hidden layer consist of combinations of biases and transfer function [120].

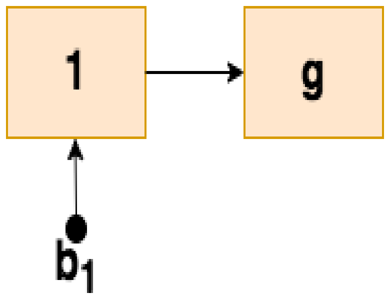


Figure 4.33: Representation of neuron H_1^1 from figure 4.32

Furthermore, another form of deep neural networks is recurrent neural networks (RNNs), which consist of multi-hidden layers, as in the feed-forward deep neural networks [121]. However, RNNs attain the capability of storing information in context nodes. Therefore the nodes can learn though feeding them with data sequences and predict a number or variety of other sequences. In other words, it is a form of deep neural networks with connections between neurons in the form of loops, and RNNs are mostly used for processing sequences of inputs [121], as figure 4.34 illustrates.

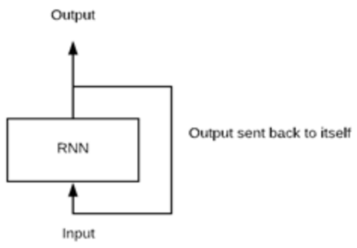


Figure 4.34: Representation of a node in the recurrent neural network [116]

For instance, if the challenge was to predict the next word in the sentence, "would you fancy a. . . . . ?"

- The RNN neurons will receive a signal that direct it to the start of the sentence.
- The network receives the word ”would” as an input and computes a vector of the number. This vector is re-input to the neuron in order to establish a memory of the network. This is done to get the network to remember it received ”would” in the first position.
- The network will similarly proceed to the next words. It takes the word ”you” and ”fancy”, then the neurons update as each word is received.
- Finally, when the word ”a” is received, the neural network computes options as to what English words can be used to finish the sentence. A properly trained RNN probably assigns a high probability to ”sandwich”, ”apple” ,”trip” etc.

Furthermore, another form of deep learning neural networks is convolutional neural networks (CNNs), which is also a multi-layered neural network with an architecture designed that is not like any other deep neural network [121]. This is because CNNs extract increasingly complex features of the data at each layer to determine the output. CNN’s are well suited for perceptual tasks [121].

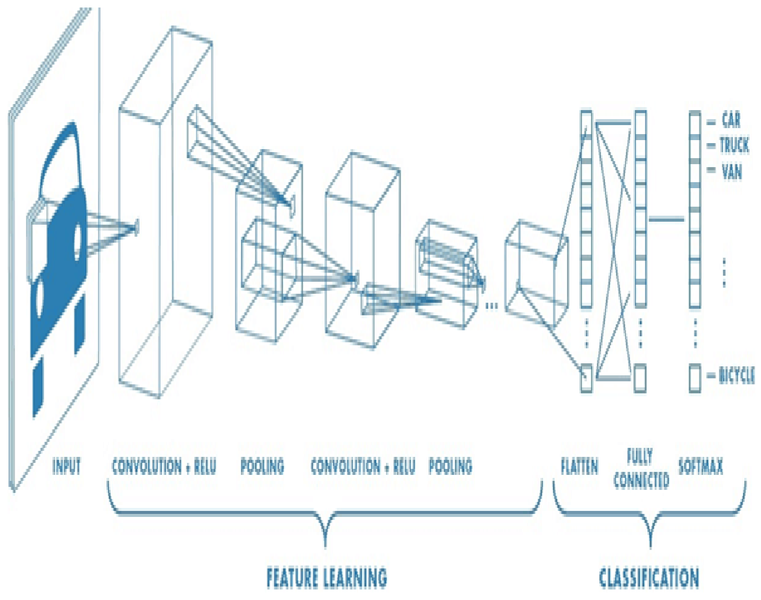


Figure 4.35: Convolutional neural network [116]

CNN are typically implemented when there is an unstructured data set that includes images, which the practitioners need to extract information from. For example, if the task is to compute an image caption:

- The CNN receives an image of, for example, a cat. This image consists of several pixels. Generally, one layer for the greyscale picture and three layers for a color picture.
- As the network learns features, it will identify special features, such as the tail of the cat.
- After the network finishes how to identify a picture, it can supply a probability for every image it recognises. This probability is in the form of winner takes all, so that the label with the highest probability wins.

Furthermore, since deep neural networks consist of more hidden layers than single hidden layer neural networks, this results in a considerably higher number of neurons. Hence more computational power is needed. The classification results obtained using training algorithms implemented on the single hidden layer neural networks explained in detail in sub-chapter 4.5.1 resulted in satisfactory results. Therefore, although using deep neural network structures can achieve higher results than a single neural network structure, the time scope of this project has not allowed further exploration of this structure. There are many training methods used for deep training methods used methods.

4.4 Training single hidden layer feed-forward artificial neural networks (ANN)

A network is only ready to be trained when it is completely structured. In order to start training neural networks, the initial weights are chosen randomly. Then, the training, or learning, begins. ANNs are based on perceptron networks and used to achieve a specific aim. In this project, the aim is to classify pollinators from features extracted from the audio signals of the species, using statistical time domain signal coding. However, for the network to work efficiently, it must be trained. Training neural networks can be categorised into two categories, which are supervised and unsupervised. Supervised training includes providing the networks with labelled data as a desired output through grading, which is the manual approach or through providing the desired outputs with an input. Example of the supervised learning algorithms are these used in this project in section (4.6). Whereas, in unsupervised learning, the network is given inputs and the desired outputs, which are the labels of the data [122]. Since the data set used in this project is all labelled then no unsupervised learning methods are used to train the neural network used in this chapter.

4.4.1 Supervised learning

In supervised learning, the technique is to use data templates as examples, with each example showing the desired output required at the output node t_l, of the network when an input pattern is given,X_n as shown in figure 4.31. This method compares the actual and desired outputs, then adjusts the network weights based on the result of the comparison. The training data are then reapplied to the input, and the process repeats until the actual output pattern is the same as the desired output pattern. This is accomplished through the application of a training rule that sets how the weights are altered. The method applied to train the feed-forward MLP network, as shown in figure 4.31, is typically the back-propagation algorithm. This is designed through the generalisation of the Widrow-Hoff learning rule to multiple-layer networks and non-linear differentiable transfer functions [93]. In terms of insect species classification, back-propagation is proven to be a valuable training method when applied to insect acoustic signals [122].

4.4.2 Unsupervised learning

In unsupervised learning, the output of the MLP is not given any predefined patterns but rather left unsupervised to obtain patterns from within the dataset. The network categorises the data into different groups based on similarities in their features. In other words, the data with common features are clustered together in networks, in a process known as 'self-organising'. For network training using an unsupervised method, datasets are presented at the input without any knowledge of the classes that these datasets are expected to fall into. As a result, the nodes in the network 'compete' in the form of 'winner takes all'. The weights of the winning node and its neighbouring nodes, as defined by some function, are then adjusted such that they begin to cluster groups of data with similar features. An example of the self-organising neural network is Kohonen's self-organising map [123].

4.5 Extreme learning machines (ELM)

The extreme learning machine (ELM) is a relatively modern training algorithm for single-layer feed-forward neural networks (SLFNs). In ELM, the nodes in the hidden layer are randomly initialised and remain constant exclusive of the need for iterative tuning, while the only parameters that need to be learnt are the weights connecting the hidden layer to the output layer. Furthermore, compared to traditional FNN learning techniques, such as back-propagation, which is discussed in the next chapter, ELM is markedly efficient. Additionally, studies have shown that although the hidden nodes are randomly generated, ELM maintains the universal approximation capabilities of SLFNs [124–126]. With the typically used activation functions, such as the sigmoid, ELM can almost reach the optimal generalisation bound of common FNN, where all of the parameters are learnt [127]. Moreover, the advantages of ELM over the traditional training algorithms of SLFNs are shown in a variety of problems from a range of fields [128, 129]. ELM is generally more efficient than SVM [130]. In addition, ELM has a generalisation ability comparable to (or better than) SVM in empirical studies [128, 129, 131, 132]. In depth comparisons of ELM and SVM can be found in [128, 133]. ELM has been used for system modelling and prediction systems, with an ELM-based predictor for real-time frequency stability assessments (FSA) of power systems [134]. The input of the predictor power system operational parameters, whilst the output is the frequency stability margin, which computes the stability degree of the power system subject to contingency. Through a frequency stability database, offline training is performed and the predictor can be applied online for real-time FSA. This predictor was implemented on New England, with a 10-generator 39-bus test system, and the simulation results show that it can accurately predict frequency stability. Furthermore, ELM has been used for other prediction systems, such as electricity price forecasting [135], temperature prediction of molten steel [136], and sales forecasting [137, 138].

4.5.1 Implementing the extreme learning machines (ELM) algorithm

As mentioned earlier, the structure of the network must be of a single hidden layer for the ELM algorithm to be performed.

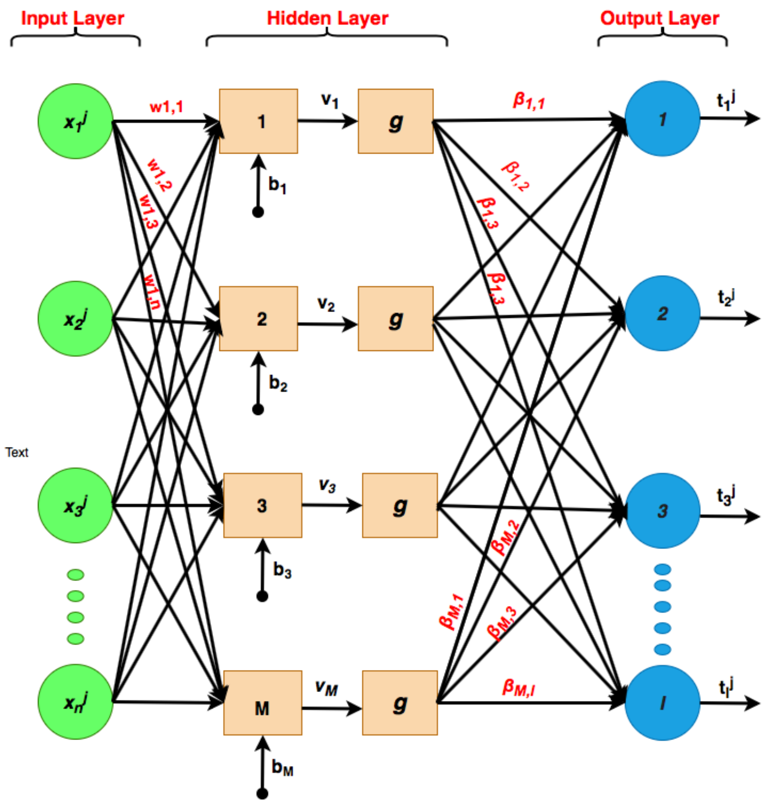


Figure 4.36: Single hidden layer neural network for extreme learning machine (ELM)

Figure(4.15) shows the input weight vector of the i_{th} hidden neuron $(1 \times n)$); for instance, W_i is the input weight vector of the first hidden neuron $W_1 = [w_{1,1}, w_{1,2}, w_{1,3}, w_{1,4}, w_{1,5}, ...w_{1,n}]$. On the other hand, the bias of the i_{th} hidden neuron is in the shape of (1×1); for example, b_1 is the bias of the hidden neuron.

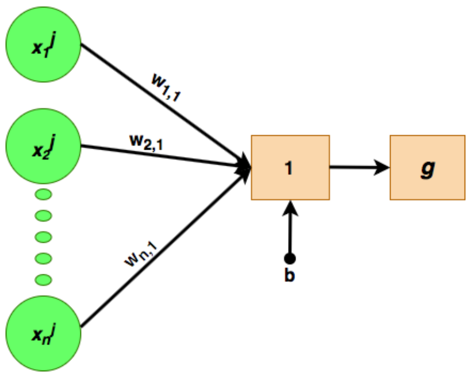


Figure 4.37: Connection of the weights to the function with the bias.

The output of the hidden neuron i for the input sample j is given by the following:

$$t_i = g(W_i.X_j + b_i), For j = 1, 2.......N \tag{4.25}$$

Where:

- t_i is the output of the neuron i (1×1).
- W_i is the input of the weight row vector of the neuron $I(1 \times n)$.
- b_i is the bias of the neuron i (1×1).
- X_j is the j input sample. $n \times 1$.
- $g(.)$ is the activation function of the hidden neuron.

For instance, the output of the neuron 1 for the input sample 1 is given by the following:

$$t_1 = g(W_1.X_1 + b_1) = g(v_1) \tag{4.26}$$

$$v_1 = [w_{11}w_{12}w_{13}....w_{1n}].\begin{bmatrix} x_1^1 \\ x_2^1 \\ x_3^1 \\ . \\ . \\ . \\ . \\ . \\ x_n^1 \end{bmatrix} + b \tag{4.27}$$

$$t_1 = g(v_1) \tag{4.28}$$

Also, the output weight vector of the i_th hidden neuron$(l \times 1)$, $\beta_i =$
$\begin{bmatrix} \beta_{i1} \\ \beta_{i2} \\ \beta_{i3} \\ . \\ . \\ . \\ . \\ \beta_{il} \end{bmatrix}$

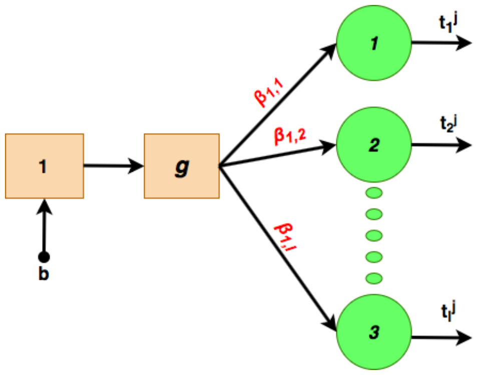


Figure 4.38: Connections between the function and the output i.e targets

Therefore, and as already mentioned, the output of the neural network structure is simply given by the following:

$$\sum_{i=1}^{M} \beta_i g(W_i.X_j + b_i) = O_j \tag{4.29}$$

$For j = 1, 2, ...N$
where:

- β_i is of length $l \times 1$
- $g(W_i.X_j + b_i)$ is of length 1×1.
- O_j is of length $l \times 1$

The input weight matrix W_in for n input neurons and M hidden neurons, and the matrix W_in, is created with the size of $M \times n$.

$$W_in = \begin{bmatrix} W_1 \\ W_2 \\ . \\ . \\ . \\ . \\ W_M \end{bmatrix} = \begin{bmatrix} W_{11}.......W_{1n} \\ .. \\ .. \\ .. \\ W_{M1}........W_{Mn} \end{bmatrix} \tag{4.30}$$

The input matrix of the network (X) for n input neurons and N stochastic sample, and the matrix X, is created with the size of $n \times N$.

$$X = \begin{bmatrix} X_1.....X_N \end{bmatrix} = \begin{bmatrix} X_1^1........X_1^N \\ .. \\ .. \\ .. \\ X_n^1........X_n^N \end{bmatrix} \tag{4.31}$$

The bias matrix of the hidden layer (b) for N stochastic sample and M hidden neurons, and the matrix b, is created, with the size of $M \times N$.

$$b = \begin{bmatrix} b_1.......b_1 \\ .. \\ .. \\ .. \\ b_M........b_M \end{bmatrix} \tag{4.32}$$

The reason for the repetition of the bias column is to calculate the H matrix, which is the hidden layer output matrix with the size of $(M \times N)$.

$$H = g(W_{in}.X + b) = g(\begin{bmatrix} W_{11}.......W_{1n} \\ .. \\ .. \\ .. \\ W_{M1}........W_{Mn} \end{bmatrix} . \begin{bmatrix} X_1^1.......X_1^N \\ .. \\ .. \\ .. \\ X_n^1........X_n^N \end{bmatrix} + \begin{bmatrix} b_1.......b_1 \\ .. \\ .. \\ .. \\ b_M........b_M \end{bmatrix}) \tag{4.33}$$

Where $g(.)$ is the activation function for each element in H, this can be written in the following form:

$$H = \begin{bmatrix} g(W_1.X_1 + b_1).......g(W_1.X_N + b_1) \\ .. \\ .. \\ .. \\ g(W_M.X_1 + b_M)........g(W_M.X_N + b_1) \end{bmatrix} \tag{4.34}$$

The output matrix b for M hidden neurons and l output neurons and b has the size of $(l\ timesM)$.

$$\beta = \begin{bmatrix} \beta_{11}.......\beta_{M1} \\ .. \\ .. \\ .. \\ \beta_{1l}........\beta_{1l} \end{bmatrix} \tag{4.35}$$

Therefore, the output matrix of the network for l output neurons and N stochastic sample, and the matrix T, with the size of $(l \times N)$, is calculated as follows:

$$T = \beta H = \begin{bmatrix} \beta_{11}.......\beta_{M1} \\ .. \\ .. \\ .. \\ \beta_{1l}........\beta_{1l} \end{bmatrix} \begin{bmatrix} g(W_1.X_1 + b_1).......g(W_1.X_N + b_1) \\ .. \\ .. \\ .. \\ g(W_M.X_1 + b_M)........g(W_M.X_N + b_1) \end{bmatrix} \tag{4.36}$$

$$T = \begin{bmatrix} t_1^1 ....t_1^N \\ .. \\ .. \\ .. \\ t_l^1 ....t_l^N \end{bmatrix} \tag{4.37}$$

The ELM technique is performed in the following three steps:

- Defining the hidden nodes, activation function, and the number of hidden neurons.
- Randomly assigning the input weights W_{in} $(M \times n)$ and bias $b(M \times 1)$.
- Calculating the output weights $\beta(l \times M)$ using the pseudo inverse as follows:

$$T = \beta H \text{ or } H\beta = T$$
$$\beta H H^T = T H^T$$
$$\beta = T H^T (H H^T)^- 1$$

4.5.2 Experimenting with ELM

The following table presents the results of the four species classifications by SVM and RF. In order to apply ELM to classify the four species as in SVM and RF, other parameters must be adjusted and the proportion of data used for training and testing. In ELM, there is also the number of epochs, which is the number of times the training vectors are used to update the weights, as well as the number of hidden neurons. Firstly, the number of hidden neurons ranges from 50-1500, while keeping the epochs fixed at 100 and using 50% of the data as training files. Furthermore, based on the SVM and RF experiments, S-TDSC is proven to provide greater accuracy than the frequency domain method; therefore, the features used for the ELM classifier experiments are extracted using

S-TDSC. In this example, the training files are shuffled, while the testing files remain unshuffled.

Performing the four species classification, with 50% of the data used for training and epochs=100. (unshuffled training and testing data).

Number of hidden neurons	**Training accuracy**	**Testing accuracy**
50	73%	65.42%
60	75.42%	66.57%
70	78.57%	71.71%
80	81%	72.71%
90	84.28%	73.16%
100	88.42%	73.85%
110	89.42%	72.28%
120	89%	74.57%
130	90%	72.71%
140	92.28%	72.71%
150	94.28%	76.85%
160	95.57%	75.14%
170	95.42%	75.85%
180	96%	76.14%
190	95.14%	75.14%
200	95.57%	76.57%

Table 4.8: Training and testing accuracies, when 50% of the data are used for training and the number of epochs is 100

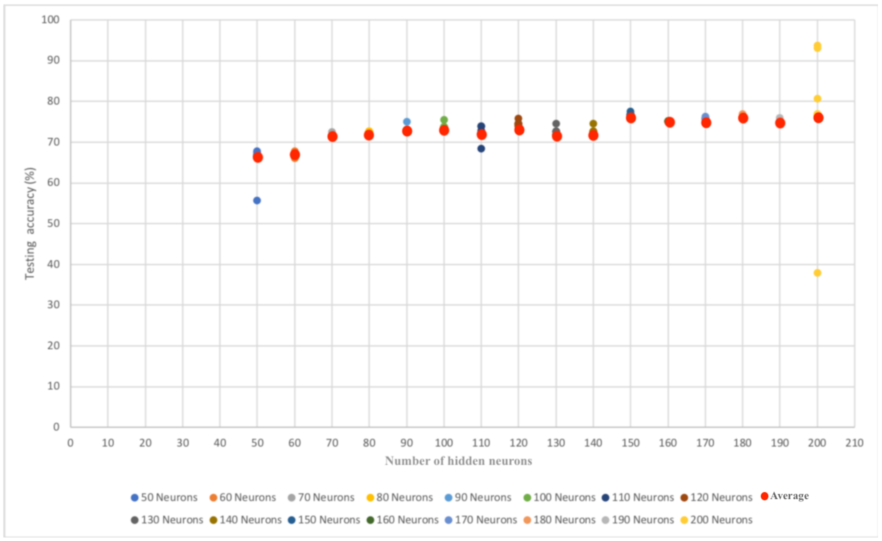


Figure 4.39: Training and testing accuracies, when 50% of the data are used for training and the number of epochs is 100

The following are confusion matrixes of the run that gives the maximum accuracy, of the seven runs averaged from a number of hidden neurons of 50 and 200. The confusion matrices corresponding to the number of hidden neurons between 60 to 190 in steps of ten are shown in appendix C.

Number of hidden neurons = 50:

- Training accuracy 68.23%
- Testing accuracy 67.85%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	26	4	1	0
Solitary bees	6	29	1	0
Bumblebees	0	1	2	0
Hover flies	7	6	1	0

Table 4.9: Confusion matrix when the number of hidden neurons is 50 and the number of epochs is 100

Number of hidden neurons = 200:

- Training accuracy 75.81%
- Testing accuracy 93.75%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	15	0	0	0
Solitary bees	0	0	0	0
Bumblebees	1	0	0	0
Hover flies	0	0	0	0

Table 4.10: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

It is noticed from table 4.8 that as the number of hidden neurons increases, the training and testing accuracies increase, with the testing accuracy ranging from 65.42% at 50 neurons to 76.75% at 200 neurons. Additionally, the reason why the number of neurons stopped at 200 is that more neurons would not be computationally treatable in the target system, which is mentioned in chapter 5. Also figure 4.39 shows that the average testing accuracy is slightly increasing from 50 to 200 neurons. On the other hand, it is noticed that when 200 neurons were used, few of the testing accuracies that make up the average were significantly above and below the average with 93.75% and below 40% respectively. However, in most of the other cases, the testing accuracies are always either equal to the average or above by a small margin.

Performing the four species classification, with 50% of the data used for training and epochs=100. (shuffled training and testing data)

when both training and testing data are shuffled at each run of the code, training accuracy increases – reaching 100% at 350 neurons, while the testing accuracy remains almost constant between 45% and 50%, regardless of the increase in hidden neurons. The next table illustrates the training and testing accuracies for 50 to 1500 hidden neurons.

Number of hidden neurons	**Training accuracy**	**Testing accuracy**
50	68.7%	47%
100	81.5%	47.7%
150	93.5%	46.5%
200	95.2%	45.2%
250	98.1%	47.1%
300	97.4%	47.7%
350	100%	48.2%
400	100%	48.6%
450	100%	49.6%
500	100%	50%
550	100%	48.2%
600	100%	50%

650	100%	48.6%
700	100%	50%
750	100%	50%
800	100%	48.6%
850	100%	51.8%
900	100%	50.1%
950	100%	51.8%
1000	100%	47.7%
1050	100%	49.3%
1100	100%	49.8%
1150	100%	50.17%
1200	100%	50.36%
1250	100%	51%
1300	100%	49.6%
1350	100%	49.6%
1400	100%	49.5%
1450	100%	50%
1500	100%	49.8%

Table 4.11: Training and testing accuracies, when 50% of the data are used for training and the number of epochs is 100

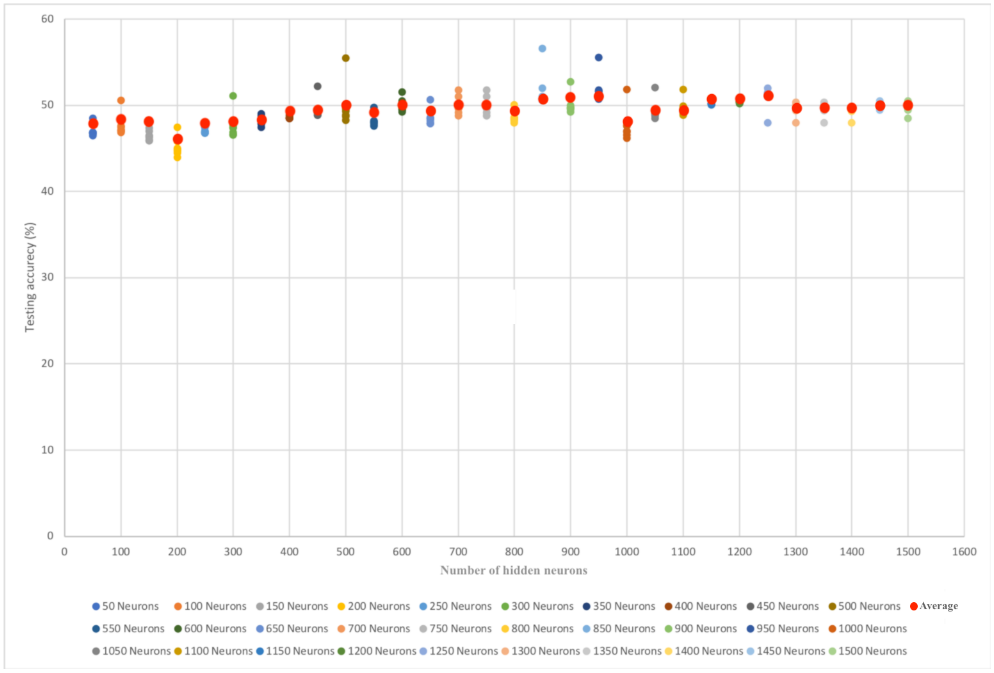


Figure 4.40: Training and testing accuracies, when 50% of the data are used for training and the number of epochs is 100

It is noticed from table 4.11 that as the number of hidden neurons increases, the training accuracy increases. The training accuracy starts with 68.7% at 50 neurons then increases to 81.5% when 100 neurons. After that, the accuracy goes up to range from 93.5% to 97.4% from 150 neurons to 300 neurons. Moreover, the training accuracy reaches 100% and remains fixed from 350 neurons to 1500 neurons. On the other hand, the testing accuracy ranges from 47% at 50 neurons to 49.8% when 1500 neurons were used. Additionally, figure 4.40 illustrates that all the average testing accuracy, as well as the trials that make it up, remain almost fixed as they range from 45.2% to 55.8%. Therefore, when shuffling both the training and testing data, the training accuracy significantly increases as the hidden neurons increases. In contrast, the testing accuracy remains fixed.

Performing the four species classification, with 70% of the data used for training, keeping the number of epochs fixed at 100. (shuffling the training data only)

Number of hidden neurons	Training accuracy	Testing accuracy
50	63.28%	73.25%
60	68.85%	79.14%
70	72.28%	74.28%
80	71%	78.57%
90	79.85%	82.28%
100	84.71%	82.57%
110	81.28%	86.28%
120	84.42%	84%
130	84.42%	86.28%
140	86.71%	86.57%
150	90.28%	85.71%
160	91.57%	89.71%
170	93.14%	89.42%
180	95%	88%
190	94.57%	90.57%
200	94.57%	90.57%

Table 4.12: Training and testing accuracies when 70% of the data are used for training and the number of epochs is 100, when only the training data are shuffled

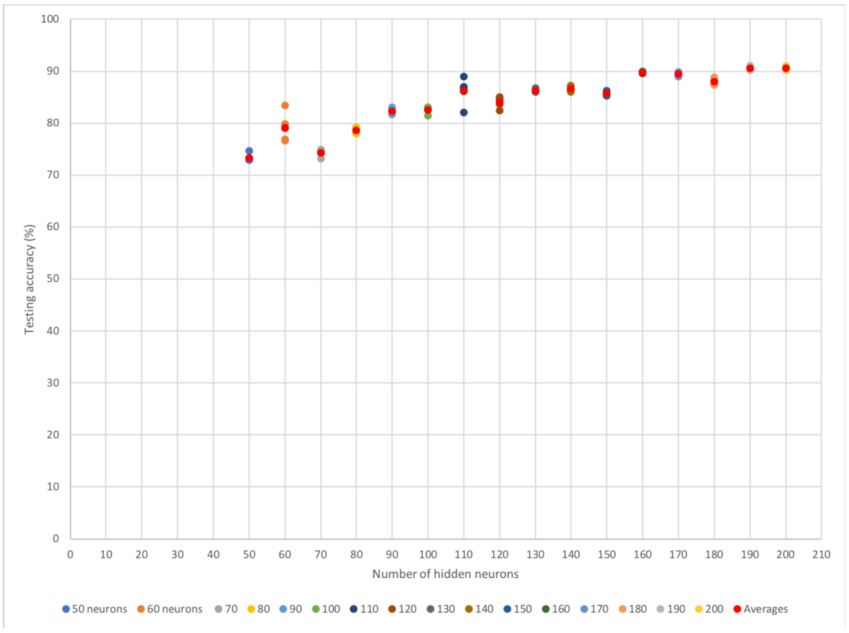


Figure 4.41: Training and testing accuracies when 70% of the data are used for training and the number of epochs is 100, when only the training data are shuffled

It can be seen from table 4.12 that when only the training data is being shuffled, the accuracy increases from 73.25% at 50 neurons to 90.57% at 200 neurons. It is also noticed from graph 4.41 that the trails that make up the averages are not significantly higher or below it. Moreover, from the resulting confusion matrices, it is noticed that this high accuracy is because the decision of the classifier is biased towards commons wasps and solitary bees as they contribute to the majority of the recording in the database. The issue of the biasing algorithm can be tackled by more data available to train the classification model. The following are the confusion matrices of the run that gives the maximum accuracy, of the seven runs averaged from the number of hidden neurons of 50 and 200. Additionally, confusion matrices developed as a result of using the numbers of hidden neurons from 60 to 190 are shown in Appendix D.

Number of hidden neurons = 50:

- Training accuracy 70.58%
- Testing accuracy 84.0

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	32	0	0	0
Solitary bees	0	10	0	0
Bumblebees	1	0	0	0
Hover flies	6	1	0	0

Table 4.13: Confusion matrix when the number of hidden neurons is 50 and the number of epochs is 100

Number of hidden neurons = 200

- Training accuracy 96.63%
- Testing accuracy 94.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	37	0	0	0
Solitary bees	0	10	0	0
Bumblebees	1	0	0	0
Hover flies	2	1	0	0

Table 4.14: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

Performing the four species classification, with 70% of the data used for training and epoch=100. (shuffled training and testing data

Number of hidden neurons	**Training accuracy**	**Testing accuracy**
50	63%	56%
100	78%	56%
150	86%	55.7%
200	94.5%	52.8%
250	96.78%	54%
300	96.9%	55.14%
350	98.4%	52.8%
400	97.1%	54%
450	100%	55.7%
550	100%	54.5%
600	100%	55.4%
650	100%	56%
700	100%	58.5%
750	100%	58.2%
800	100%	57.7%
850	100%	58%
900	10%	56.5%
950	100%	54.5%
1000	100%	56.8%
1050	100%	52.8%
1100	100%	55.7%
1150	100%	55.7%
1200	100%	57.7%
1250	100%	57.7%
1300	100%	58.2%
1350	100%	56.5%

1400	100%	55.4%
1450	100%	57.1%
1500	100%	54.2%

Table 4.15: Training and testing accuracies when 70% of the data are used for training and the number of epochs is 100, while the training and testing data are shuffled

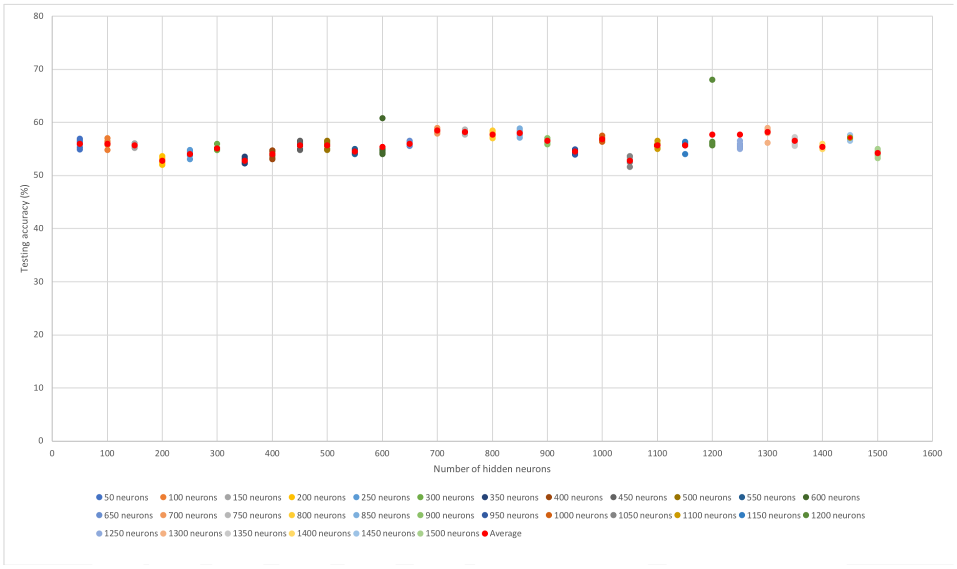


Figure 4.42: Training and testing accuracies when 70% of the data are used for training and the number of epochs is 100, when only the training data are shuffled

The testing accuracy of this network structure does not increase further, regardless of the increase in hidden neurons and epochs, as accuracy is capped between 54% and 58%.However, the training accuracy significantly goes up from 63% when 50 neurons were used to 100% when 400 neurons were used and remained fixed there until the end. On the other hand, it is noticed the accuracies of all the trials that make up the average either increase or decrease as the average rises or decline. The only factor associated with increased accuracy is the use of more data for training and the next experiment proves this.

Performing a binary classification (bumblebee classifier), with 50% of the data used for training and the number of epochs, is 100. (shuffled training and testing data)

To prove that the main factor in increasing the accuracy further is for more recordings to be available for training, this experiment was conducted. This experiment is a bumblebee classifier, with the recordings of bumblebees grouped in class1 and the audio recordings from the other three species grouped in class 2. Table 4.16 shows that the testing accuracy ranges from 67% to 69%, with the training data and testing data shuffled at each run.

Number of hidden neurons	**Training accuracy**	**Testing accuracy**
50	73%	70%
100	88%	67%
150	93%	66%
200	97%	67%
250	98%	68%
300	98%	68%
350	99%	66%
400	98%	67%
450	99%	66%
500	99%	68%
550	99%	67%
600	100%	69%
650	100%	69%

700	99%	69%
800	100%	66%
850	100%	69%
900	100%	68%
950	100%	67%
1000	100%	66%
1050	100%	68%
1100	100%	70%
1150	100%	69%
1200	100%	69%
1250	100%	69%
1300	100%	68%
1350	100%	67%
1400	100%	68%
1450	100%	66%
1500	100%	69%

Table 4.16: Training and testing accuracies when 50% of the data are used for training and the number of epochs is 100, whilst the training and testing data are shuffled

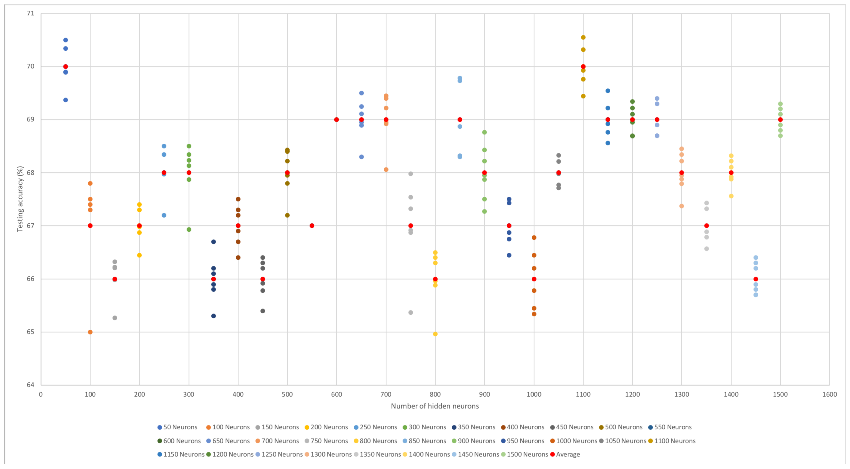


Figure 4.43: Training and testing accuracies when 70% of the data are used for training and the number of epochs is 100, when the training and data are shuffled

Figure 4.43 illustrates that trials that make up the averages are within +1 or -1 range from the average and only when 100 neurons were used the minimum accuracy falls to beyond -1 with 65% when the actual average was 67%. The following are confusion matrices of the run that gives the maximum accuracy, of the seven runs averaged from a number of hidden neurons of 50 and 1500. In addition confusion matrices developed in the same manner for a number of hidden neurons ranging from 100 to 100 to 1450 in steps of 50s are shown in appendix E.

Number of hidden neurons = 50:

- Training accuracy 67.22%
- Testing accuracy 74.0%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	25	7
Another species	6	12

Table 4.17: Confusion matrix when the number of hidden neurons is 50 and the number of epochs is 100

Number of hidden neurons = 1500:

- Training accuracy 100.0%
- Testing accuracy 73.80%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	45	16
Another species	6	17

Table 4.18: Confusion matrix when the number of hidden neurons is 50 and the number of epochs is 100

Although ELM has shown a noticeable increase in terms of the testing accuracy compared to the testing accuracy resulting from SVM and random forests, it is still essential to use the hand-written digits dataset to verify the performance of ELM. Therefore, when 70% of the hand-written digits dataset was used for training, the testing result after an average of 7 runs is 98% and the confusion matrix that resulted in the maximum accuracy out of the 7 runs is as follows:

	Target									
Predicted	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	58	0	0	0	0	0	0	0	0	0
1	0	65	1	2	0	0	0	0	1	0
2	1	0	50	1	0	0	0	0	1	0
3	0	0	0	59	0	0	0	0	0	1
4	0	0	0	0	43	0	0	0	0	0
5	0	0	0	0	0	48	0	0	0	0
6	0	0	0	1	0	0	48	0	0	0
7	0	0	0	1	0	0	0	57	0	0
8	0	1	0	0	0	0	0	0	44	1
9	0	0	0	0	0	1	0	0	0	59

Table 4.19: Confusion matrix of the hand-written digits dataset resulting from ELM algorithm

4.6 Back-propagation

This algorithm was developed in the 1970s, but its importance was not fully understood until 1986 when it was proven to function faster than all other approaches when applied to several neural networks [139]. Thus, allowing neural networks to solve problems that had previously been insoluble. Furthermore, the back-propagation algorithm is one of the most investigated and frequently used techniques for neural networks training [140].

The following is a simple neural network with a single hidden layer.

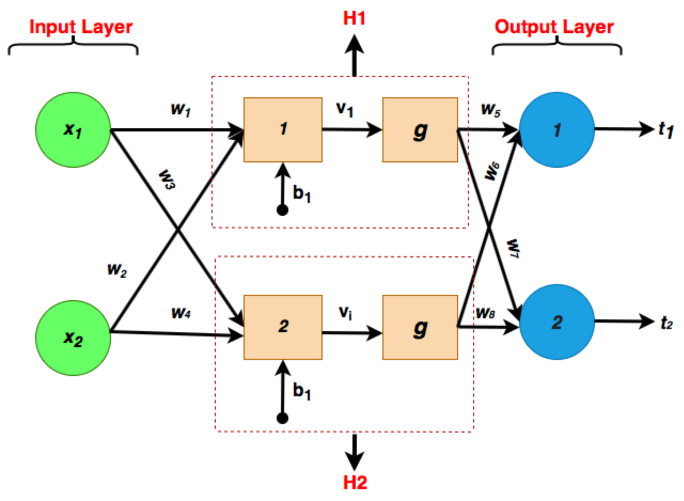


Figure 4.44: Structure of neural network trained using backpropagation

Where:

$$H_1 = (X_1 \times \omega_1 + X_2 \times \omega_2 + b_1) \tag{4.38}$$

$$H_1 = (X_1 \times \omega_4 + X_2 \times \omega_4 + b_1) \tag{4.39}$$

A non-linear activation function is applied to the output of every hidden layer because non-linear functions have more than one degree and a curved shape when plotted. This allows the network to be more powerful and better able to perform complex learning from data, representing non-linear complex arbitrary functional mapping from inputs to outputs. Additionally, one of the most popular activation functions is the sigmoid function [99]. Therefore, when the latter is applied to H1 and H2, the output of the hidden layer is represented as follows:

$$H_1 = \frac{1}{1 + \exp^{-H_1}} \tag{4.40}$$

$$H_2 = \frac{1}{1 + \exp^{-H_2}} \tag{4.41}$$

To investigate the backpropagation algorithm, we assume that the input feature vectors X1 and X2 are 0.05 and 0.10, respectively, and the target values for T1 and T2 are 0.01 and 0.99, respectively. Firstly, in backpropagation, all the weights and biases are given random values, as represented in figure 4.40.

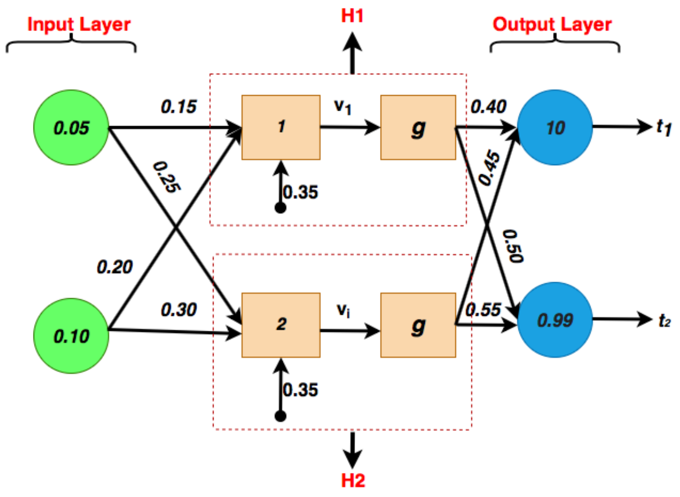


Figure 4.45: Neural network trained with backpropagation, with initial random values for weights

The next step is to perform what is commonly known as the ‘forward press’, which is done by first calculating H1 and H2, then obtaining the outputs t1 and t2.

$$H_1 = (X_1 \times \omega_1 + X_2 \times \omega_2 + b_1) \tag{4.42}$$

$$H_1 = (0.05 \times 0.15 + 0.10 \times 0.20 + 0.35) \tag{4.43}$$

$$H_1 = 0.3775 \tag{4.44}$$

Therefore, the output at H1 is as follows:

$$H_1 = \frac{1}{1 + \exp^{-H_1}} \tag{4.45}$$

$$H_1 = \frac{1}{1 + \exp^{-0.3775}} = 0.593269992 \tag{4.46}$$

Similarly, $H_2 = 0.596884378$. The next step is to calculate y1 and y2:

$$t_1 = ((\frac{1}{1 + \exp^{-H_1}}) \times \omega_5 + ((\frac{1}{1 + \exp^{-H_2}}) \times \omega_6 + b_2 \tag{4.47}$$

$$t_1 = (0.4 \times 0.593269992) + (0.596884378 \times 0.45 + 0.6) \tag{4.48}$$

$$t_1 = 1.105905967 \tag{4.49}$$

The output of t_1 is calculated as follows:

$$Outt_1 = \frac{1}{1 + \exp^{-t_1}} \tag{4.50}$$

$$Outt_1 = \frac{1}{1 + \exp^{-1.105905967}} \tag{4.51}$$

$$Outt1 = 0.75136507 \tag{4.52}$$

Similarly, $Outt2 = 0.772928465$. As a result, the output values of $t1$ and $t2$ do not match the target values of $T1$ and $T2$ and an error occurs. Furthermore, the final step of the forward press is calculating the total error, using the loss function, which is defined by the following equation:

$$E_{total} = \sum \frac{1}{2}(Target - t)^2 \tag{4.53}$$

$$E_{total} = \frac{1}{2}(T_1 - Out(t_1))^2 + \frac{1}{2}(T_2 - Out(t_2))^2 \tag{4.54}$$

$$E_{total} = \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772)^2 \tag{4.55}$$

$$E_{total} = 0.274811083 + 0.023560026 \tag{4.56}$$

$$E_{total} = 0.298371109 \tag{4.57}$$

At this point, the error rate is obtained and needs to be back propagated through the backward press to update the weights and minimise the error, starting by updating the error at w_5.

$$\text{Error at } w_5 = \tfrac{\sigma E_{total}}{w_5}$$

$$\frac{\sigma E_{total}}{\sigma w_5} = \frac{\sigma E_{total}}{\sigma Out(t_1)} \times \frac{\sigma Out(t_1)}{\sigma t_1} \times \frac{\sigma t_1}{\sigma w_5} \tag{4.58}$$

$$E_{total} = \frac{1}{2}(T_1 - Out(t_1)^2) + \frac{1}{2}(T_2 - Out(t_2))^2 \tag{4.59}$$

$$\frac{\sigma E_{total}}{\sigma out(t_1)} = 2 \times \frac{1}{2} \times (T_1 - out(t_1))^{2-1} \times -1 + 0 \tag{4.60}$$

$$\frac{\sigma E_{total}}{\sigma out(t_1)} = -(T_1 - Out(t_1)) \tag{4.61}$$

$$\frac{\sigma E_{total}}{\sigma out(t_1)} = -(0.01 - 0.75136507) \tag{4.62}$$

$$\frac{\sigma E_{total}}{\sigma out(t_1)} = 0.74136507 \tag{4.63}$$

$$Out(t_1) = \frac{1}{1 + e^{-t_1}} \tag{4.64}$$

$$\frac{\sigma Out(t_1)}{\sigma t_1} = Out(t_1)(1 - out(t_1) \tag{4.65}$$

$$\frac{\sigma Out(t_1)}{\sigma t_1} = 0.75136507 \times (1 - 0.75136507) \tag{4.66}$$

$$\frac{\sigma Out(t_1)}{\sigma t_1} = 0.186815602 \tag{4.67}$$

similarly,

$$\frac{\sigma(t_1)}{\omega_5} = 1 \times out(H_1) \times \omega_5^{(1-1)} + 0 + 0 \tag{4.68}$$

$$\frac{\sigma(t_1)}{\omega_5} = out(H_1) \tag{4.69}$$

$$\frac{\sigma(t_1)}{\omega_5} = 0.593269992 \tag{4.70}$$

$$\frac{\sigma E_{total}}{\sigma w_5} = \frac{\sigma E_{total}}{\sigma Out(t_1)} \times \frac{\sigma Out(t_1)}{\sigma t_1} \times \frac{\sigma t_1}{\sigma w_5} \tag{4.71}$$

$$\frac{\sigma E_{total}}{\sigma w_5} = 0.74136507 \times 0.186815602 \times 0.593269992 \tag{4.72}$$

$$\frac{\sigma E_{total}}{\sigma w_5} = 0.082167041 \tag{4.73}$$

Where is the learning rate used to scale the magnitude of the weight update, therefore minimising the neural network's loss function, the learning rate used for this example is 0.5. Similarly, the output weightsω_6,ω_7, and ω_8 are calculated. At this point, the back-propagation algorithm has reached the hidden layer, and the next step is to update the input weightsω_1 ,ω_2,ω_3 and w_4. First, the error is calculated at ω_1.

$$\frac{\sigma E_{total}}{\sigma w_1} = \frac{\sigma E_{total}}{\sigma Out(H_1)} \times \frac{\sigma Out(H_1)}{\sigma H_1} \times \frac{\sigma H_1}{\sigma w_1} \tag{4.74}$$

The first fraction of equation 4.73 is $\frac{\sigma E_{total}}{\sigma w_1}$

$$\frac{\sigma E_{total}}{\sigma Out(H_1}) = \frac{\sigma E_1}{\sigma Out(H_1)} + \frac{\sigma E_2}{\sigma Out(H_1)} \tag{4.75}$$

To solve $\frac{\sigma E_1}{\sigma Out(H_1)}$:

$$\frac{\sigma E_1}{\sigma Out(H_1)} = \frac{\sigma E_1}{\sigma t_1} \times \frac{\sigma t_1}{\sigma Out(H_1)} \tag{4.76}$$

To solve: $\frac{\sigma E_1}{\sigma t_1}$:

$$\frac{\sigma E_1}{\sigma t_1} = 0.74136507 \times 0.186815602 \tag{4.77}$$

$$\frac{\sigma E_1}{\sigma t_1} = 0.138498562 \tag{4.78}$$

Since,

$$\frac{\sigma y_1}{\sigma Out(H_1)} = \omega_5 = 40 \tag{4.79}$$

Therefor,

$$\frac{\sigma E_1}{\sigma Out(H_1)} = 0.138498562 \times 0.40 \tag{4.80}$$

$$\frac{\sigma E_1}{\sigma Out(H_1)} = 0.055399425 \tag{4.81}$$

By repeating the same approach, $\frac{\sigma E_2}{\sigma Out H_2}$ is found to be -0.019049119. Therefore,

$$\frac{\sigma E_t otal}{\sigma Out(H_1)} = 0.055399425 + (-0.019049119) \tag{4.82}$$

To solve the second fraction of (4.73), this is (Out H1)/H1 $\frac{\sigma Out(H_1)}{\sigma(H_1)}$

$$frac\sigma E_t otal\sigma Out(H_1) = Out(H_1) \times (1 - Out(H_1) = 0.5932(1 - 0.593) \tag{4.83}$$

$$frac\sigma E_t otal\sigma Out(H_1) = 0.5932(1 - 0.593) \tag{4.84}$$

where,

$$Out(H_1) = \frac{1}{1 + e^{-H_1}} \tag{4.85}$$

and,

$$H_1 = \omega_1 \times x_1 + \omega_2 \times x_2 + b_1 \tag{4.86}$$

The final function in equation (4.73) is $\frac{\sigma H_1}{\sigma \omega_1}$

$$\frac{\sigma H_1}{\sigma \omega_1} = 0.05 \tag{4.87}$$

Now, substituting all the fractions obtained in (4.73):

$$\frac{\sigma E_{total}}{\sigma \omega_1} = 0.03635 \times 0.241300 \times 0.05 \tag{4.88}$$

$$\frac{\sigma E_{total}}{\sigma \omega_1} = 0.000438568 \tag{4.89}$$

As a result, the change in ω_1 is 0.000438568, and the next step is to update$\omega_1$based on the change value, using the following equation:

$$\omega_1 = \omega_1 - \mu \times \frac{\sigma E_{total}}{\sigma \omega_1} \tag{4.90}$$

$$\omega_1 = 0.15 \times -0.5 \times 0.000438 \tag{4.91}$$

$$\omega_1 = 0.149780716 \tag{4.92}$$

Similarly, continuing the backward press to update the rest of the input weights, the updated values are as follows:

- $w_2 = 0.19956143$
- $w_3 = 0.24975114$
- $w_4 = 0.29950229$

Finally, the forward press is reapplied to identify whether the obtained values at the output layer are closer to the target. The error of the network is calculated to be 0.291027924, compared to the original error of 0.298371109. The reduction is not significant, but when this process is repeated 10,000 times, the error is substantially reduced to 0.0000351085. As a result, when the two inputs are fed forward, the output neuron computes T1 and T2 to be 0.05 and 0.1, respectively. The number at which the forward press and the backward press are repeated is commonly known as the number of epochs.

Furthermore, up until this point, ELM achieved the best result of classification in the example of the four species classification. Therefore, in order to compare ELM and back-propagation, the parameters of the neural network trained with back-propagation are kept the same as in ELM when the latter has achieved its best classification accuracy. Furthermore, the following experimental results were achieved by setting the number of hidden layers kept to 1, the number of hidden neurons to 1500, and a learning rate varying from 0.001 to 0.1. However, the function is sigmoid instead of hardlim because while experimenting, sigmoid gave a better classification.

Learning rate	**Training accuracy**	**Testing accuracy**
0.0001	60%	45.2%
0.001	60.5%	46.1%
0.01	64%	45%
0.1	62%	42.2%

Table 4.20: Results of a binary classification (bumblebee classifier), with 50% of the data used for training and the number of epochs is 100. (shuffled training and testing data)

To conclude, it is clear that the accuracy obtained using a neural network trained with back-propagation is not as high as the neural network trained with ELM. While using 1500 hidden neurons in ELM, the testing accuracy reached 70%. However, when the same number of hidden neurons was used for back-propagation, the accuracy varied between 42.2% to 45.5% depending on the learning rate. Finally, when using the hand-written digits example to verify the code made for the back-propagation algorithm, when using 1500 hidden neurons with a learning rate of 0.001, the resultant average accuracy of 7 runs of the code is 91%.

	Target									
Predicted	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	87	0	0	0	1	0	0	0	0	0
1	0	79	2	0	0	2	0	0	1	4
2	0	0	80	3	0	0	0	0	1	0
3	1	0	0	85	0	1	1	2	0	1
4	0	1	0	0	90	1	0	0	0	0
5	0	2	0	2	1	89	1	0	1	2
6	2	1	0	0	0	1	79	0	0	0
7	1	1	0	0	0	0	0	81	0	2
8	0	7	1	1	2	0	0	1	85	1
9	1	0	0	9	0	7	0	2	2	71

Table 4.21: Confusion matrix of the hand-written digits dataset resulting from back-propagation algorithm

4.7 Observation of the computational complexity analysis of each algorithm

This project aims to implement a real-time embedded system capable of carrying out complex mathematical operations on large matrices developed by the feature extraction methods in chapter 3 and the classification algorithms observed in chapter 4. Moreover, as in chapter 3, the main parameters taken into consideration in the following computational complexity analysis are the time taken, and the memory consumed for each code of the algorithms to be executed. These results were obtained using the same Python functions mentioned in chapter 3 through the same computer.

In addition, the length of the input vector to the classification stage is fixed at 122 samples, so no more additions are added to it or subtracted from it. Also, several experiments have been carried out for SVM and random forests, and it was found that the type of kernel used for SVM and the Gini impurity value for random forest had to be kept fixed at Laplace RBF and 0.25 respectively. These two values were chosen to achieve an accuracy high enough to be valid to use in this thesis. As a result, the only parameter that was found to be valid for the computational complexity analysis is the proportion of the training files to the testing files. Moreover, the computational complexity analysis was carried out using the same methods used for the analysis used in section 3.5.

Based on figure 4.46, the time taken to execute the code is linearly proportional to the proportion of the training files to the testing files. Additionally, the time taken to execute the code when 90% of the data was used for training had increased by 38.47% compared to when 50% of the data was used for training. Also the the maximum accuracy obtained was as mentioned earlier 51.1% when 90% of the files used for training.

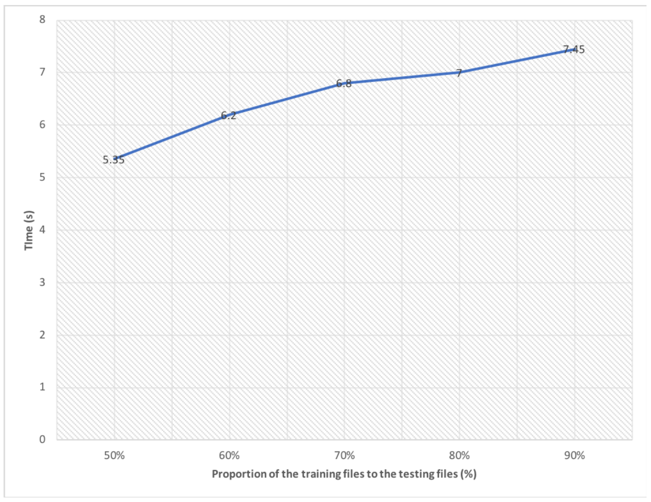


Figure 4.46: Time taken to execute SVM code Vs. The proportion of the training files to the testing files

On the other hand, the following graphs illustrates that Resident Set Size when 50% of the data used for training 100MB then increases gradually to reach equal to 108.781MB when the training files used reached 90%. Therefore, the amount of memory consumed had increased by 8.781%, when 90% of the data was used for training compared to when 50% was used for training; however, this increase is not considered substantial.

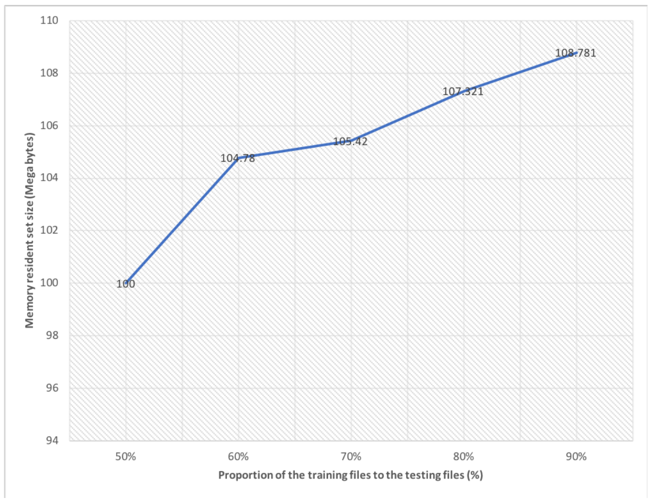


Figure 4.47: Resident Set Size equal Vs. The proportion of the training files to the testing files

Additionally, according to figure (4.48), the virtual memory size had also shown an increase from 408MB when 50% of the audio files were used for training until they reached 410.44MB when 70% of the files were used for training to remaining fixed at this amount when 80% of the files were used for training also. Finally, the amount of virtual memory size increased again to 413 MB when 90% of the files used for training. Therefore, there had been an increase of 0.62% of memory consumption when 90% of the data was used for training, which is not substantial.

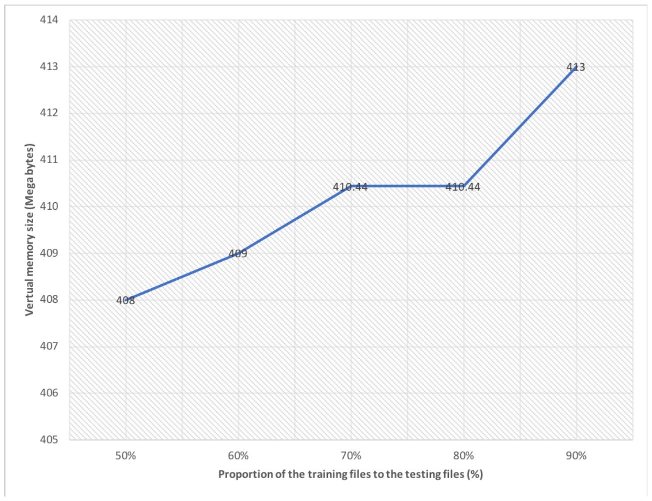


Figure 4.48: Virtual memory size Vs. The proportion of the training files to the testing files

Based on the following figure, executing the random forest code also shows a linear relationship between the number of files used for training and the time taken the code to be executed. Furthermore, the time of execution had increased by 27.43%, when 90% of the data was used for training compared to when 50% of the data used for training. Whereas in SVM, the time has increased by 38.47% under the same file proportions.

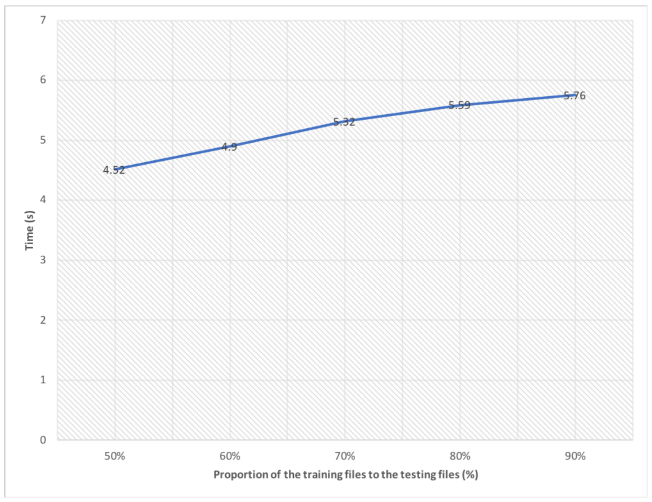


Figure 4.49: Time taken to execute the random forest code Vs. The proportion of the training files to the testing files

Furthermore, the resident set size memory consumed to execute the code had shown an increase from 108.2MB when 50% of the data was used for training to 110.22MB when 90% of the data was used for training. However, it is noted that the amount of resident size memory almost remained fixed when the number of training files was 70% and 80%. Therefore, the increase in memory consumption is around 2%, which is not substantial increase.

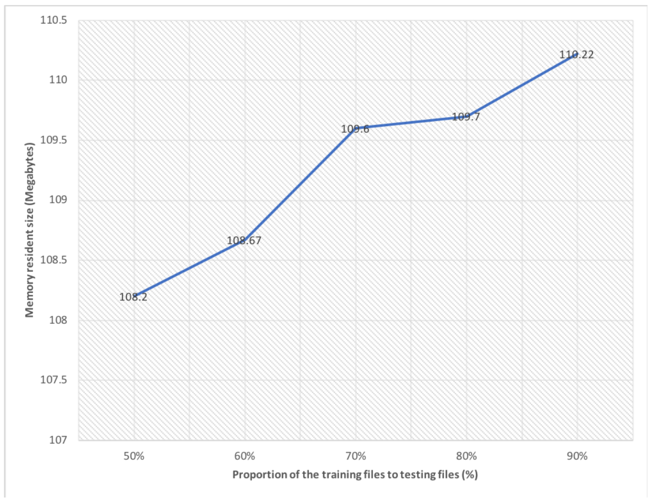


Figure 4.50: Resident Set Size equal Vs. The proportion of the training files to the testing files

It is also noted that the amount of virtual memory size has shown an overall increase as the amount of data used for training gets higher. It is also illustrated in figure 4.51 that the consumption of memory remains almost fixed at around 416MB when 50% and 60% of the data was used for training, then the amount of memory required increases rapidly to around 4.18MB. Then the amount of memory required remains almost fixed again at around 418.16MB. The latter shows an increase of 0.50% in memory consumption, which is not a significant increase.

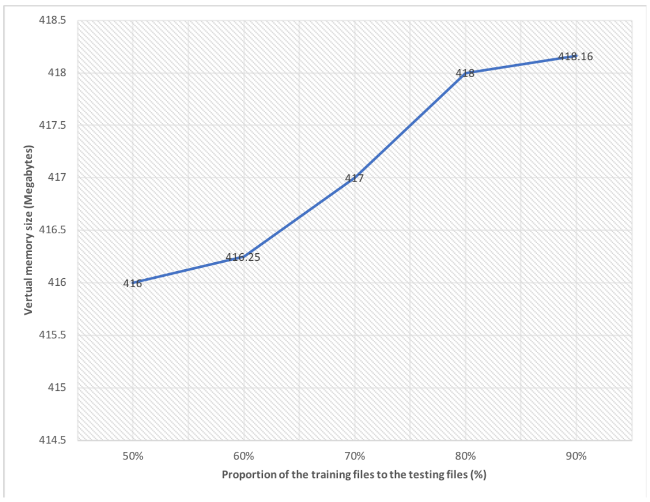


Figure 4.51: Virtual memory size Vs. The proportion of the training files to the testing files

On the other hand, when using neural networks based classification techniques, ELM was found to give the best accuracy, with parameters of 70% of the data used for training, 993 hidden neurons, and 100 experiments. As a result, these parameters were fixed for both ELM and back-propagation when the computational complexity analysis was carried out.

In terms of the time taken for the ELM code to be executed was 2.82s, whereas, for back-propagation, it was around 2 minutes. Therefore in back-propagation the time taken to execute the code had increased by 4155.32%. On the other hand, when the ELM code was executed, the amount of resident memory size, as well as virtual memory size consumed, was 105.22MB and 200.18MB, respectively. However, the back-propagation consumed around 500.43MB of resident memory size and 632MB of virtual memory size. Therefore, when the back-propagation algorithm was used, the memory consumed had increased by 375.603% and 215.716% for the resident memory size and virtual memory size respectively.

As a result, it is evident that ELM has a significant advantage over the other algorithms, as it has not only provided the best testing accuracy but is less computationally expensive. Therefore it is decided to use ELM for the implementation of the embedded system in chapter 5.

Chapter 5

Implementation of the Embedded System

This project aims to implement a real-time embedded system capable of carrying out the complex mathematical operations resulting from the feature extraction and classification stages. Computational power and battery life are the most critical aspects of the embedded system. As a result, it is vital that the number of neurons be low to keep the computational power of the system as less expansive as possible. As a result, S-TDSC and ELM have proven to be the least computationally expensive in the feature extraction and classification stages in section 3.5 and section 4.7 respectively.

Due to the computational complexity resulting from executing the code of S-TDSC and the ELM classification algorithm, constrictions were raised in terms of choosing an appropriate micro-controller. The latter has to record audio and implement a real-time embedded system that can carry out complex mathematical operations on the large matrices developed from the feature extraction and classification stages with the highest accuracy.

For example, when considering Arduino Uno and Mbed to execute S-TDSC and ELM codes, both micro-controllers crash within seconds of executing the code at the feature extraction technique without even going to the classification stage. This crash is because, as shown in section 3.5, S-TDSC requires 111.84128MB of resident memory size and 4199.84MB of virtual memory size. Therefore, it is not possible to use either Arduino Uno with specifications of 32KB of Flash memory and 2KB of SRAM or Mbed with specifications of 8kB RAM and 32KB.

Additionally, before building the embedded system, an experiment was conducted to simulate the embedded system through a Matlab code with a graphical user interface tool (GUI). The latter is created to perform a binary classification (bumblebee classifier), with 50% of the data used for training, and the number of epochs is 100 without shuffling the training and testing data as shown in the last experiment in section (4.5.2). The graphical user interface tool (GUI) in Matlab allows any user from any intellectual background to upload an audio recording made in the field, and the number of the class will be shown. For instance, the following graph shows that when a recording of a bumblebee is selected, and the ELM method chosen as a classifier, S- TDSC is used to extract the features from the selected audio recording, the result is 1, referring to the class to which bumblebees belong.

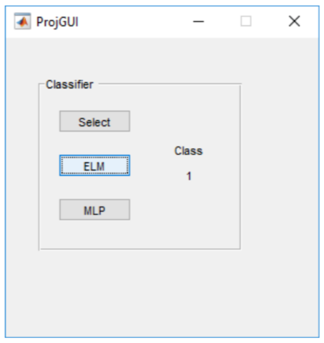


Figure 5.1: Classifier of extreme learning machines (ELM) and multilayer perceptron (MLP) using Matlab where if the result is class 1 then the audio is a bumblebees and if the result is class 2 then the pollinator is either hoverfly, common wasp or a solitary

However, if an audio recording of a hoverfly, common wasp or a solitary bee is selected, the class number appears as 2.

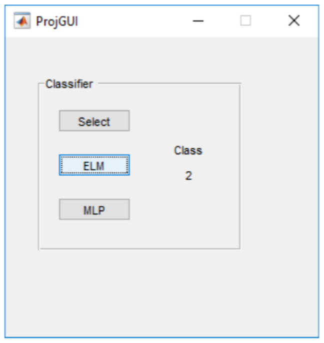


Figure 5.2: Classifier of extreme learning machines (ELM) and multilayer perceptron (MLP) using Matlab where if the result is class 1 then the audio is a bumblebees and if the result is class 2 then the pollinator is either hoverfly, common wasp or a solitary

At this point, the simulation of the embedded system was conducted by

Matlab. Therefore the only challenge left is to find the most appropriate microcontroller that is capable of handling the 111.84128MB of resident memory size and 4199.84MB of virtual memory size to perform S-TDSC as well as resident memory of 105.22MB and virtual memory size of 200.18MB to perform ELM. As a result, Raspberry Pi was found to be the most suitable board to carry out all the tasks required for this project for several reasons. Firstly, Raspberry Pi has a quad-core processor, a more powerful processor than the ATmega328P of Arduino and the ARM Cortex-M3 of Mbed. Secondly, Raspberry Pi has a 1 GB RAM, which is significantly larger than the 32KB RAM of Arduino and Mbed. Hence, Raspberry Pi is perfect for implementing the feature extraction and the classification algorithm, as the large matrices multiplication required by S-TDSC and ELM are not computationally expensive to perform. Thirdly, the small size of the board makes it very easy to embed it into any embedded system. Fourthly, Raspberry Pi has 40 GPIO pins; therefore, it is easily connected to various sensors, and it has an audio-in port for a microphone to carry out the audio recordings. Finally, this board can be connected to other hardware devices using many protocols such as serial, spi, and i2c, which is very important as it makes Raspberry Pi compatible with most devices connected to it. Lastly, Raspberry Pi is a cheap board as its only around 30£.

Figure 5.3: Raspberry Pi

When the embedded system was implemented for this project, the sounds of the species were simulated using a speaker and rerecorded using the TASCAM microphone, which is the same device used for recording the sounds from the field in the first place. Then when all the audio recordings for the bee classifier were rerecorded, the audio recordings saved in the SD card of the TACAM microphone were inputted into the Raspberry Pi to carry out the feature extraction and the classification. Furthermore, after the features were extracted from the rerecorded audio signals using S-TDSC, ELM was implemented with 1500 hidden layer neurons, while 50% of the data was used for training the algorithm. The result was found to be almost 70% accuracy, which is almost the same result obtained when training and testing ELM using the original recordings. The following confusion matrix is developed after classification is carried out using the rerecorded audio signals.

	Target	
Predicted	Bumblebees	Another species
Bumblebees	39	10
Another species	13	22

Table 5.1: Confusion matrix when the number of hidden neurons is 1500 and the number of epochs is 100

Figure 5.4: TASCAM audio recorder

The reason for using TASCAM as an audio recording device is because the quality of the sound captures using it is very good and almost the same to the original audio file in terms of the characteristics of both signals. However, when feeding in the audio signal directly to the Raspberry Pi using a cheap USB microphone, there was a considerable amount of noise included with the rerecorded audio signal.

Figure 5.5: USB Microphone connected to Raspberry Pi

Finally, this embedded system has been deployed under laboratory conditions; however, in terms of field deployment, there are other alternatives in hardware components that could be less expensive with just the computational power needed to carry out the testing processes. For example, instead of using the Raspberry pi Three, as shown in figure 5.5, the Raspberry Pi Zero could be used, which has a 1GHz single-core CPU and 512MB RAM. The results were not compared with benchmarks such as existing systems to quantify the relative performance, due to the time limitation of this thesis.

5.1 Summery

This chapter illustrated an embedded system consisting of a microphone and Raspberry PI model 3. The sound of flight is simulated using a speaker and captured using two different microphones to evaluate which microphone captures a better quality of sound. The sound is then fed to the Raspberry Pi in order to implement the feature extraction using the S-TDSC technique and classification using the ELM algorithm. The experiment carried out in this chapter is the bumblebee classifier using the same data set used in chapter 4, and the result achieved is 70% accuracy. It is also found that for applications that require the combination of hardware and artificial intelligence algorithms to perform classification, Raspberry PI is the more applicable than most typical micro-controllers such as Arduino and Mbed. This is due to the processing power that Raspberry PI acquires, as it can manipulate matrices of significant size and as well as interface sensors simultaneously.

Chapter 6

Conclusion and future work

There are several benefits that are provided by pollination on agriculture and ecosystem, and as pollinators are declining every year, this has caused concern worldwide. The work that has already been done in pollinator species identification requires a professional with a trained eye to determine the type of the species. These typical methods also require the pollinator species to be killed in order to be identified. Hence, this project aims to establish if an embedded system could be implemented to identify pollinator species using the combination of bio-acoustics and machine learning. This thesis demonstrates that when the sound of the flight of pollinators is captured, an enhanced TDSC technique and the application of ANNs can indeed accomplish identifying four different pollinator species with an accuracy reaching 58%.

More specifically, in chapter 1, this thesis reviews the current knowledge on pollinator decline and provides an in-depth analysis of the various causes of pollinator decline. Moreover, this chapter also describes the economic and ecological importance of insect pollinators in detail, providing several examples as to how pollinator species enhance the world economically and biologically. After discussing the economic importance of pollinator species, chapter 1 provides a history of pollinator decline in many countries such as the UK and the Netherlands. Finally, this chapter introduces pollinator monitoring, which includes the typical pollinator monitoring methods, such as pan traps and net sampling, indicating the significant disadvantages of these two methods. Hence, the need for monitoring and identifying pollinator species based on bio-acoustics.

Chapter 2 starts with an in-depth background on national pollinator monitoring in the UK, which is followed by a more detailed observation on the common monitoring methods that were previously introduced in chapter 1, as well as the advantages and disadvantages of these methods. This is then immediately followed by one of the core principles of this project, which is bio-acoustics, where a more detailed background on it than in chapter 1 is provided. Many examples from other projects follow this background were bio-acoustics and machine learning are combined to identify various species such as bats, frogs, and birds. In addition, this chapter provides a detailed biological discussion of the four pollinator species used in this project: honey bees, bumblebees, common wasps, and hoverflies. This observation includes information such as an approximation of how many sub-species are found under each of these species, the anatomy of each species and data on their recorded frequency of sounds under different situations such as when flying staying still.

Chapter 3 starts with the description of the raw data, which was collected by Thomas Dally, who was a PhD student at the University of Leeds. This description included where and when the audio files were collected and the quality of the collected data. Then this chapter introduces TDSC and its applications to identify insects in the past, then showes the D-matrix technique, which is a method that is based on TDSC. This led to the development of the novel technique S-TDSC, which eliminates the need of codebooks and allows any audio signal to only be codded to 122 samples. In addition, various frequency-domain methods were observed, and one of them was chosen to be implemented to extract the features from the audio files for the sake of comparison between it and S-TDSC. The latter has shown a clear advantage over the frequency domain-based method, hence a section of why choosing time domain over frequency domain in this project was provided. This analysis finishes with a computational complexity analysis of S-TDSC and the frequency domain based method.

Chapter 4 provides an observation of 4 different machine learning techniques, which are SVM, random forests, ELM, and back-propagation. Each of these algorithms was observed and implemented as a classification technique on the 122 features extracted from S-TDSC. The audio recording used to train and test each algorithm was the same, containing audio files from the four different pollinators specie, which are bumblebees, hoverflies, common wasps, and solitary bees. ELM proved to give maximum accuracy to about 76.57%. Therefore in the ELM section, more experiments were carried out, where various parameters were altered to increase the accuracy of this algorithm. Furthermore, due to the lack of data to train the algorithms, which has resulted in each of these algorithms being insufficiently tested, dummy data was used to verify the performance of the algorithms. This dummy data consists of samples of hand-written digits that include 1797 images. As a result, the performances of the algorithms were verified and the testing accuracy of every algorithm was above 90%. Additionally, as ELM showed the highest accuracy when it was implemented on the pollinator specie audio files, ELM has also has provided the highest accuracy in the hand-written digits data. Additionally, in the ELM section, a bumblebee classifier using ELM was implemented, where the audio files from hoverflies, common wasps, and solitary bees against the bumblebees audio files and the accuracy reached 70%. an in-depth computational complexity analysis of each algorithm is provided in the sub-chapter (4.7). Since one of the main arguments of this thesis is the low computational complexity required to implement the embedded system, this section has shown that ELM is the most valid choice of classification. This is because apart from the fact that ELM provides the best testing accuracy, it is also the least computationally complex out the 4 algorithms observed in this thesis.

Finally, chapter 5 illustrated the implementation of the embedded system that captures the audio files using a microphone and a Raspberry Pi 3 to perform S-TDSC to extract the features from the captured audio signals and performs ELM for the classification and the result in the form of a confusion matrix was provided. In conclusion, the research presented in this thesis has constructed the tools, both in software and hardware. This allows the identification of pollinator species to occur in a desktop computer after the sound has been collected in a field or placing the embedded system in a field to detect the presence of pollinator species as classify it in real-time.

6.1 Future work

This thesis has already proven that the more data available for training, the better the classification accuracy of the algorithm. Therefore, the most critical work that should be done to enhance this project is to increase the accuracy of the algorithm by collecting more data to train the system. Furthermore, this data can include different genus recording so the classification can go beyond the species level and to the genus level within the same species, for instance, there are several bumblebee species, such as buff-tailed bumblebee (bombus terrestris), southern cuckoo bumblebee (bombus vestalis), white-tailed bumblebee (bombus lucorum) and gypsy cuckoo bumblebee (bombus bohemicus). The following figure illustrates the male, worker, and queen species of each sub-species as well as the UK's location of where they are abundant [141]. Another alternative of classification is to identify species based on gender by training the algorithm with audio files of male and female species.

Audio recordings could also be recorded in laboratory conditions to examine if the sound of flight would change depending on some conditions. For example, exposing species to simulated pollution such as CO2 emission to check if species would react differently in terms of sound signals when exposed. The latter approach could give an insight into how to improve the testing accuracy further.

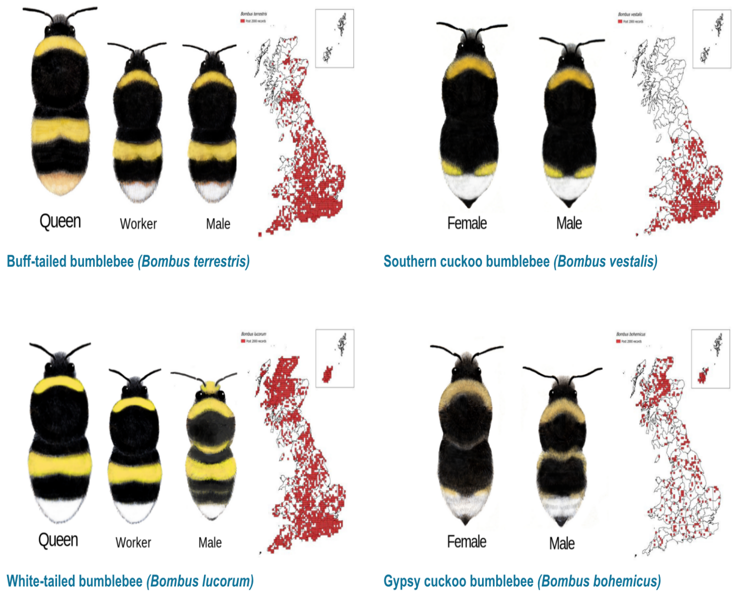

Queen
Worker
Male
Buff-tailed bumblebee (Bombus terrestris)
Female
Male
Southern cuckoo bumblebee (Bombus vestalis)
Queen
Worker
Male
White-tailed bumblebee (Bombus lucorum)
Female
Male
Gypsy cuckoo bumblebee (Bombus bohemicus)

Figure 6.1: Sub-species of bumblebees [139]

Moreover, another significant improvement would be implementing deep learning neural networks in the training phase to observe if there would be any improvement in the testing accuracy. Some deep learning neural network structures are observed in sub-section 4.3.2. Additionally, RNN and CNN are both deep learning techniques that can be implemented as a classification methods for this project. As with all other deep learning algorithms, their accuracy would increase when more data is provided for training as the following figure illustrates.

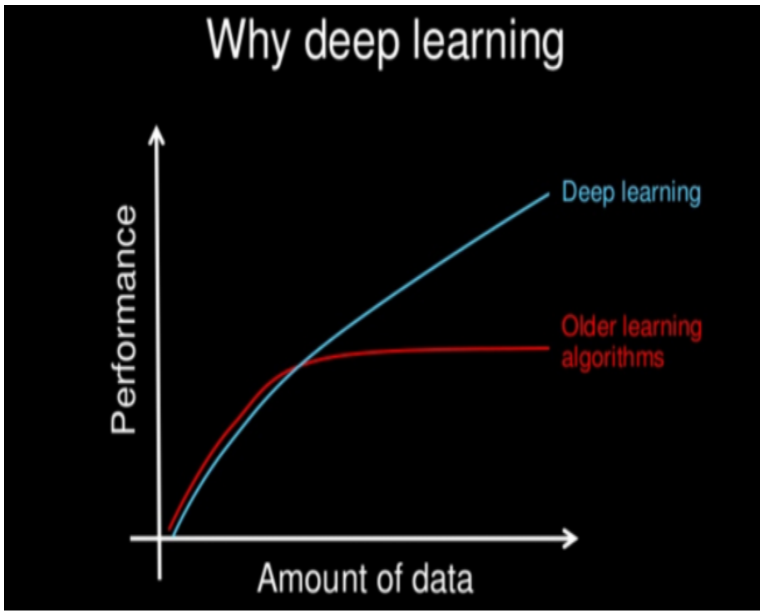


Figure 6.2: Amount of data available for training Vs. Accuracy [142]

Research has shown that RNN is ideal for text and speech analysis, whilst CNN is ideal for images and video processing [142]. However, unless RNN and CNN are built, trained and tested with the dataset of this thesis, it is not possible to provide a computational complexity analysis.

Furthermore, one of the project's motives is to implement an embedded system that is entirely independent of human intervention. However, this embedded system still requires someone to take it to a field and collect it afterward. Therefore, the embedded system to be enhanced further is to create a GPS guided robot such as a four-wheeled robot or a drone. This robot should be designed to be able to carry an embedded system from point A to point B based on the GPS coordinates and back to point B when the data has been collected and analysed. Finally, for such robots to be capable of taking measurements is considerably large fields, there could be fixed stations along the route to provide larger batteries.

Since the collection method of the audio files was made through following bees around, whilst placing the microphone reasonably close to the species. The sound quality is considerably high. However, after the algorithms were trained using the collected high-quality audio files, the trained embedded system is placed in a field to classify pollinator species. When the embedded system is in a field, several issues may arise in many scenarios. For instance, wind noise can significantly impact the classification accuracy; noises can be in the form of a flying plane, a car passing by or wind. Therefore, filters must be observed for the pre-processing stage.

Additionally, more audio recordings should be made with noise embedded in the audio files and train the classification algorithm to inspect if the noise source would be classified independently by the algorithm.

For instance, inspecting whether the wind would be identified as a class of its own or not. Also, analysing how the testing accuracy is changed in case wind was classified as an independent class.

Another challenge with placing the embedded system in a field would be the same bee passing continuously, and the embedded system keeps on counting it more than one time. Moreover, an issue can also arise if more than just one bee of the same species would pass by the microphone range, resulting in an overlap of the classification. On the other hand, these challenges are easily avoided with the destructive method of classifying bees, as the bee is already dead under a microscope when it is identified and will not be identified twice.

Furthermore, As shown in this thesis, pollinator species are essential for our ecosystem, and as these species are declining, more monitoring is required. Hence, using the work implemented in this project should not be restricted to scientists only. However, the involvement of citizen scientists is also a viable option. Further research may also performed on an incentive mechanism, which can make people participate. The participation can be through the involvement of local schools, tourist information centres and park authorities interested in attracting customers while raising public awareness of biodiversity monitoring issues. Gamification that is the engagement of users through a game in a none-game context is also a route that can be followed because hundreds of millions of people globally play electronic games [142]; hence it is a large community to attract. Additionally, the computational sustainability scenario attracts much public attention when more importance is shown in safeguarding the ecosystem.

Moreover, the feature extraction and classification techniques implemented for this project can provide an excellent starting point for a mobile application project. This is because the weights of the neural network and the feature extraction technique have already been computed. Therefore, a mobile phone that the public can easily carry to fields can help gather more audio data, which has been a hurdle for this project and help monitor pollinator species. Modern mobile phones would be sensitive enough to capture bee sound as there are already existing mobile applications to identify cicada species.

Finally, the same techniques and algorithm used in this research may be applied outside the biodiversity monitoring domain by using citizen science and smartphones to monitor other environmental factors using sound. An example application would be monitoring soundscape and tranquillity around urban parks or detecting faults in electrical equipment, such as alternators, that emit distinctive noise when close to failure.

Appendix A

Confusion matrices developed from SVM for the 4 species classifier

When 60% of the files used for training, the maximum accuracy of the run is 52%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	18	5	0	1
Solitary bees	2	7	3	1
Bumblebees	5	6	1	0
Hover flies	7	2	0	10

Table A.1: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

When 70% of the files used for training, the maximum accuracy of the run is 57%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	7	1	0	0
Solitary bees	2	8	2	0
Bumblebees	3	3	13	1
Hover flies	0	5	5	1

Table A.2: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

When 80% of the files used for training, the maximum accuracy of the run is 58.8%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	9	0	2	0
Solitary bees	0	3	3	1
Bumblebees	1	2	7	0
Hover flies	3	0	2	1

Table A.3: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

When 90% of the files used for training, the maximum accuracy of the run is 58.9%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	5	0	0	0
Solitary bees	0	0	0	0
Bumblebees	0	1	4	0
Hover flies	3	0	3	1

Table A.4: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

Confusion matrices are obtained using the frequency domain method as a feature extraction technique. When 50% of the files are used for training, the maximum accuracy of the specific run is 39.8%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	17	7	2	1
Solitary bees	14	21	6	5
Bumblebees	4	4	2	4
Hover flies	11	6	1	3

Table A.5: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

When 60% of the files are used for training, the maximum accuracy of the specific run is 40%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	2	2	2	4
Solitary bees	1	5	4	11
Bumblebees	2	5	11	3
Hover flies	4	6	8	17

Table A.6: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

When 70% of the files are used for training, the maximum accuracy of the specific run is 45%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	18	7	4	0
Solitary bees	8	2	4	2
Bumblebees	6	1	9	0
Hover flies	1	1	2	0

Table A.7: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

When 80% of the files are used for training, the maximum accuracy of the specific run is 43%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	18	7	4	0
Solitary bees	8	2	4	2
Bumblebees	6	1	9	0
Hover flies	1	1	2	0

Table A.8: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

When 90% of the files are used for training, the maximum accuracy of the specific run is 54%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	1	2	1	1
Solitary bees	0	5	0	0
Bumblebees	0	0	0	1
Hover flies	5	0	0	6

Table A.9: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

Appendix B

Confusion matrices developed from Random forests for the 4 species classifier

When 60% of the files used for training, the maximum accuracy of the run is 57%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	12	1	5	1
Solitary bees	2	0	6	0
Bumblebees	1	1	21	2
Hover flies	4	0	6	6

Table B.1: Confusion matrix when 60% of the data used for training

When 70% of the files used for training, the maximum accuracy of the run is 65%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	1	3	0	2
Solitary bees	1	22	1	1
Bumblebees	0	4	6	2
Hover flies	1	2	1	4

Table B.2: Confusion matrix when 70% of the data used for training

When 80% of the files used for training, the maximum accuracy of the run is 68%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	0	0	2	1
Solitary bees	1	8	0	1
Bumblebees	1	2	9	0
Hover flies	0	1	3	6

Table B.3: Confusion matrix when 80% of the data used for training

When 90% of the files used for training, the maximum accuracy of the run is 70%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	7	0	1	0
Solitary bees	1	0	1	0
Bumblebees	1	0	4	0
Hover flies	1	0	0	1

Table B.4: Confusion matrix when 90% of the data used for training

The following are the confusion matrices obtained using the frequency domain method as a feature extraction technique. When 50% of the files are used for training, the maximum accuracy of the specific run is 48%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	5	0	13	2
Solitary bees	4	2	4	5
Bumblebees	7	1	32	7
Hover flies	4	0	9	13

Table B.5: Confusion matrix when 60% of the data used for training

When 60% of the files are used for training, the maximum accuracy of the specific run is 46%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	27	9	2	4
Solitary bees	5	11	0	4
Bumblebees	7	2	1	1
Hover flies	5	5	3	1

Table B.6: Confusion matrix when 60% of the data used for training

When 70% of the files are used for training, the maximum accuracy of the specific run is 46%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	9	4	1	2
Solitary bees	2	3	0	3
Bumblebees	2	5	1	4
Hover flies	5	5	2	17

Table B.7: Confusion matrix when 70% of the data used for training

When 80% of the files are used for training, the maximum accuracy of the specific run is 50%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	15	1	0	0
Solitary bees	4	5	0	1
Bumblebees	6	1	2	0
Hover flies	8	1	0	0

Table B.8: Confusion matrix when 80% of the data used for training

When 90% of the files are used for training, the maximum accuracy of the specific run is 68%:

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	9	0	0	1
Solitary bees	3	4	0	0
Bumblebees	1	0	1	0
Hover flies	2	0	0	1

Table B.9: Confusion matrix when 90% of the data used for training

Appendix C

Confusion matrices developed from ELM when both of the training and testing data are unshuffled for the 4 species

Number of neurons = 60

- Training accuracy 80.0%
- Testing accuracy 69.04%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	25	4	1	0
Solitary bees	4	30	0	0
Bumblebees	4	2	3	0
Hover flies	6	4	1	0

Table C.1: Confusion matrix when when the number of the hidden neurons is 60 and the number of epochs is a 100

Number of neurons = 70

- Training accuracy 80.52%
- Testing accuracy 73.80%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	30	5	2	0
Solitary bees	1	30	0	0
Bumblebees	2	1	2	0
Hover flies	6	4	1	0

Table C.2: Confusion matrix when when the number of the hidden neurons is 70 and the number of epochs is a 100

Number of neurons = 80

- Training accuracy 74.11%
- Testing accuracy 76.19%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	31	5	1	0
Solitary bees	0	30	0	0
Bumblebees	1	1	3	0
Hover flies	7	4	1	0

Table C.3: Confusion matrix when when the number of the hidden neurons is 80 and the number of epochs is a 100

Number of neurons = 90

- Training accuracy 87.05%
- Testing accuracy 78.57%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	33	5	2	0
Solitary bees	4	31	1	0
Bumblebees	0	0	2	0
Hover flies	2	4	0	0

Table C.4: Confusion matrix when when the number of the hidden neurons is 90 and the number of epochs is a 100

Number of neurons = 100

- Training accuracy 90.05%
- Testing accuracy 93.75%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	15	0	0	0
Solitary bees	0	0	0	0
Bumblebees	1	0	0	0
Hover flies	0	0	0	0

Table C.5: Confusion matrix when when the number of the hidden neurons is 100 and the number of epochs is a 100

Number of neurons = 110

- Training accuracy 92.94%
- Testing accuracy 82.14%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	32	0	2	0
Solitary bees	3	35	0	0
Bumblebees	0	1	2	0
Hover flies	4	4	1	0

Table C.6: Confusion matrix when when the number of the hidden neurons is 110 and the number of epochs is a 100

Number of neurons = 120

- Training accuracy 90.58%
- Testing accuracy 82.14%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	31	1	3	0
Solitary bees	4	36	0	0
Bumblebees	0	0	2	0
Hover flies	4	3	0	0

Table C.7: Confusion matrix when when the number of the hidden neurons is 120 and the number of epochs is a 100

Number of neurons = 130

- Training accuracy 88.23%
- Testing accuracy 77.38%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	30	4	1	0
Solitary bees	4	32	0	0
Bumblebees	1	2	3	0
Hover flies	4	2	1	0

Table C.8: Confusion matrix when when the number of the hidden neurons is 130 and the number of epochs is a 100

Number of neurons = 140

- Training accuracy 91.76%
- Testing accuracy 77.38%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	29	2	1	0
Solitary bees	3	34	1	0
Bumblebees	4	3	2	0
Hover flies	3	1	1	0

Table C.9: Confusion matrix when when the number of the hidden neurons is 140 and the number of epochs is a 100

Number of neurons = 150

- Training accuracy 96.47%
- Testing accuracy 82.14%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	34	4	1	0
Solitary bees	0	31	0	0
Bumblebees	1	2	4	0
Hover flies	4	3	0	0

Table C.10: Confusion matrix when when the number of the hidden neurons is 150 and the number of epochs is a 100

Number of neurons = 160

- Training accuracy 96.47%
- Testing accuracy 80.95%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	34	3	0	0
Solitary bees	0	30	1	0
Bumblebees	1	2	4	0
Hover flies	4	5	0	0

Table C.11: Confusion matrix when when the number of the hidden neurons is 160 and the number of epochs is a 100

Number of neurons = 170

- Training accuracy 95.29%
- Testing accuracy 79.76%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	31	5	1	0
Solitary bees	5	33	0	0
Bumblebees	2	0	3	0
Hover flies	1	2	1	0

Table C.12: Confusion matrix when when the number of the hidden neurons is 170 and the number of epochs is a 100

Number of neurons = 180

- Training accuracy 97.64%
- Testing accuracy 83.33%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	33	3	0	0
Solitary bees	3	33	1	0
Bumblebees	0	0	4	0
Hover flies	3	4	0	0

Table C.13: Confusion matrix when when the number of the hidden neurons is 180 and the number of epochs is a 100

Number of neurons = 190

- Training accuracy 92.94%
- Testing accuracy 80.95%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	33	0	2	0
Solitary bees	1	33	0	0
Bumblebees	0	2	2	0
Hover flies	5	5	1	0

Table C.14: Confusion matrix when when the number of the hidden neurons is 190 and the number of epochs is a 100

Appendix D

Confusion matrices developed from ELM when only the training data are is being shuffled for the 4 species

Number of neurons = 60

- Training accuracy 68.06%
- Testing accuracy 86.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	33	1	0	0
Solitary bees	3	10	0	0
Bumblebees	0	0	0	0
Hover flies	3	0	0	0

Table D.1: Confusion matrix when when the number of the hidden neurons is 60 and the number of epochs is a 100

Number of neurons = 70

- Training accuracy 73.10%
- Testing accuracy 88.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	34	0	0	0
Solitary bees	2	10	0	0
Bumblebees	0	0	0	0
Hover flies	3	1	0	0

Table D.2: Confusion matrix when when the number of the hidden neurons is 70 and the number of epochs is a 100

Number of neurons = 80

- Training accuracy 72.26%
- Testing accuracy 84.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	32	1	0	0
Solitary bees	1	10	0	0
Bumblebees	2	0	0	0
Hover flies	4	0	0	0

Table D.3: Confusion matrix when when the number of the hidden neurons is 80 and the number of epochs is a 100

Number of neurons = 90

- Training accuracy 79.83%
- Testing accuracy 90.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	34	0	0	0
Solitary bees	1	11	0	0
Bumblebees	2	0	0	0
Hover flies	2	0	0	0

Table D.4: Confusion matrix when when the number of the hidden neurons is 90 and the number of epochs is a 100

Number of neurons = 100

- Training accuracy 84.87%
- Testing accuracy 88.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	34	0	0	0
Solitary bees	2	10	0	0
Bumblebees	0	0	0	0
Hover flies	3	1	0	0

Table D.5: Confusion matrix when when the number of the hidden neurons is 100 and the number of epochs is a 100

Number of neurons = 110

- Training accuracy 83.19%
- Testing accuracy 90.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	37	2	0	0
Solitary bees	0	8	0	0
Bumblebees	1	0	0	0
Hover flies	1	1	0	0

Table D.6: Confusion matrix when when the number of the hidden neurons is 110 and the number of epochs is a 100

Number of neurons = 120

- Training accuracy 90.58%
- Testing accuracy 82.14%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	31	1	3	0
Solitary bees	4	36	0	0
Bumblebees	0	0	2	0
Hover flies	4	3	0	0

Table D.7: Confusion matrix when when the number of the hidden neurons is 120 and the number of epochs is a 100

Number of neurons = 130

- Training accuracy 88.23%
- Testing accuracy 77.38%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	30	4	1	0
Solitary bees	4	32	0	0
Bumblebees	1	2	3	0
Hover flies	4	2	1	0

Table D.8: Confusion matrix when when the number of the hidden neurons is 130 and the number of epochs is a 100

Number of neurons = 140

- Training accuracy 87.39%
- Testing accuracy 92.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	37	1	0	0
Solitary bees	0	9	0	0
Bumblebees	2	0	0	0
Hover flies	0	1	0	0

Table D.9: Confusion matrix when when the number of the hidden neurons is 140 and the number of epochs is a 100

Number of neurons = 150

- Training accuracy 94.95%
- Testing accuracy 88.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	34	0	0	0
Solitary bees	2	10	0	0
Bumblebees	0	1	0	0
Hover flies	3	0	0	0

Table D.10: Confusion matrix when when the number of the hidden neurons is 150 and the number of epochs is a 100

Number of neurons = 160

- Training accuracy 93.27%
- Testing accuracy 94.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	38	1	0	0
Solitary bees	0	9	0	0
Bumblebees	0	0	0	0
Hover flies	1	1	0	0

Table D.11: Confusion matrix when when the number of the hidden neurons is 160 and the number of epochs is a 100

Number of neurons = 170

- Training accuracy 89.07%
- Testing accuracy 94.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	36	0	0	0
Solitary bees	1	11	0	0
Bumblebees	1	0	0	0
Hover flies	1	0	0	0

Table D.12: Confusion matrix when when the number of the hidden neurons is 170 and the number of epochs is a 100

Number of neurons = 180

- Training accuracy 94.95%
- Testing accuracy 94.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	36	0	0	0
Solitary bees	1	11	0	0
Bumblebees	1	0	0	0
Hover flies	1	0	0	0

Table D.13: Confusion matrix when when the number of the hidden neurons is 180 and the number of epochs is a 100

Number of neurons = 190

- Training accuracy 98.31%
- Testing accuracy 96.0%

	Target			
Predicted	Common wasps	Solitary bees	Bumblebees	Hover flies
Common wasps	38	0	0	0
Solitary bees	0	10	0	0
Bumblebees	1	0	0	0
Hover flies	0	1	0	0

Table D.14: Confusion matrix when when the number of the hidden neurons is 190 and the number of epochs is a 100

Appendix E

Confusion matrices developed from ELM when both the training and testing data for the bumblebee classifier

Number of neurons = 100

- Training accuracy 90.58%
- Testing accuracy 66.66%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	41	18
Another species	9	16

Table E.1: Confusion matrix when the number of hidden neurons is 100 and the number of epochs is 100

Number of neurons = 150

- Training accuracy 94.11%
- Testing accuracy 67.85%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	42	20
Another species	7	15

Table E.2: Confusion matrix when the number of hidden neurons is 150 and the number of epochs is 100

Number of neurons = 200

- Training accuracy 96.47%
- Testing accuracy 69.04%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	34	12
Another species	14	24

Table E.3: Confusion matrix when the number of hidden neurons is 200 and the number of epochs is 100

Number of neurons = 250

- Training accuracy 96.47%
- Testing accuracy 71.42%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	42	18
Another species	6	18

Table E.4: Confusion matrix when the number of hidden neurons is 250 and the number of epochs is 100

Number of neurons = 300

- Training accuracy 98.82%
- Testing accuracy 71.42%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	36	10
Another species	14	24

Table E.5: Confusion matrix when the number of hidden neurons is 300 and the number of epochs is 100

Number of neurons = 350

- Training accuracy 98.82%
- Testing accuracy 69.04%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	43	20
Another species	6	15

Table E.6: Confusion matrix when the number of hidden neurons is 350 and the number of epochs is 100

Number of neurons = 400

- Training accuracy 97.64%
- Testing accuracy 70.23%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	42	13
Another species	12	17

Table E.7: Confusion matrix when the number of hidden neurons is 400 and the number of epochs is 100

Number of neurons = 450

- Training accuracy 98.82%
- Testing accuracy 67.85%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	35	16
Another species	11	22

Table E.8: Confusion matrix when the number of hidden neurons is 450 and the number of epochs is 100

Number of neurons = 500

- Training accuracy 100.00%
- Testing accuracy 71.42%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	37	15
Another species	9	23

Table E.9: Confusion matrix when the number of hidden neurons is 500 and the number of epochs is 100

Number of neurons = 550

- Training accuracy 100.00%
- Testing accuracy 71.42%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	41	12
Another species	12	19

Table E.10: Confusion matrix when the number of hidden neurons is 550 and the number of epochs is 100

Number of neurons = 600

- Training accuracy 98.82%
- Testing accuracy 73.80%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	40	11
Another species	11	22

Table E.11: Confusion matrix when the number of hidden neurons is 600 and the number of epochs is 100

Number of neurons = 650

- Training accuracy 100.00%
- Testing accuracy 75.0%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	38	9
Another species	12	25

Table E.12: Confusion matrix when the number of hidden neurons is 650 and the number of epochs is 100

Number of neurons = 700

- Training accuracy 100.00%
- Testing accuracy 73.80%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	42	9
Another species	13	20

Table E.13: Confusion matrix when the number of hidden neurons is 700 and the number of epochs is 100

Number of neurons = 750

- Training accuracy 100.0%
- Testing accuracy 69.04%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	39	16
Another species	10	19

Table E.14: Confusion matrix when the number of hidden neurons is 750 and the number of epochs is 100

Number of neurons = 800

- Training accuracy 100.0%
- Testing accuracy 69.04%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	38	17
Another species	9	20

Table E.15: Confusion matrix when the number of hidden neurons is 800 and the number of epochs is 100

Number of neurons = 850

- Training accuracy 98.82%
- Testing accuracy 72.61%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	37	13
Another species	10	24

Table E.16: Confusion matrix when the number of hidden neurons is 850 and the number of epochs is 100

Number of neurons = 900

- Training accuracy 100.0%
- Testing accuracy 75.0%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	43	17
Another species	4	20

Table E.17: Confusion matrix when the number of hidden neurons is 900 and the number of epochs is 100

Number of neurons = 950

- Training accuracy 100.0%
- Testing accuracy 69.04%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	35	10
Another species	16	23

Table E.18: Confusion matrix when the number of hidden neurons is 950 and the number of epochs is 100

Number of neurons = 1000

- Training accuracy 100.0%
- Testing accuracy 76.19%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	44	15
Another species	5	20

Table E.19: Confusion matrix when the number of hidden neurons is 1000 and the number of epochs is 100

Number of neurons = 1050

- Training accuracy 100.0%
- Testing accuracy 76.19%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	44	15
Another species	5	20

Table E.20: Confusion matrix when the number of hidden neurons is 1050 and the number of epochs is 100

Number of neurons = 1100

- Training accuracy 100.0%
- Testing accuracy 72.61%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	39	10
Another species	13	22

Table E.21: Confusion matrix when the number of hidden neurons is 1100 and the number of epochs is 100

Number of neurons = 1150

- Training accuracy 98.82%
- Testing accuracy 75.0%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	40	8
Another species	13	23

Table E.22: Confusion matrix when the number of hidden neurons is 1150 and the number of epochs is 100

Number of neurons = 1200

- Training accuracy 100.0%
- Testing accuracy 72.61%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	40	13
Another species	10	21

Table E.23: Confusion matrix when the number of hidden neurons is 1200 and the number of epochs is 100

Number of neurons = 1250

- Training accuracy 100.0%
- Testing accuracy 73.80%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	42	6
Another species	16	20

Table E.24: Confusion matrix when the number of hidden neurons is 1250 and the number of epochs is 100

Number of neurons = 1300

- Training accuracy 100.0%
- Testing accuracy 76.19%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	41	10
Another species	10	23

Table E.25: Confusion matrix when the number of hidden neurons is 1300 and the number of epochs is 100

Number of neurons = 1350

- Training accuracy 100.0%
- Testing accuracy 70.23%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	39	14
Another species	11	20

Table E.26: Confusion matrix when the number of hidden neurons is 1350 and the number of epochs is 100

Number of neurons = 1400

- Training accuracy 100.0%
- Testing accuracy 69.04%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	38	13
Another species	13	20

Table E.27: Confusion matrix when the number of hidden neurons is 1400 and the number of epochs is 100

Number of neurons = 1450

- Training accuracy 100.0%
- Testing accuracy 69.04%

	Target	
Predicted	Bumblebees	Another species
Bumblebees	42	14
Another species	12	16

Table E.28: Confusion matrix when the number of hidden neurons is 1450 and the number of epochs is 100

Appendix F

Publications

- **Khalil, H. M (2018)** Identification of Pollinator Species using Bio-acoustics and Artificial Intelligence.
  Royal Entomological Society. Annual Conference (Edge HILL University).
  Presentation, including questions and answers session.

- **Khalil, H. M (2018)** Robotics and Neural Networks.
  Neural Networks Society (Leeds University).
  Presentation, including questions and answers session.

- **Khalil, H. M (2018)** Automated Classification of Insect Pollinator Species using Bio-acoustics.
  The adaptive Many-Core architecture and systems. Workshop.
  Poster, including questions and answers session.

Bibliography

[1] S. G. Potts, J. C. Biesmeijer, C. Kremen, P. Neumann, O. Schweiger, and W. E. Kunin, “Global pollinator declines: trends, impacts and drivers,” *Trends in ecology & evolution*, vol. 25, no. 6, pp. 345–353, 2010.

[2] C. on Biological Diversity, “Pollinators - introduction,” 6.8.2012”.

[3] J. Ollerton, R. Winfree, and S. Tarrant, “How many flowering plants are pollinated by animals?,” *Oikos*, vol. 120, no. 3, pp. 321–326, 2011.

[4] N. Gallai, J.-M. Salles, J. Settele, and B. E. Vaissière, “Economic valuation of the vulnerability of world agriculture confronted with pollinator decline,” *Ecological economics*, vol. 68, no. 3, pp. 810–821, 2009.

[5] A. J. Vanbergen and t. I. P. Initiative, “Threats to an ecosystem service: pressures on pollinators,” *Frontiers in Ecology and the Environment*, vol. 11, no. 5, pp. 251–259, 2013.

[6] T. D. Breeze, A. P. Bailey, K. G. Balcombe, and S. G. Potts, “Pollination services in the uk: How important are honeybees?,” *Agriculture, Ecosystems & Environment*, vol. 142, no. 3-4, pp. 137–143, 2011.

[7] L. A. Garibaldi, I. Steffan-Dewenter, R. Winfree, M. A. Aizen, R. Bommarco, S. A. Cunningham, C. Kremen, L. G. Carvalheiro, L. D. Harder, O. Afik, *et al.*, “Wild pollinators enhance fruit set of crops regardless of honey bee abundance,” *science*, vol. 339, no. 6127, pp. 1608–1611, 2013.

[8] O. Rollin, V. Bretagnolle, L. Fortel, L. Guilbaud, and M. Henry, "Habitat, spatial and temporal drivers of diversity patterns in a wild bee assemblage," *Biodiversity and Conservation*, vol. 24, no. 5, pp. 1195–1214, 2015.

[9] D. Goulson, *Bumblebees: behaviour, ecology, and conservation.* Oxford University Press on Demand, 2010.

[10] J. Ollerton, V. Price, W. S. Armbruster, J. Memmott, S. Watts, N. M. Waser, Ø. Totland, D. Goulson, R. Alarcón, J. C. Stout, *et al.*, "Overplaying the role of honey bees as pollinators: a comment on aebi and neumann (2011)," *Trends in Ecology and Evolution*, vol. 27, no. 3, p. 141, 2012.

[11] J. C. Biesmeijer, S. P. Roberts, M. Reemer, R. Ohlemüller, M. Edwards, T. Peeters, A. Schaffers, S. G. Potts, R. Kleukers, C. Thomas, *et al.*, "Parallel declines in pollinators and insect-pollinated plants in britain and the netherlands," *Science*, vol. 313, no. 5785, pp. 351–354, 2006.

[12] L. G. Carvalheiro, W. E. Kunin, P. Keil, J. Aguirre-Gutiérrez, W. N. Ellis, R. Fox, Q. Groom, S. Hennekens, W. Van Landuyt, D. Maes, *et al.*, "Species richness declines and biotic homogenisation have slowed down for nw-european pollinators and plants," *Ecology letters*, vol. 16, no. 7, pp. 870–878, 2013.

[13] D. Goulson, E. Nicholls, C. Botías, and E. L. Rotheray, "Bee declines driven by combined stress from parasites, pesticides, and lack of flowers," *Science*, vol. 347, no. 6229, p. 1255957, 2015.

[14] G. Powney, C. Carvell, M. Edwards, R. Morris, H. Roy, B. Woodcock, and N. Isaac, "Widespread losses of pollinating insects in britain. nat. commun. 10, 1018," 2019.

[15] CBD, "Convention on biological diversity: Pollinators - introduction.," 2014. Last accessed 16 September 2016.

[16] FAO, “Protocol to detect and monitor pollinator communities: Guidance for practitioners,” 2016. Last accessed 16 September 2017.

[17] DEFRA, “The national pollinator strategy: for bees and other pollinators in england.,” 2014. Last accessed 16 September 2016.

[18] DEFRA, “The national pollinator strategy 2014 to 2024: implementation plan (defra, ed.,” 2015. Last accessed 16 December 2016.

[19] M. Baude, W. E. Kunin, N. D. Boatman, S. Conyers, N. Davies, M. A. Gillespie, R. D. Morton, S. M. Smart, and J. Memmott, “Historical nectar assessment reveals the fall and rise of floral resources in britain,” *Nature*, vol. 530, no. 7588, pp. 85–88, 2016.

[20] J. P. González-Varo, J. C. Biesmeijer, R. Bommarco, S. G. Potts, O. Schweiger, H. G. Smith, I. Steffan-Dewenter, H. Szentgyörgyi, M. Woyciechowski, and M. Vilà, “Combined effects of global change pressures on animal-mediated pollination,” *Trends in ecology & evolution*, vol. 28, no. 9, pp. 524–530, 2013.

[21] N. R. Council *et al.*, *Status of pollinators in North America*. National Academies Press, 2007.

[22] G. Emily, “Bees and the crops they pollinate are at risk from climate change, ipcc report to warn,” 2014. Last accessed 16 December 2017.

[23] A.-M. Klein, B. E. Vaissiere, J. H. Cane, I. Steffan-Dewenter, S. A. Cunningham, C. Kremen, and T. Tscharntke, “Importance of pollinators in changing landscapes for world crops,” *Proceedings of the royal society B: biological sciences*, vol. 274, no. 1608, pp. 303–313, 2007.

[24] S. G. Potts, H. T. Ngo, J. C. Biesmeijer, T. D. Breeze, L. V. Dicks, L. A. Garibaldi, R. Hill, J. Settele, and A. Vanbergen, “The assessment report of the intergovernmental science-policy platform on biodiversity and ecosystem services on pollinators, pollination and food production,” 2016.

[25] M. E. Saunders, "Insect pollinators collect pollen from wind-pollinated plants: implications for pollination ecology and sustainable agriculture," *Insect conservation and diversity*, vol. 11, no. 1, pp. 13–31, 2018.

[26] N. L. García, "The current situation on the international honey market," *Bee World*, vol. 95, no. 3, pp. 89–94, 2018.

[27] H. Lee, D. A. Sumner, and A. Champetier, "Pollination markets and the coupled futures of almonds and honey bees: simulating impacts of shifts in demands and costs," *American Journal of Agricultural Economics*, vol. 101, no. 1, pp. 230–249, 2019.

[28] W. V. Reid, H. A. Mooney, A. Cropper, D. Capistrano, S. R. Carpenter, K. Chopra, P. Dasgupta, T. Dietz, A. K. Duraiappah, R. Hassan, *et al.*, *Ecosystems and human well-being-Synthesis: A report of the Millennium Ecosystem Assessment.* Island Press, 2005.

[29] S. G. Potts, S. P. Roberts, R. Dean, G. Marris, M. A. Brown, R. Jones, P. Neumann, and J. Settele, "Declines of managed honey bees and beekeepers in europe," *Journal of apicultural research*, vol. 49, no. 1, pp. 15–22, 2010.

[30] BeeBase, "Beebase: Hive count," 2014. cited 2018 28 January.

[31] B. B. Association, "British beekeepers association: Honey survey," 2015. cited 2016 28 January.

[32] J. Ollerton, H. Erenler, M. Edwards, and R. Crockett, "Extinctions of aculeate pollinators in britain and the role of large-scale agricultural changes," *Science*, vol. 346, no. 6215, pp. 1360–1362, 2014.

[33] E. Chesmore and C. Nellenbach, "Acoustic methods for the automated detection and identification of insects," in *III International Symposium on Sensors in Horticulture 562*, pp. 223–231, 1997.

[34] J. Biesmeijer, S. Roberts, M. Reemer, R. Ohlemüller, M. Edwards, T. Peeters, A. Schaffers, S. Potts, R. Kleukers, C. Thomas, J. Settele, and W. Kunin, "Parallel declines in pollinators and insect-pollinated plants in britain and the netherlands," *Science (New York, N.Y.)*, vol. 313, pp. 351–4, 08 2006.

[35] A. J. Vanbergen, M. S. Heard, T. Breeze, S. G. Potts, and N. Hanley, "Status and value of pollinators and pollination services," 2014.

[36] M. J. Pocock, H. E. Roy, C. D. Preston, and D. B. Roy, "The biological records centre: a pioneer of citizen science," *Biological Journal of the Linnean Society*, vol. 115, no. 3, pp. 475–493, 2015.

[37] A. Miller-Rushing, R. Primack, and R. Bonney, "The history of public participation in ecological research," *Frontiers in Ecology and the Environment*, vol. 10, no. 6, pp. 285–290, 2012.

[38] T. T, "Kljuc za dolocanje pogostih vrst cmrljev [a key for determination of common bumblebee species]," 2014. cited 2016 28 January.

[39] J. Ghazoul, "Pollen and seed dispersal among dispersed plants," *Biological Reviews*, vol. 80, no. 3, pp. 413–443, 2005.

[40] J. C. Grixti, L. T. Wong, S. A. Cameron, and C. Favret, "Decline of bumble bees (bombus) in the north american midwest," *Biological conservation*, vol. 142, no. 1, pp. 75–84, 2009.

[41] S. A. Cameron, J. D. Lozier, J. P. Strange, J. B. Koch, N. Cordes, L. F. Solter, and T. L. Griswold, "Patterns of widespread decline in north american bumble bees," *Proceedings of the National Academy of Sciences*, vol. 108, no. 2, pp. 662–667, 2011.

[42] C. Carvell, N. Isaac, M. Jitlal, J. Peyton, G. Powney, D. Roy, A. Vanbergen, R. O'Connor, C. Jones, B. Kunin, *et al.*, "Design and testing of a national pollinator and pollination monitoring framework," 2017.

[43] P. Cardoso, T. L. Erwin, P. A. Borges, and T. R. New, "The seven impediments in invertebrate conservation and how to overcome them," *Biological Conservation*, vol. 144, no. 11, pp. 2647–2655, 2011.

[44] D. Chesmore, "Automated bioacoustic identification of insects for phytosanitary and ecological applications," *Computational bioacoustics for assessing biodiversity*, p. 59, 2008.

[45] A. Dafni, P. G. Kevan, B. C. Husband, *et al.*, "Practical pollination biology.," *Practical pollination biology.*, 2005.

[46] S. Potts, P. Kevan, and J. Boone, "Conservation in pollination: Collecting, surveying and monitoring," *Pollination Ecology: A Practical Approach*, pp. 401–434, 01 2005.

[47] C. Westphal, R. Bommarco, G. Carré, E. Lamborn, N. Morison, T. Petanidou, S. G. Potts, S. P. Roberts, H. Szentgyörgyi, T. Tscheulin, *et al.*, "Measuring bee diversity in different european habitats and biogeographical regions," *Ecological monographs*, vol. 78, no. 4, pp. 653–671, 2008.

[48] A. Nielsen, I. Steffan-Dewenter, C. Westphal, O. Messinger, S. G. Potts, S. P. Roberts, J. Settele, H. Szentgyörgyi, B. E. Vaissière, M. Vaitis, *et al.*, "Assessing bee species richness in two mediterranean communities: importance of habitat type and sampling techniques," *Ecological Research*, vol. 26, no. 5, pp. 969–983, 2011.

[49] T. J. Popic, Y. C. Davila, and G. M. Wardle, "Evaluation of common methods for sampling invertebrate pollinator assemblages: net sampling out-perform pan traps," *PloS one*, vol. 8, no. 6, 2013.

[50] J. H. Cane, R. L. Minckley, and L. J. Kervin, "Sampling bees (hymenoptera: Apiformes) for pollinator community studies: pitfalls of pantrapping," *Journal of the Kansas Entomological Society*, pp. 225–231, 2000.

[51] W. D. KIRK, “Ecologically seiective coioured traps,” *Ecological Entomology*, vol. 9, no. 1, pp. 35–41, 1984.

[52] J. M. Leong and R. W. Thorp, “Colour-coded sampling: the pan trap colour preferences of oligolectic and nonoligolectic bees associated with a vernal pool plant,” *Ecological Entomology*, vol. 24, no. 3, pp. 329–335, 1999.

[53] M. E. Saunders and G. W. Luck, “Pan trap catches of pollinator insects vary with habitat,” *Australian Journal of Entomology*, vol. 52, no. 2, pp. 106–113, 2013.

[54] T. R. ToLER, E. W. EvANs, and V. J. Tepedino, “Pan-trapping for bees (hymenoptera: Apiformes) in utah’s west desert: the importance of color diversity,” *Pan Pacific Entomologist*, vol. 81, no. 3-4, pp. 103–113, 2005.

[55] T. H. Roulston, S. A. Smith, and A. L. Brewster, “A comparison of pan trap and intensive net sampling techniques for documenting a bee (hymenoptera: Apiformes) fauna,” *Journal of the Kansas Entomological Society*, vol. 80, no. 2, pp. 179–181, 2007.

[56] J. S. Wilson, T. Griswold, and O. J. Messinger, “Sampling bee communities (hymenoptera: Apiformes) in a desert landscape: are pan traps sufficient?,” *Journal of the Kansas Entomological Society*, vol. 81, no. 3, pp. 288–300, 2008.

[57] K. A. Baum and K. E. Wallen, “Potential bias in pan trapping as a function of floral abundance,” *Journal of the Kansas entomological society*, vol. 84, no. 2, pp. 155–159, 2011.

[58] C. S. Bartholomew and D. Prowell, “Pan compared to malaise trapping for bees (hymenoptera: Apoidea) in a longleaf pine savanna,” *Journal of the Kansas Entomological Society*, vol. 78, no. 4, pp. 390–392, 2005.

[59] J. W. Campbell and J. Hanula, “Efficiency of malaise traps and colored pan traps for collecting flower visiting insects from three forested ecosystems,” *Journal of Insect Conservation*, vol. 11, no. 4, pp. 399–408, 2007.

[60] W. P. Stephen and S. Rao, “Unscented color traps for non-apis bees (hymenoptera: Apiformes),” *Journal of the Kansas Entomological Society*, pp. 373–380, 2005.

[61] C. Kimoto, S. J. DeBano, R. W. Thorp, S. Rao, and W. P. Stephen, “Investigating temporal patterns of a native bee community in a remnant north american bunchgrass prairie using blue vane traps,” *Journal of Insect Science*, vol. 12, no. 1, p. 108, 2012.

[62] M. A. Hall and E. L. Reboud, “High sampling effectiveness for non-bee flower visitors using vane traps in both open and wooded habitats,” *Austral Entomology*, vol. 58, no. 4, pp. 836–847, 2019.

[63] N. K. Joshi, T. Leslie, E. G. Rajotte, M. A. Kammerer, M. Otieno, and D. J. Biddinger, “Comparative trapping efficiency to characterize bee abundance, diversity, and community composition in apple orchards,” *Annals of the Entomological Society of America*, vol. 108, no. 5, pp. 785–799, 2015.

[64] K. W. McCravy, “A review of sampling and monitoring methods for beneficial arthropods in agroecosystems,” *Insects*, vol. 9, no. 4, p. 170, 2018.

[65] R. Grundel, K. J. Frohnapple, R. P. Jean, and N. B. Pavlovic, “Effectiveness of bowl trapping and netting for inventory of a bee community,” *Environmental Entomology*, vol. 40, no. 2, pp. 374–380, 2011.

[66] A. Gradišek, G. Slapničar, J. Šorn, M. Luštrek, M. Gams, and J. Grad, “Predicting species identity of bumblebees through analysis of flight buzzing sounds,” *Bioacoustics*, vol. 26, no. 1, pp. 63–76, 2017.

[67] C.-J. Huang, Y.-J. Yang, D.-X. Yang, and Y.-J. Chen, "Frog classification using machine learning techniques," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3737–3743, 2009.

[68] J. Cheng, Y. Sun, and L. Ji, "A call-independent and automatic acoustic system for the individual recognition of animals: A novel model using four passerines," *Pattern Recognition*, vol. 43, no. 11, pp. 3846–3852, 2010.

[69] C.-H. Lee, C.-C. Han, and C.-C. Chuang, "Automatic classification of bird species from their sounds using two-dimensional cepstral coefficients," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 8, pp. 1541–1550, 2008.

[70] M. A. Acevedo, C. J. Corrada-Bravo, H. Corrada-Bravo, L. J. Villanueva-Rivera, and T. M. Aide, "Automated classification of bird and amphibian calls using machine learning: A comparison of methods," *Ecological Informatics*, vol. 4, no. 4, pp. 206–214, 2009.

[71] M. J. King, S. L. Buchmann, and H. Spangler, "Activity of asynchronous flight muscle from two bee families during sonication (buzzing).," *Journal of Experimental Biology*, vol. 199, no. 10, pp. 2317–2321, 1996.

[72] M. T. Lopes, L. L. Gioppo, T. T. Higushi, C. A. Kaestner, C. N. Silla Jr, and A. L. Koerich, "Automatic bird species identification for large number of species," in *2011 IEEE International Symposium on Multimedia*, pp. 117–122, IEEE, 2011.

[73] T. Ganchev, I. Potamitis, and N. Fakotakis, "Acoustic monitoring of singing insects," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 4, pp. IV–721, IEEE, 2007.

[74] W. J. Sutherland, D. B. Roy, and T. Amano, "An agenda for the future of biological recording for ecological monitoring and citizen science," *Biological Journal of the Linnean Society*, vol. 115, no. 3, pp. 779–784, 2015.

[75] M. J. King, "Buzz foraging mechanism of bumble bees," *Journal of Apicultural Research*, vol. 32, no. 1, pp. 41–49, 1993.

[76] W. Kirchner and J. Röschard, "Hissing in bumblebees: an interspecific defence signal," *Insectes sociaux*, vol. 46, no. 3, pp. 239–243, 1999.

[77] P. A. De Luca and M. Vallejo-Marin, "What's the 'buzz'about? the ecology and evolutionary significance of buzz-pollination," *Current opinion in plant biology*, vol. 16, no. 4, pp. 429–435, 2013.

[78] P. A. De Luca, D. A. Cox, and M. Vallejo-Marín, "Comparison of pollination and defensive buzzes in bumblebees indicates species-specific and context-dependent vibrations," *Naturwissenschaften*, vol. 101, no. 4, pp. 331–338, 2014.

[79] G. W. King RA, "Time-encoded speech," *Electronics Letters*, 1978.

[80] C. M. Jernigan, "Bee anatomy," 2017. cited 2018 28 January.

[81] J. Miley, "The buzz of bees explained," 2019. cited 2019 28 March.

[82] A. Rashed, M. Khan, J. Dawson, J. Yack, and T. Sherratt, "Do hoverflies (Diptera: Syrphidae) sound like the Hymenoptera they morphologically resemble?," *Behavioral Ecology*, vol. 20, pp. 396–402, 11 2008.

[83] H. Stratosphere, "9 different types of wasps (plus how they're different than bees)," Not known. cited 2019 28 March.

[84] E. Britannica, "Hover fly)," January 11, 2018. cited 2019 28 November.

[85] R. King, "TESPAR/FANN – An Effective New Capability for Voice Verification in the Defence Environment. Proceeding of the," *Proceeding of the Royal*, 1995.

[86] W. Lucking, G. Darnell, and E. Chesmore, "Acoustical condition monitoring of a mechanical gearbox using artificial neural networks," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 5, pp. 3307–3311, IEEE, 1994.

[87] S. MD and E. Chesmore, "Automatic classification of heart sounds and murmurs using time domain signal coding (tdsc), artificial neural networks and expert systems.," p. 37–46., In: Conference on Recent Advances in Soft Computing'98 1998; DeMontfort University, 1998.

[88] J. C. R. Licklider and I. Pollack, "Effects of differentiation, integration, and infinite peak clipping upon the intelligibility of speech," *The Journal of the Acoustical Society of America*, vol. 20, no. 1, pp. 42–51, 1948.

[89] F. Bond and C. Cahn, "On the sampling the zeros of bandwidth limited signals," *IRE transactions on information theory*, vol. 4, no. 3, pp. 110–113, 1958.

[90] . C. C. . H. J. Bond, F.E., "A relation between zero crossings and fourier coefficients for bandwidth limited functions," *Transactions on Information Theory*, vol. 6, no. 1, pp. 51–52, 1960.

[91] M. Swarbrick, *Acoustic Diagnosis of Heart Defects using Time Domain Signal Processing and Artificial Neural Networks.PhD These.* PhD thesis, niversity of Hull, 2001.

[92] S. M. . F. O. Chesmore, E.D., "Automated analysis of insect sounds using tespar and expert systems – a new method for species identification," *nformation Technology, Plant Pathology and Biodiversity*, pp. 273–287, 1998.

[93] J. Farr, *Automated Bioacoustic Identification of Statutory Quarantined Insect Pests. PhD Thesis.* PhD thesis, University of Hull, 2007.

[94] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[95] W. Chu and B. Champagne, "A noise-robust fft-based auditory spectrum with application in audio classification," *IEEE Transactions on audio, speech, and language processing*, vol. 16, no. 1, pp. 137–150, 2007.

[96] S. G. Brush, "Irreversibility and indeterminism: Fourier to heisenberg," *Journal of the History of Ideas*, vol. 37, no. 4, pp. 603–630, 1976.

[97] R. Yan, *Base wavelet selection criteria for non-stationary vibration analysis in bearing health diagnosis.* University of Massachusetts Amherst, 2007.

[98] C.-H. Chuan, S. Vasana, and A. Asaithambi, "Using wavelets and gaussian mixture models for audio classification," in *2012 IEEE International Symposium on Multimedia*, pp. 421–426, IEEE, 2012.

[99] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[100] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[101] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken language processing: A guide to theory, algorithm, and system development.* Prentice hall PTR, 2001.

[102] T. Singh, "Mfcc's made easy.," 15.7.2019".

[103] H. Fayek, "Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between," 21. 4. 2016.

[104] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system with an efficient search strategy," *Journal of New Music Research*, vol. 32, no. 2, pp. 211–221, 2003.

[105] A. L. Mcllraith and H. Card, "Birdsong recognition with dsp and neural network," *IEEE Wescanex Proceeding*, 1995.

[106] G. Grigg, A. Taylor, H. Mc Callum, and G. Watson, “Monitoring frog communities: an application of machine learning,” in *Proceedings of eighth innovative applications of artificial intelligence conference, Portland Oregon*, pp. 1564–1569, 1996.

[107] N. VAUGHAN, G. JONES, and S. HARRIS, “Identification of british bat species by multivariate analysis of echolocation call parameters,” *Bioacoustics*, vol. 7, no. 3, pp. 189–207, 1997.

[108] E. D. Chesmore, “Application of time domain signal coding and artificial neural networks to passive acoustical identification of animals,” *Applied Acoustics*, vol. 62, no. 12, pp. 1359–1374, 2001.

[109] V. Vapnik and A. Lerner, “”pattern recognition using generalized portrait method,” *Automation and Remote Control*, vol. 24, pp. 774—-780, 1963.

[110] E. Kim, “Everything you wanted to know about the kernel trick,” 12.20.2017”.

[111] M.-W. Mak and S.-Y. Kung, “Low-power svm classifiers for sound event classification on mobile devices,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1985–1988, IEEE, 2012.

[112] T. K. Ho, “Random decision forests,” in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR ’95, (USA), p. 278, IEEE Computer Society, 1995.

[113] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[114] Y. Amit and D. Geman, “Shape quantization and recognition with randomized trees,” *Neural computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[115] V. Zhou, "A simple explanation of gini impurity what gini impurity is (with examples) and how it's used to train decision trees.," 29.03.2019".

[116] J. Rocca, "Ensemble methods: bagging, boosting and stacking," 2Apr 23, 2019.

[117] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[118] R. Leandro, C. Silva, and M. Santos, "Feeding neural network models with gps observations: a challenging task," in *Dynamic Planet*, pp. 186–193, Springer, 2007.

[119] H. Research, "The number of hidden layers," 7.05.2020".

[120] vikashraj luhaniwal, "Analyzing different types of activation functions in neural networks — which one to prefer?," 7.05.2019".

[121] Guru99, "Deep learning tutorial for beginners: Neural network classification," 7.05.2018".

[122] O. E. Chesmore ED, "Automated identification of field-recorded songs of four british grasshoppers using bioacoustic signal recognition," *Bull Entomol Res*, 2004.

[123] T. Kohonen, *Self-Organization and Associative Memory.* Springer-Verlag Berlin Heidelberg, 1987.

[124] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16-18, pp. 3056–3062, 2007.

[125] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16-18, pp. 3460–3468, 2008.

[126] G.-B. Huang, L. Chen, C. K. Siew, *et al.*, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[127] S. Lin, X. Liu, J. Fang, and Z. Xu, “Is extreme learning machine feasible? a theoretical assessment (part ii),” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 21–34, 2014.

[128] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[129] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[130] C. Cortes and V. Vapnik, “Support vector machine,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[131] M. Fernández-Delgado, E. Cernadas, S. Barro, J. Ribeiro, and J. Neves, “Direct kernel perceptron (dkp): Ultra-fast kernel elm-based classification with non-iterative closed-form weight calculation,” *Neural Networks*, vol. 50, pp. 60–71, 2014.

[132] G. Huang, S. Song, J. N. Gupta, and C. Wu, “Semi-supervised and unsupervised extreme learning machines,” *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.

[133] G.-B. Huang, “An insight into extreme learning machines: random neurons, random features and kernels,” *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[134] Y. Xu, Y. Dai, Z. Y. Dong, R. Zhang, and K. Meng, “Extreme learning machine-based predictor for real-time frequency stability assessment of electric power systems,” *Neural Computing and Applications*, vol. 22, no. 3-4, pp. 501–508, 2013.

[135] Y. Tan, R. Dong, H. Chen, and H. He, “Neural network based identification of hysteresis in human meridian systems,” *International journal of*

applied mathematics and computer science, vol. 22, no. 3, pp. 685–694, 2012.

[136] H.-X. Tian and Z.-Z. Mao, "An ensemble elm based on modified adaboost. rt algorithm for predicting the temperature of molten steel in ladle furnace," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 73–80, 2009.

[137] F. Chen and T. Ou, "Sales forecasting system based on gray extreme learning machine with taguchi method in retail industry," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1336–1345, 2011.

[138] W. Wong and Z. Guo, "A hybrid intelligent model for medium-term sales forecasting in fashion retail supply chains using extreme learning machine and harmony search algorithm," *International Journal of Production Economics*, vol. 128, no. 2, pp. 614–624, 2010.

[139] D. Ruhmelhart, G. Hinton, and R. Wiliams, "Learning representations by back-propagation errors," *Nature*, vol. 323, no. 533-536, p. 10, 1986.

[140] R. Rojas, "The backpropagation algorithm," in *Neural networks*, pp. 149–182, Springer, 1996.

[141] A. N. N. T. association, "Bumblebee id guide," 7.05.2020".

[142] T. points, "Tensorflow - cnn and rnn difference) and what's in-between," 21. 4. 2016.