Bird Sound Classification

Individual Project

AUTHOR:
SINDURAN SIVARAJAN

SUPERVISORS:
BJÖRN SCHULLER
EDUARDO COUTINHO

Department of Computing
IMPERIAL COLLEGE LONDON

Submitted in part fulfilment of the requirements for the degree of BACHELOR OF ENGINEERING in Computing

JUNE 2016

Abstract

The analysis of birds present in an area is of great importance to the study of ecology and can provide crucial information relating to the environmental changes. Current methods primarily requires an Ornithologist to perform manual classification, either directly or indirectly using collected visual and audio data. Recent advancements in Machine Learning techniques has attracted a wide variety of uses including using audio data for classification. The aim of this research is to apply state-of-the-art Machine Learning techniques suited to classification to classify birds into its corresponding species, family and order with the use of various input features that have proven to be successful in similar tasks in the past. The dataset used is the RefSys Dataset obtained from Animal Sound Archive consisting of 6476 audio recordings in total. Having processed the database the final database consisted 36 species/15 family/8 order. The models that were tested ranged from simple popular models such as Support Vector Machines to complex deep neural networks. The best models tested here achieve an Unweighted Average Recall of 90.7%, 88.2% and 79.6% for species, family and order respectively using Artificial Neural Network. The results obtained in this project has shown a significant improvement over state-of-the-art approaches on the same dataset.

Dedication and acknowledgements

I would like to thank Björn Schuller for suggesting this project. I would also like express my appreciation of Eduardo Coutinho for his whole hearted cooperation and constant encouragement throughout the project.

Table of Contents

Chapter 1

Introduction

Birds are found everywhere, and the species present in a specific area can give more information on the biodiversity [10] and of the species richness and endemism [18] of wildlife in that particular area. In addition the difference in distributions of the different species of birds in a particular geographical area over a period of time, can also suggest changes in climate [6] and predation.

Animals in the wild are primarily classified using visual and/or acoustical data. Current methods of classification of birds include visual analysis using photography and/or telescope due to their optical advances allowing little loss in quality. At present it requires human intervention, in particular that of an ornithologist to identify any species that exist in the biodiversity for a highly accurate classification or in few cases machine learning algorithms are used to automate the process but suffers from low classification rate or noise in images such as bad weather. Birds sound is another aspect that is commonly used in the classification of birds, but again requires human intervention. Animal classification with sound has proved to be effective in the past [4] many new species have been discovered by differences in calls of the birds rather than visual difference since the birds have little morphological differences [42].

Machine Learning has had recent advancements and has also proved to be highly useful in the automated classification and analysis of sounds. However the application of bird classification from sounds using machine learning techniques has only been analysed with a few number of species or manually processed/noise free audio samples and is thus unpractical for application in the real world for ecologists [41], though some useful specific analysis for large scale application exist [32]. While the classification rate analysis for a small number of species may look promising when extending it to the large number of species that are actually present the results may not be as satisfactory due to small differences in variances of species. Current machine learning algorithms that have proved to be successful in classification are Artificial Neural Networks and Convolutional Neural Networks which have not been explored as much in the area of bird classification as Support Vector Machine has [23].

The aim of this project is to investigate multiple machine learning models in the classification of birds. Initially engineered feature sets will be analysed with models that are suited to work with a relatively small input size. The models include Support Vector Classification (SVC), K-Nearest Neighbour (KNN), Linear Discriminant Analysis (LDA) and Artificial Neural Network (ANN). Subsequent to this automatic feature extraction will be investigated with inputs such as Spectrogram, Wavelets and raw audio data using Convolutional Neural Network (CNN) to extract key features. The advantage of using engineered features is that spectral features have been specifically analysed and extracted therefore the machine learning algorithm only has to classify the given input. However with automatic feature extraction no prior knowledge has to be known as the Machine Learning model will extract key features. The performance of all the models are given in Results (Chapter 6). ANN has led to the best performance on the Test set, giving an accuracy of 91.2% and an Unweighted Average Recall of 90.7% in classifying 36 species of birds. The findings show that there is a potential of Machine Learning in the classification of birds if further research is done. Some further works are discussed in Conclusions and Outlook (Chapter 7).

CHAPTER 2

BACKGROUND

Living organisms are classified in many ways, with the most general being the kingdom whereby organisms are classified as being one of Animals, Plants, Fungi, Bacteria or Protists. Animal Kingdom is then divided further into smaller groups known as Division (Phylum) and then Subdivision (Subphylum) [39] and whereby major classes include invertebrates and vertebrates. Order is a taxonomic rank in which organisms are further split into organisms with differences, for example humans are Mammals and order of humans is Primates. Order is then further classified using family, in which organisms with similar features are gathered together.

All breeds of dogs for example belong to Canidae family, whereas humans belong to the Hominidae family along with Chimpanzees, Gorilla and Orangutan [14]. Following Family is the Genus which is then followed by the most specific classification of species and is named using the Binomial (two name) system, where the first indicates the genus and the second the species. Modern Humans are Homo sapiens, whereas other organisms in the Genus Homo have all become extinct including Homo Neanderthalensis commonly known as Neanderthals. Modern taxonomy and the hierarchical classification is cemented with the work by Carolus Linnaeus and Charles Darwin [12].

Different animals produce sounds in different ways. The speech process that most humans take for granted is also produced in a complex way. Humans as they breathe air out of their lungs, passes the larynx [24], which causes the air to vibrate, this is then finely tuned to produce the required phrase, by shaping the throat, tongue and mouth in a specific combination. Birds on the other hand use their syrinx to make noises, the figure below (Figure 2.1) shows the position of the syrinx. Different species of birds have different number of syringeal muscles [11], and hence their range of calls vary allowing them to be differentiated.

Recent developments in Computational Paralinguistic has allowed computers to extract engineered features for use in classification [43] from not the context of speech but rather the

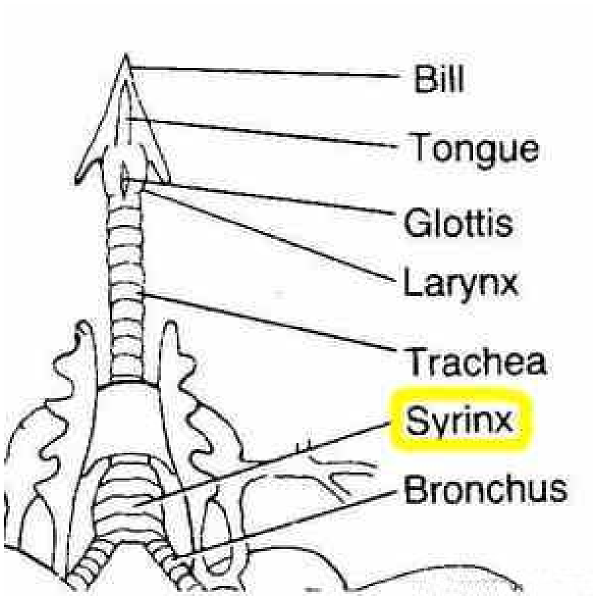


FIGURE 2.1. The location of the Syrinx can be seen clearly[11]

vocal features. This principle can also be observed in different species of animals, particularly birds. Different species have different calls, and can be differentiated if the right features are extracted and analysed.

2.1 Previous Works

A considerable amount of research exists in the area of bird classification. “Wavelets in recognition of bird sounds” [44] for instance used wavelets to classify 8 species, and attained an accuracy of 96%. However they cannot all be used as comparison to this project, as different databases were used and would not give rise to a fair comparison. Related work that has been done using the same database includes Extreme Machine Learning applied to Bird Sound Classification [41]. Different number of species were tested ranging from 10 to 54 and the features used include engineered features which will be one of the feature set investigated in this project. A mean Unweighted Average Recall (UAR) of 89.56% and 85.30% was obtained for 30 and 40 species respectively. These UAR will be used as a comparison to the results obtained in this project. Another research that used the same database is large-scale identification of birds in audio recordings, however the number of species investigated is very large [33].

2.2 Machine Learning Models

Machine Learning is defined by Tom Mitchell in his book [37] as “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". Supervised learning is when the output is known for the corresponding inputs, and is also provided for the machine to learn.

Animal Classification will be a supervised learning task, since the dataset consists of audio samples (input) and the corresponding animal class (output). Machine Learning has been applied quite frequently in the classification of animals, and it mainly involves images of animals and occasionally sound in particular Spectrograms as input. With images and spectrograms each pixel can be analysed for patterns, with sound however relevant features will have to be extracted if they are to be fed directly as input. The choice of input features will play a significant role in the classification rate and the confusion between classes.

Classifying animals from their acoustical properties is a multiclass classification problem, in which there are multiple output classes the input can be classed as, and it is the machine learning algorithms or rather our responsibility to ensure that as many samples are classified correctly.

Below are the most popular machine learning models that are commonly used for supervised classification in machine learning, due to their ability to learn complex patterns rather quickly or to a high accuracy.

2.2.1 Support Vector Machines

Support Vector Machine (SVM) is a machine learning model, that takes a multidimensional vector and the class they belong, and creates a boundary between the different classes and their input, so that future data can be easily classified by inspecting the boundary in which it falls in [47]. It is important for the dimension of the input to be much smaller than the number of samples to avoid poor performance. Thus it is only suitable for smaller engineered feature sets.

In mathematical terms, one or more hyper-planes are constructed, which acts as a boundary allowing classification to be performed. The confusion between classes can be reduced by ensuring the distance between points on sides of the hyper-plane have a large enough margin, thus ensuring that generalization error is reduced by classifying unseen data correctly. The figure below (Figure 2.2) shows how a large margin ensures that there is great separation and thus less confusion. Unlike other classifiers which consider all the data points, SVM only considers points that are close to the decision boundary. However depending on the other parameters, a large margin may not always lead to the optimal hyper-plane as if there are errors in dataset, and a particular data point lies very close to data points of other classes.

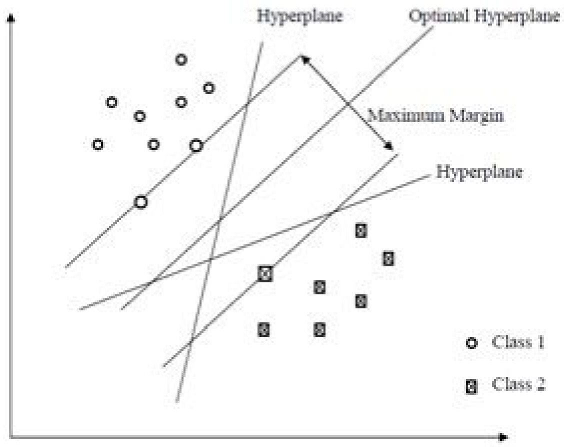


FIGURE 2.2. An optimal hyper-plane separating data points with maximum margin [2]

Different kernel function exists which ultimately determine the shape of the hyperplane. The most common kernels used are linear kernel and radial basis function (see Figure 2.3). Linear kernel split the plane using linear combinations, whereas radial basis function (RBF) also known as Gaussian kernels are commonly used to approximate functions and are functions that are based on the absolute distance from the 'centre' point ($r = ||x - x_i||$) [38]. The RBF kernel between two data points, x and x' is defined by:

$$K(x,x') = e^{-\gamma||x-x'||^2}$$

where $||x - x'||^2$ is the Euclidean distance, γ is a parameter specified and $K(x,x')$ is given as a feature vector.

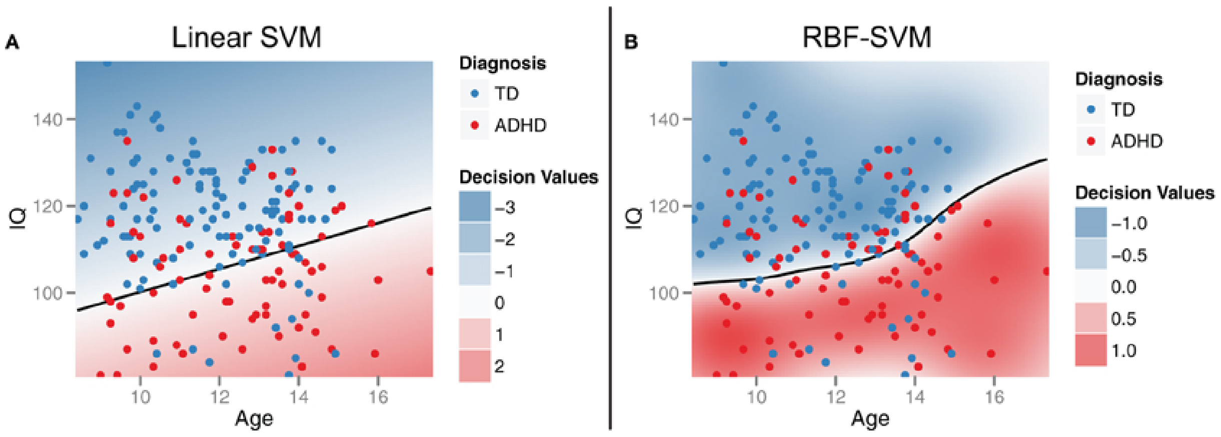


FIGURE 2.3. The shape of decision boundary is determined by the kernel function [16]

2.2.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a statistical technique that is commonly used in classification tasks. In its simplest form, it identifies the key features that best determine the class and then uses predetermined rule from the training phase to classify the newly encountered instance. The class is considered to be the dependent variable and the features that give rise to the discrimination between the classes are considered to be the independent variable. Thus the dimension of the separator that separates the classes is dependent on the number of independent variables. Figure 2.4 shows how LDA separates the classes for two attributes.

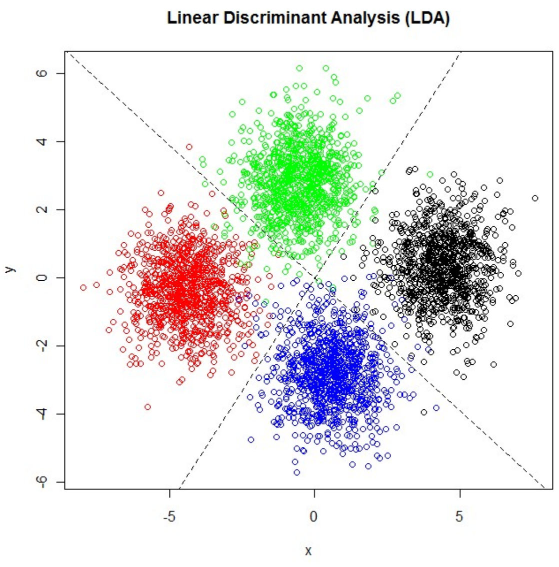


FIGURE 2.4. The points are classified into groups and then linearly separated [50]

The general idea for classification is to classify the instance into the class with the highest conditional probability. The challenge however lies with computing the metrics needed to compute the conditional probability. Bayes theorem states that:

$$P(C=c|X=x) = P(C=c) * \frac{P(X=x|C=c)}{P(X=x)},$$

where C is the class and X is the instance

Since the current observed instance is the only instance considered P(X=x) = 1, and thus can be removed. Assuming the classes all are equally probable P(C=c) (the prior probability) can be written as:

$$P(C=c) = \frac{1}{number\ Of\ Classes}$$

$P(X=x|C=c)$ is the probability density function (pdf) which gives the probability of instance occurring, given that only class c is considered and it is modelled by the multivariate normal distribution:

$$P(X = x|C = c) = \frac{e^{-\frac{1}{2}(x-\mu_k)^T * \Sigma^{-1} * (x-\mu_k)}}{\sqrt{2\pi^k * |\Sigma_k|}}$$

Where μ_k is the mean if class k and Σ_k is the covariance matrix of class k which are both derived from the training examples. However since the $P(C = c|X = x)$ is not normalized, it is normalized by dividing the obtained conditional probability, by the sum of the conditional probabilities for all classes.[34]

2.2.3 K Nearest Neighbour

K Nearest Neighbour (KNN) is an instance based learning algorithm which instead of generalising the training data and extract common patterns between like classes, it uses a memory based approach in which new instances are compared with the training examples that have been provided [40]. KNN works on the principle of majority voting [40], the key parameters of KNN are the number of nearest neighbours considered (K), the algorithm used for computing the nearest neighbours and how different nearest neighbours affect the output.

The training instances are simply stored upon training, though in most cases it is stored in an ordered manner, so that upon retrieval, the nearest neighbours can easily be computed thus reducing classification time.
Upon classification the K neighbours are retrieved and they are weighted depending on the weighting algorithm used, and the class of the new instance is calculated by the formula below.

$$v(x_d) \leftarrow C_i \leftrightarrow (\forall j \neq i) \sum_r w_r \cdot E(C_i, V(x_r)) > \sum_r w_r \cdot E(C_j, V(x_r))$$

where $E(C,V)$ gives 1 if and only if C and V are equal else 0.
The formula simply states that the class of instance i will be C_i if and only if the number of neighbours that are of class C_i is strictly greater than the number of neighbours for all other class C_j, with respects to some weight function w. The common weight function is the inverse distance which takes the reciprocal of the Euclidean distance of the instance in question and the current training instance being considered [40]. If however the feature of an instance is discrete, Euclidean distance would not work, and therefore other distance metrics are used e.g. Hamming Distance [29].

Figure 2.5 below coherently explains K Nearest Neighbour, if we consider all the points within the large dotted red circle as our nearest Neighbours, and X in the centre as our new data point, then if we use KNN without any weight then X would be classified as - since the number of - is much greater than +. However if an inverse distance weight is used, then it is possible for X to be classified as +, since they are much closer than the - which are further away.

Being a lazy learning method it has the clear advantage that training time is really short, since it primarily involves storing the training instances. As a result of this, no processing is done

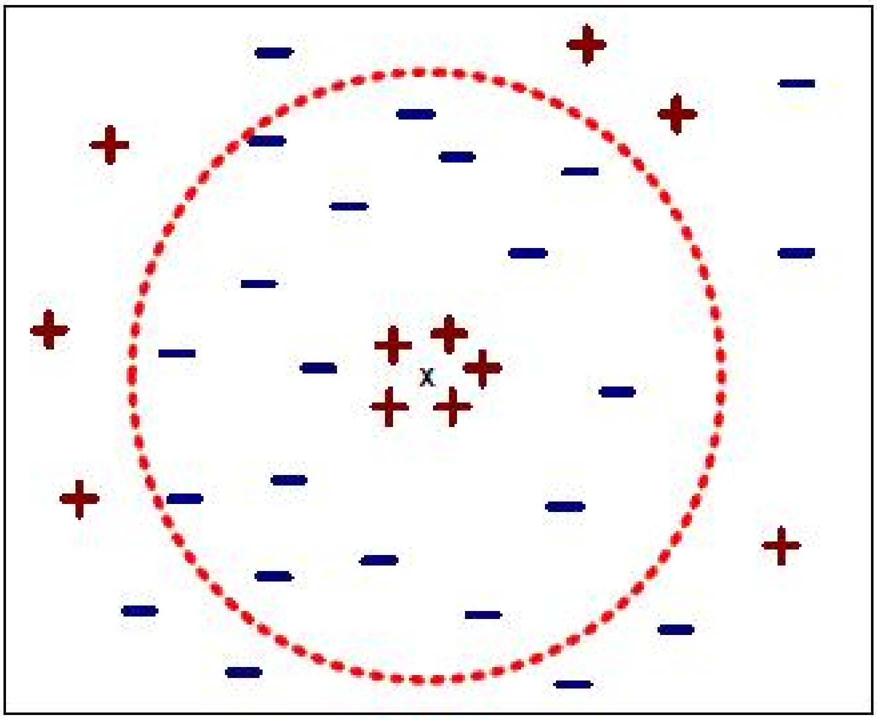


FIGURE 2.5. An illustration of a typical KNN classification [8]

on the training instances, to learn any characteristics. Weighted KNN is also robust to noisy training data, since the number of neighbours that are affected by noise is likely to be small and when using a weighted classification then the noisy data will be effectively omitted. However no concept is learned and thus a hypothesis cannot be interpreted, since it does not explicitly learn a generalised hypothesis. The classification time will be quite large, since almost all the work required is done at this stage. There is also the major problem of the curse of dimensionality [25], in which large number of features can cause difficulties in classification. In distance-weighted KNN for example, there may be attributes that remain constant or have little changes but do not affect the class, whereas there may be other attributes that may change quite drastically, but can play a significant role in the classification. This can be avoided by some form of feature reduction prior to training and classification.

2.2.4 Artificial Neural Network

Based on the biological neural network, Artificial Neural Network (ANN) is commonly applied to learn complex functions that are unknown using training data. Unlike some of the previous models in which a function is approximated using the input and the corresponding output, with ANN the model familiarises the patterns in the input and the output, much like the human brain. There are three types of layers in a neural network, in particular the input layer, which has a node for every input value, hidden layers which are fully connected to the previous layer and the output layer that is also fully connected and that corresponds to the output. Each layer in the ANN consists of one or more neuron which corresponds to the perikaryon in the biological neural network [28]. The net input for node j in the current layer sums linear combination of the output from the N nodes of previous layer,

$$\text{node } j \text{ net input} = \sum_{i=1}^{N} x_i w_{ij} + w_0$$

where w_0 is known as the bias and is used to specify the threshold after which the neuron will be activated [49]. The neuron then applies an activation function to the net input. The activation function transforms the otherwise linear input to a nonlinear output which models more common real world problems. Sigmoid function is a common activation function that squashes the input to fall between 0 and 1

$$f(x) = \frac{1}{1+e^{-x}}$$

Another activation function that has proved to be more common and effective is the ReLu (rectified linear unit); $f(x) = max(0,x)$, computationally can be calculated very quickly.

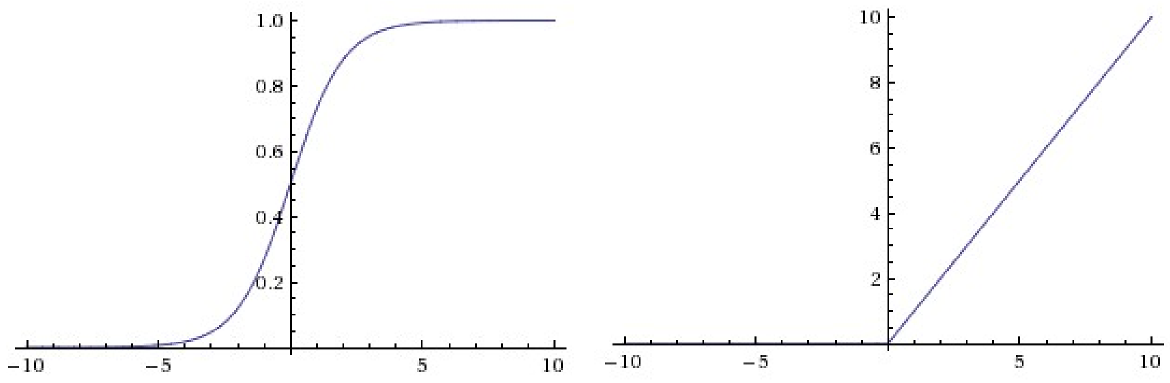


Figure 2.6: Figure on the left shows the sigmoid function while right shows ReLu [5]

The weights of each link between neurons of two adjacent layer is learned by backpropagation from the training data. Backpropagation at a high level works by feeding the training data through the network and inspecting the output, the weights of the network are then adjusted to reduce the error between the expected output and the actual output using gradient descent. The weights are updated by $\triangle w_{ij} = -\eta \frac{\delta E}{\delta w_{ij}}$, where i and j refer to the nodes that the weighted connection exists between, η is the learning rate and affects how quickly the network learns (discussed in more detail in Parameter Optimisation, Section 5.2). The reason for the negative sign is due to the gradient descent, so that $\frac{\delta E}{\delta w_{ij}}$ is minimised. This is known as backpropagation, since the errors are calculated backwards, starting from the output layer. The parameters that are chosen and optimised by the user are known as hyperparameters and this includes the activation functions, learning rate and most importantly the network architecture, this includes the number of layers and the number of nodes/neurons in the layers. Momentum is also a hyperparameter and ensures that the network does not converge to a local optima. Momentum adapts the learning rate by altering the learning rate in the direction that minimises error. Momentum takes a value in the range [0,1] and at the end of each iteration a fraction of the previous learning rate is added. A high momentum ensures that the network converges quickly to the global minima, however if the momentum is too high then the network will become unstable. A large number of libraries exist, that provide implementation to ANN. The library that will be used in this project is PyBrain [1], it has a simple to use API and a verbose documentation allowing even a novice user to implement their own ANN relatively quickly.

2.2.5 Convolutional Neural Network

Convolutional Neural Network (CNN) is an example of deep network. Deep networks unlike ANN have many hidden layers, and as a result different training algorithms are used to ensure that the weights do not drift to zero. Convolutional Neural Network have proved to be effective in Computer Vision and its presence has also recently gained popularity in natural language processing and audio processing [17].

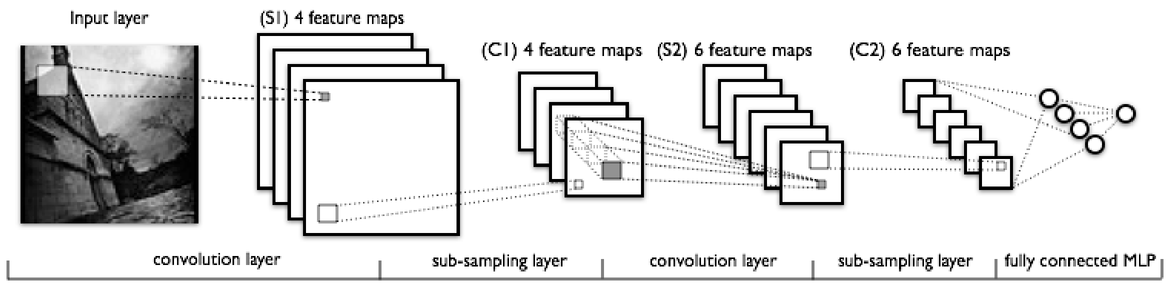


FIGURE 2.7. The figure depicts a typical CNN, the layers are discussed in more detail below [48]

Convolutional Layer applies different filters (kernel function) at every offset of the previous layer matrix, and stores the result, the subsampling layer then downsamples the large matrices obtained from convolutional layer. Subsampling layer reduces the matrix by taking the average (average pooling) or max (max pooling) of non-overlapping sub regions in the matrix. This process of convolving then downsampling is repeated until key features are extracted, and is then fed through multiple dense (fully connected) layers.

A key useful property of CNN is that the key features of the input does not have to be specified but will be extracted by the CNN itself. Thus not much pre-processing has to be done to the input and is the ideal model for automatic feature extraction.

[1]`http://www.pybrain.org/`

Chapter 3

Dataset

Many online resources exist with access to animal noise dataset. Xeno-canto [1] for example, gives access to a plethora of free user collected bird noises along with the species and some metadata about the geographical location. At present it consists of 290,754 recordings of 9,498 species from 2,748 users. Macaulay library [2] is a potential paid resource; claiming to be the largest archive of animal sounds consists of "175,000 audio recordings covering 75 percent of the world's bird species". Animal Sound Archive [3] will be the dataset used throughout this project. It consists of 6,476 recordings of 272 Species/ 64 Families/ 23 Orders summing up to 279 minutes. Of the 6,476 sound files 3,684 consist of bird songs, 2,475 consist of bird calls, 237 are of juveniles and the remaining audio files were undefined. However classes with too few instances (<50) were removed to prevent overfitting, since the number of instances present was very low relative to other classes. The specific classes considered are shown below.

The data recorded for each audio sample are listed in the appendix. Some further data are included but are not directly relevant such as date created. The quality indicates how much noise is present in the data and is element of {a,b,c,u}, where a is the highest quality and the dataset primarily consists of data of quality b. This project will focus on the audio sample only instead of the metadata, since this information may not always be available.

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

[2] `http://macaulaylibrary.org`

[3] `http://www.animalsoundarchive.org/RefSys/Statistics.php`

Order	Test Set 1	Test Set 2	Test Set 3
Accipitriformes, class 1	20	23	15
Podicipediformes, class 2	35	33	30
Anseriformes, class 3	55	58	54
Strigiformes, class 4	73	88	87
Charadriiformes, class 5	86	88	84
Piciformes, class 6	100	98	100
Gruiformes, class 7	154	148	159
Passeriformes, class 8	162	149	155

Figure 3.1: Test Sets for Order

Family	Test Set 1	Test Set 2	Test Set 3
Laridae, class 1	24	24	23
Corvidae, class 2	31	25	36
Motacillidae, class 3	49	43	37
Anatidae, class 4	46	33	38
Acrocephalidae, class 5	127	140	113
Strigidae, class 6	60	54	59
Picidae, class 7	78	93	77
Rallidae, class 8	142	126	130
Sylviidae, class 9	157	144	136
Turdidae, class 10	124	124	143
Paridae, class 11	138	131	149
Fringillidae, class 12	148	141	142
Phylloscopidae, class 13	175	182	179
Emberizidae, class 14	195	191	189
Muscicapidae, class 15	220	255	247

Figure 3.2: Test Sets for Family

Species	Test Set 1	Test Set 2	Test 3
Aegolius funereus, class 1	18	16	16
Dendrocopos medius, class 2	18	17	22
Sylvia melanocephala, class 3	13	20	21
Dendrocopos major, class 4	20	20	13
Dryocopus martius, class 5	17	17	23
Picus canus, class 6	19	18	18
Sylvia borin, class 7	31	20	28
Turdus pilaris, class 8	20	18	16
Troglodytes troglodytes, class 9	21	27	24
Acrocephalus arundinaceus, class 10	38	24	23
Acrocephalus scirpaceus, class 11	56	55	62
Fulica atra, class 12	20	16	26
Turdus viscivorus, class 13	19	16	26
Lophophanes cristatus, class 14	25	17	22
Phylloscopus bonelli, class 15	26	20	18
Porzana porzana, class 16	41	39	31
Emberiza calandra, class 17	24	22	22
Asio otus, class 18	23	29	18
Sylvia communis, class 19	19	30	28
Saxicola rubetra, class 20	30	26	23
Emberiza citrinella, class 21	23	32	31
Sylvia atricapilla, class 22	55	58	42
Rallus aquaticus, class 23	68	53	65
Emberiza schoeniclus, class 24	42	34	43
Turdus philomelos, class 25	47	57	49
Phylloscopus ibericus, class 26	31	50	52
Phylloscopus collybita, class 27	57	59	49
Luscinia luscinia, class 28	69	63	76
Phylloscopus trochilus, class 29	50	55	32
Parus major, class 30	47	45	52
Phoenicurus phoenicurus, class 31	56	50	49
Turdus merula, class 32	49	50	60
Luscinia megarhynchos, class 33	67	72	70
Periparus ater, class 34	61	63	65
Fringilla coelebs, class 35	84	89	88
Emberiza hortulana, class 36	91	97	91

Figure 3.3: Test Sets for Species

CHAPTER 4

FEATURES

4.1 Engineered Features

The engineered feature set consists of 384 features. The 384 features comprises of 16 LLD and 16 corresponding delta regression coefficients to which 12 functionals are applied yielding the 384 features. The LLD consists of MFCC 1-12, RMS energy, probability of voicing and fundamental frequency via autocorrelation function (ACF). The 12 functionals are arithmentic mean, standard deviation, skewness, kurtosis, maximum and minimum value, range, relative position of max and min value, linear regression slope, offset and quadratic error [20], this is summarised in Table 4.1.

Low Level Descriptors	Functionals
ZCR	mean
RMS Energy	standard deviation
F0	kurtosis,skewness
HNR	extremes: value, relative position, range
MFCC 1-12	linear regression: offset, slope, MSE

Table 4.1: Summary of engineered features investigated [20]

openSMILE [1] will be the toolkit that will be used to extract the engineered features from audio files [22]. It is an open source toolkit with a simple command line interface and is very simple to use and can be automated. Some configuration files are provided that extract different low level descriptors. The configuration file to be used is $IS09\_emotion.conf$ (Interspeech 2009) [21]. Although the configuration file has been designed for emotion recognition it has been used with successful results in animal classification in the past [41].

4.2 Automatic Feature Extraction

4.2.1 Raw Audio Data

Time series is the raw audio data. It simply is an array of amplitude values for each sample of the audio file.

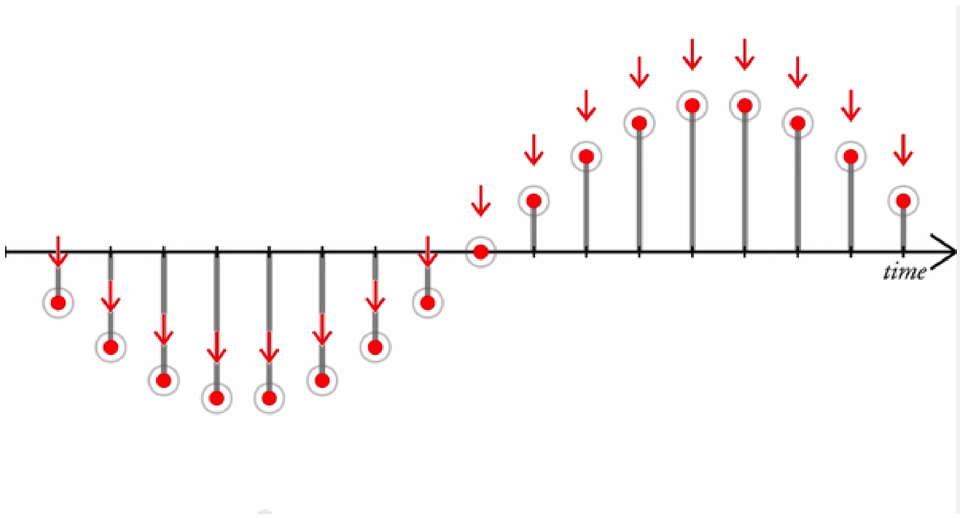


FIGURE 4.1. Amplitude of audio over the series of samples [46]

Time series measures relevant data at regular interval, and in audio the interval is the samples and the relevant data is the amplitude. The image above shows how the data is sampled at regular intervals. The number of features extracted is thus dependent on the length of the audio file and also the sample rate. Having analysed the number of features it seems that it is too large for some machine learning techniques and therefore will only be tested with Convolutional Neural Networks to extract features that seem to be useful in classifying the species.

4.2.2 Spectrogram

Spectrogram has been commonly applied in sound classifications particularly of animals [33]. The spectrogram is generated by using the Fast Fourier Transform algorithm (FFT). The timeseries data is divided into overlapping segments and then FFT algorithm is applied to obtain the magnitude, which is then plotted on the y axis with time on the x axis. However the spectrogram will also include background noise and this can greatly affect classification. To deal with this, background noise will be removed through a series of morphological image manipulation techniques that have been applied to a similar context [33]. In particular median clipping which calculates the median of every row and column and if the pixel value in a particular row and column is more than the respective medians by a factor of $\frac{1}{8}$ then it is set to 255 (white), dilation filter which expands regions through the use structuring element and closing filters which will remove small background holes that are not relevant [33]. Figure 4.2 below shows a spectrogram before and after the noise removal is applied and the difference is quite explicit. The axis have been removed from the image as it is not relevant in the classification since the scale is kept constant.

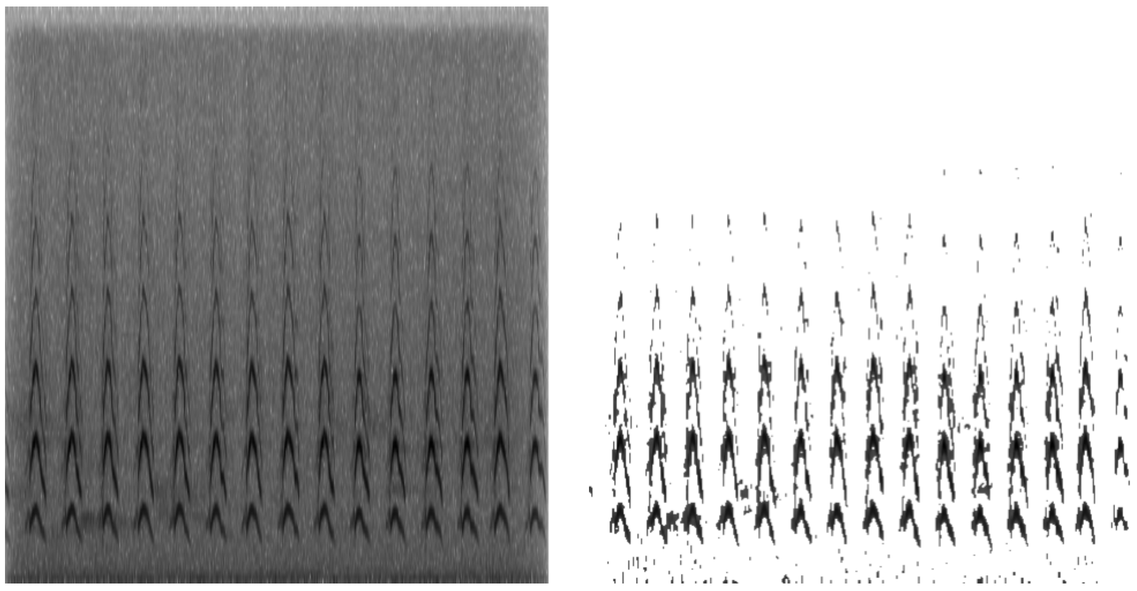


FIGURE 4.2. Left shows spectrogram obtained directly from sound file, right shows after morphological processing

4.2.3 Discrete Wavelet Transform

Wavelets is commonly used in signal processing, the key concept is to analyse the different frequencies in the audio file. Unlike Fourier Transform which provides a global approximation, Wavelets provide local approximations. A moving window is used which creates spectrum in that window, this window is then shifted along and the size of the window is varied.

Daubechies Wavelets are orthogonal wavelets that use vanishing moments to represent a signal. P vanishing moments, allows any polynomial signal up to order $P-1$ to be represented completely. A higher vanishing moment, can represent more complex signals accurately. A db$V$ Daubechies wavelets, has V vanishing moments and therefore it is important to choose the appropriate number of vanishing moments so that the number of features is not too large yet the signal is completely represented. Daubechies wavelets have been used for bird sound classification in particular db10 with satisfactory results [44]. The image below shows the different Daubechies Wavelet functions.

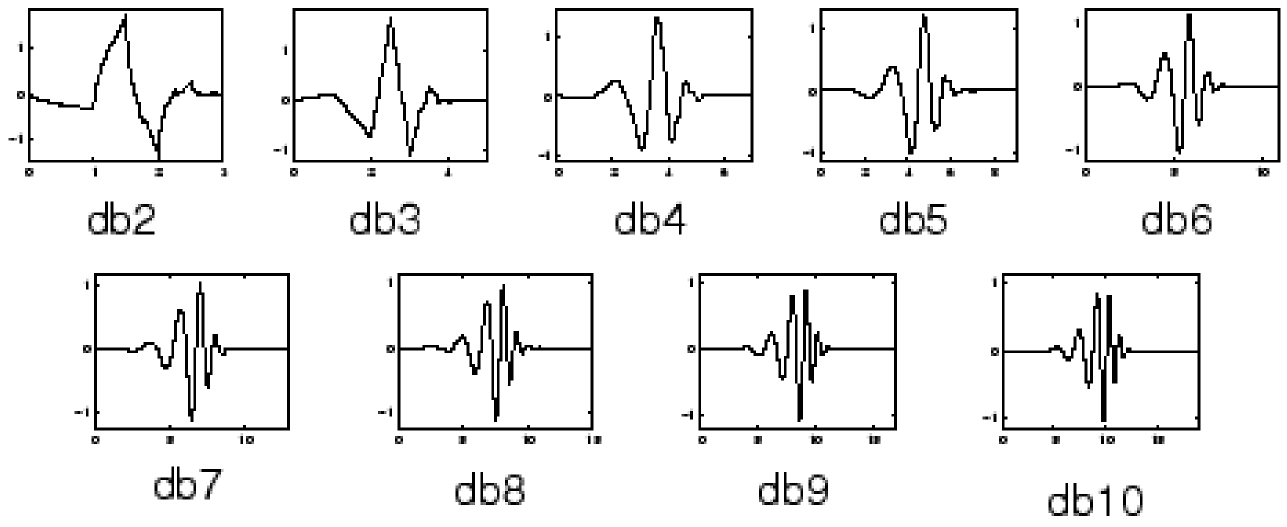


FIGURE 4.3. The different Daubechies Wavelets [36]

Chapter 5

Implementation

5.1 Overfitting

A model *m* overfits the training data, if there is another model *m'*, where *m* performs better on the training set than *m'* but *m'* has a better performance on the testing set [37], this is illustrated in Figure 5.1. The model will learn the noisy data too, and will incorrectly classify future instances. Underfitting on the other hand occurs when the model performs poorly on both the training set and the testing set. This is primarily caused by lack of variance or too much noise in the dataset and most importantly incorrect model parameters. Neural Network (NN) are also easily likely to overfit if the number of hidden nodes or layers are too large, leading to the network learning the training set perfectly. Overfitting is commonly avoided by using three disjoint sets. The first set is used to train the network (training set), the second set is used to optimise the parameters (validation set) and the final set is used to test the model. It is important that the instances in the test set do not occur in the training or validation set to ensure that the final performance is not biased. Another common measure is to reduce the dimension of the input, by feature selection algorithms which reduce the size of the input vector to only the features that affect the output. Also background noise from spectrogram representation of the sound files can be removed by applying Median Clipping, Dilation followed by Closing Filters [33]. Alternatively generic noise can be removed from the audio files by pre-processing the audio files using sound processing tool such as SoX [7].

5.2 Parameter Optimisation

Choosing the parameters plays an important role in the accuracy but also plays an important role on overfitting/underfitting. In Neural Network for example large number of layers lead to

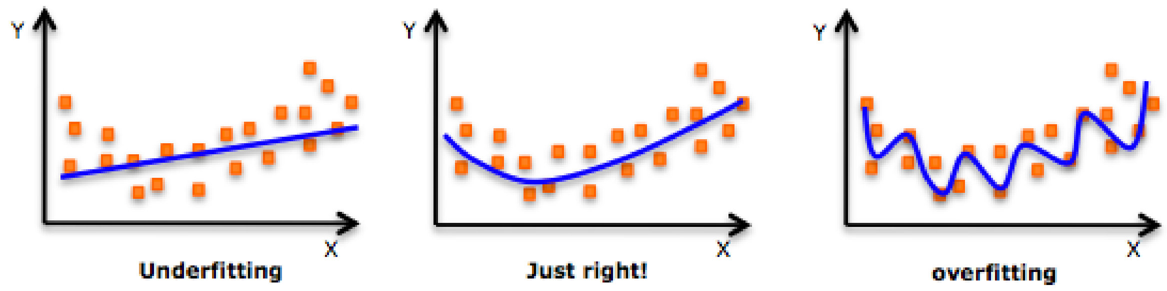


FIGURE 5.1. The blue line shows the learned function and the orange boxes show the points that were considered [26]

overfitting. Another important key parameter in Neural Networks is the learning rate, having a learning rate that is too high causes the adjustments of weight to be large and hence the network does not converge to optimality, whereas a low learning rate will cause the NN to converge slowly and can converge to a local optima as depicted in Figure 5.2.

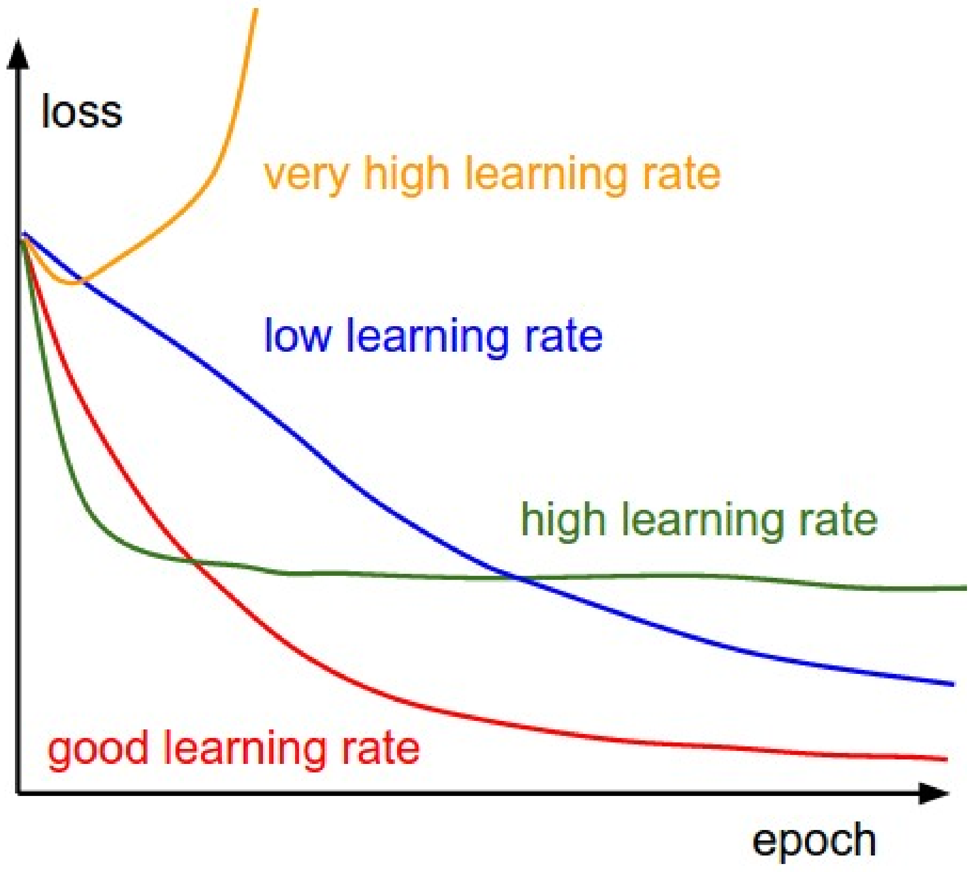


FIGURE 5.2. The loss should converge to 0 at an increasing rate over the epochs with a good learning rate [5]

Support Vector machines also has a lot of parameters that have to be optimised, the key parameters here are Cost and Gamma. The cost determines the error margin between data points and the hyperplanes, choosing a cost function that is too high can cause the model to overfit [3]. Gamma on the other hand is useful for non-linear classification, it determines the shape of the hyperplane.

5.3 Pre-processing

The dataset that will be used to train the model has some challenges. In particular it has been subjected to the imbalanced dataset problem, whereby not all classes have similar number of instances [15]. After removing classes with too few instance, to handle the remaining classes there were a few possible solutions and these are discussed in more detail below.

One way to deal with imbalanced dataset is by downsampling which involves reducing the number of instances by tossing away some instances of classes with high number of instances to match the classes with lower number of instances [27]. This has the advantage that it is simple to implement and has proven to be more effective than upsampling [19]. However this method has the major disadvantage that useful and valuable instances are thrown away upon training.

Another common solution to deal with imbalanced dataset is by upsampling which involves generating synthetic data or making multiple copies of the instances for classes with low number of instances (minority class) to match the majority class [27]. This has the advantage that it does not throw away useful instances. However this method has proved in the past to cause overfitting and can be computationally intensive if the number of instances and classes are very large [31].

A common solution that changes how the model learns instead of the dataset itself is known as Cost Sensitive Learning [31]. It works by changing the loss/error function to give a different error value to False Positive and False Negatives depending on which is more significant, and hence model will avoid the error with a higher error value. The advantage of this is that it changes the model instead of the data. However it can be ineffective on severely imbalanced datasets and does not guarantee convergence to optimal model [45].

Boosting like Cost Sensitive Learning is another technique that changes the model instead of the data. It trains multiple classifiers that improve on the errors of the previous classifiers. Again this technique has the advantage that it does not change the dataset itself. However it is prone to overfit the dataset [30].

Having considered each of the possible solution, a final decision was reached to not consider classes that have too few instances for the machine learning algorithm to learn anything useful. Upsampling the train set was decided as the solution to be applied on the remaining classes, this was chosen so that valuable data are not made redundant by downsampling. The dataset now consisted of only 36 species/15 families/8 orders.

5.4 Method

Besides ensuring that the same performance measures are used to evaluate a model, it is key to ensure that the same data is not used for training, validation and testing, since an instance in the test set may affect the result differently if it was in the training set and vice versa. To deal with this the dataset was shuffled and then split into three sets, which were then saved, so that upon evaluation of each classifier it was sufficient to read these data as training, validation and test set. Tables 3.2, 3.1 and 3.3 in chapter 3 show how many instances of a specific class were in each test set. Also to prevent overfitting, the model is optimised on the validation set and once the optimal performance has been obtained on the validation set, the performance on the test set is reported. This ensures that the model has not been trained to overfit the validation set and gives an approximate measure of how the model will perform to unseen real input. The performance recorded takes the sum of the three folds and then computes the performance measures instead of taking the average of all folds.

Python[1], Theano[2], Lasagne[3] and nolearn[4] were the libraries used for Convolutional Neural Network (CNN). The input used were Spectrograms, Time Series and Wavelets of the audio files as our input, this had the benefit that we didn't have to extract features ourselves, but CNN would identify and extract the most relevant features itself from the provided input. This however took very long to train, running one simple model on a high-end CPU took approximately 60 minutes for the dataset for one epoch, (each epoch can be considered to be a smaller subset of the input data) and training the network for a good accuracy requires large number of epochs (average 10^2), and will hence take four days and four hours. The model required to be improved iteratively by optimising the parameters.
One common method that is used to accelerate the training of CNNs is by using GPU since it has been designed effectively to process matrices very efficiently. We used CUDA enabled graphics machine for this and with this optimisation one epoch took only a few minutes which is faster by a magnitude of just under 60, and thus will only take less than one day to train one model. This along with early stopping meant that we didn't always have to run as much epochs as required and can hence train it quicker than by purely using the CPU for training it.

[1]`https://www.python.org/`
[2]`http://www.deeplearning.net/software/theano/`
[3]`http://lasagne.readthedocs.io/en/latest/`
[4]`https://pythonhosted.org/nolearn/`

Analysing different parameters of a model are independent of each other and can thus be parallelised by keeping one parameter as a variable and the other parameters constant. HTCondor[5] allows just that. It takes an executable file and the different parameters then when submitted to the queue, the different parameters are passed on to the executable and are executed on different machines when they are idle. With this optimisation, it was possible to search thousand different model parameters in the time it takes to evaluate just one model parameter.

[5] `https://research.cs.wisc.edu/htcondor/`

CHAPTER 6

RESULTS

The success of this project will be determined by how well the model is able to successfully classify previously unseen instances provided that the classes have been previously encountered in the training data. Despite having a large dataset as mentioned before it was subjected to the data imbalance problem and as a result some classes were not considered due to insufficient instances being available. As a result, the data available had now decreased and although the data available was still quite large (>3000), it was concluded that the dataset would be split into 3 sets and each of these set would be used as a training set, validation set and test set in a round robin order. This is so that the model is does not overfit the validation set. Cross-validation ensures that every instance is used for testing, validation and also for training respectively in each fold, in addition the performance is measured on the final confusion matrix thus if a particular fold has a higher accuracy this will not affect the final accuracy if the other folds have a lower accuracy [9].

There is an issue that with some classes they may be confused a lot with other classes and this can be observed with a confusion matrix, which has the actual class as the rows and predicted class as the columns. Therefore the diagonal entries shows the number of times the classes were correctly identified, known as the True Positives (TP) and True Negatives (TN) [35]. For each class the False Positive (FP) occurs when the other classes are falsely identified as being the positive (current) class. False Negative (FN) on the other hand occurs when the current class is falsely identified as being one of the other class. Using these metrics some further useful metrics can be calculated, in particular the precision, recall and F_α.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{6.1}$$

$$Precision = \frac{TP}{TP+FP} \tag{6.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.3}$$

$$F_{\alpha} = \frac{(1+\alpha)^2 * Precision * Recall}{\alpha^2 * Precision + Recall} \tag{6.4}$$

Recall is the fraction of instances that are relevant and correctly identified. Precision on the other hand, gives the fraction of instances that are correctly predicted as being the right class [13] Combining these two values gives the F_{α} value, which allows us to compare the two values as one. In a multiclass problem, a common way approach is to take the unweighted average recall (UAR) which simply takes the average of the recall calculated for each class and this helps in identifying any classification errors, particularly in an imbalanced dataset.

		Predicted	
		1	**2**
Actual	**1**	TP	FP
	2	FN	TN

Table 6.1: A confusion matrix for a binary classification with class 1 as the current class

6.1 Support Vector Classification

The key parameters for SVC were the kernel, gamma and C. The kernel determines the function that will be used to predict the output using the given input e.g. Radial Basis Function will have a more curved function than a linear kernel. Gamma gives a measure of how much a single example influences the radius; a high gamma means that the radius should be close. C decides the error margin, a high C will lead to a smaller margin hyperplane, whereas a low C will choose a hyperplane with large margins between examples at the cost of misclassifying some examples. All the ranks (i.e. Species, Family and Order) followed the same pattern for the key parameters mentioned above. Radial Basis function appeared to be the best kernel and 0.1 was the best gamma, C seemed to make not much of an impact on the UAR.

	Order	Family	Species
Kernel	RBF	RBF	RBF
Gamma	0.1	0.1	0.1
C	30	90	50
Accuracy	75.7	82.3	84.1
UAR	69.9	80.5	82.8
UAP	68.3	78.7	82.7
Unweighted Average F1 Value	69.1	79.6	82.8

Table 6.2: Optimal Parameters and results on Test set for SVC, where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively. The optimal kernel for all the ranks was the Radial Basis Function (RBF).

The Confusion Matrix below shows how much confusion there was for the classification of order for the test set and it can be seen that class 1 was quite frequently confused with classes 5 and 6, this may be due to the fact that despite upsampling the training set, there may not have been much variance in the training data of class 1. The other classes although there is some confusion it is quite small relative to the number of times it was correctly classified.

		Predicted Class							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual Class	**1**	**32.76%**	6.90%	5.17%	6.90%	20.69%	18.97%	5.17%	3.45%
	2	6.12%	**57.14%**	9.18%	0.00%	6.12%	8.16%	6.12%	7.14%
	3	0.60%	1.20%	**81.44%**	1.80%	9.58%	3.59%	1.20%	0.60%
	4	3.23%	1.21%	6.05%	**79.84%**	3.23%	2.02%	2.82%	1.61%
	5	2.71%	2.71%	11.24%	1.55%	**74.03%**	3.88%	0.78%	3.10%
	6	2.01%	2.01%	4.36%	3.02%	5.03%	**77.85%**	3.36%	2.35%
	7	1.30%	1.74%	7.38%	3.04%	4.56%	4.34%	**75.70%**	1.95%
	8	0.86%	2.58%	1.07%	2.15%	2.15%	6.01%	5.15%	**80.04%**

Table 6.3: Confusion Matrix obtained with Support Vector Classification for Order on Test Set

6.2 NuSVC

This is very closely related to SVC but the C parameter here is replaced with a nu parameter. Nu takes a value that is strictly greater than zero and less than or equal to one, it provides a relative upper bound on the training errors and a relative lower bound on the support vectors. RBF kernel again seemed to be the best kernel to use, and gamma again followed the same pattern as above, but nu unlike C affected the UAR and optimal UAR was reached at different values depending on the categories and the nu values are mentioned in Table 6.4. Figure 6.1 below shows how the UAR changes for different values of Nu for Order.

	Order	Family	Species
Kernel	RBF	RBF	RBF
Gamma	0.1	0.1	0.1
Nu	0.201	0.001	0.001
Accuracy	78.4	84.9	86.9
UAR	70.4	81.8	84.9
UAP	70.8	82.8	85.8
Unweighted Average F1 Value	70.6	82.3	85.3

Table 6.4: Optimal Parameters and results on Test set for NuSVC, where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively. The optimal kernel for all the ranks was the Radial Basis Function (RBF).

The confusion matrix for order below again shows the same confusion patterns as before. Classes with fewer instances are recognised slightly less accurately than before however classes 4,5,6,7 and 8 are recognised with a better accuracy than before.

Actual Class \ Predicted Class	1	2	3	4	5	6	7	8
1	**29.31%**	6.90%	5.17%	8.62%	22.41%	18.97%	5.17%	3.45%
2	6.12%	**51.02%**	7.14%	0.00%	7.14%	9.18%	9.18%	10.20%
3	0.60%	1.80%	**75.45%**	1.80%	11.98%	3.59%	4.19%	0.60%
4	2.42%	0.40%	3.63%	**85.48%**	2.42%	2.02%	1.61%	2.02%
5	2.71%	1.94%	7.75%	1.16%	**76.74%**	3.88%	1.55%	4.26%
6	1.34%	1.68%	3.69%	3.02%	4.03%	**78.86%**	4.36%	3.02%
7	0.87%	0.65%	3.47%	3.04%	2.60%	3.47%	**84.16%**	1.74%
8	0.86%	1.72%	0.86%	1.93%	2.15%	5.58%	4.51%	**82.40%**

Table 6.5: Confusion Matrix obtained with NuSVC for Order on Test Set

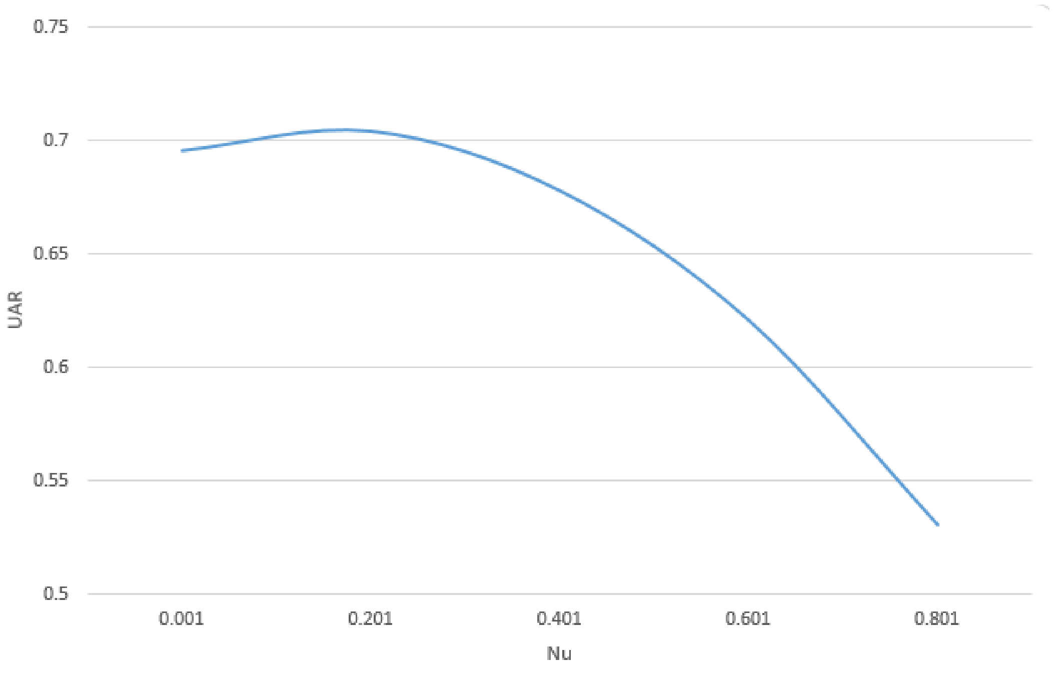
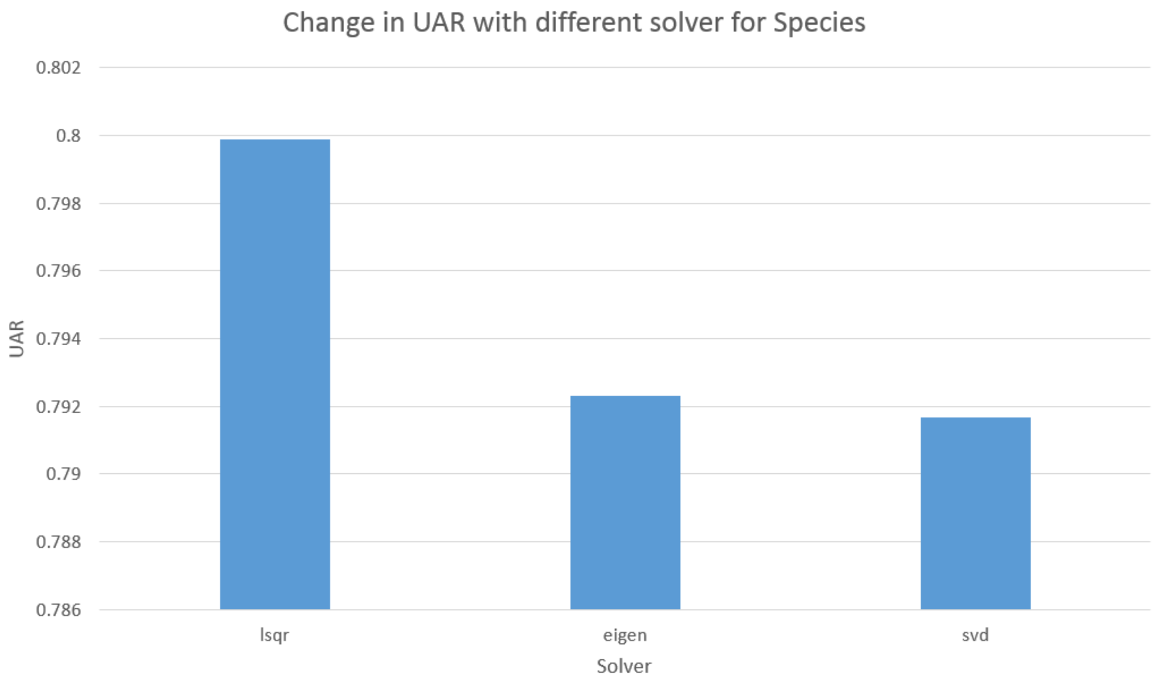


Figure 6.1: Graph showing how UAR changes with different Nu values for Order

6.3 Linear Discriminant Analysis

The significant parameters here are the solver used and the shrinkage value. The different solvers were Singular Value Decomposition (SVD), Least Squares (LSQR) and Eigenvalue decomposition (Eigen) and they are used to perform dimensionality reduction. The shrinkage value only works on LSQR and Eigenvalue and it is used to improve the estimation of the covariance matrices, it can take a value between zero and one or could be 'auto' or 'none', zero denotes that no shrinkage happens and so the empirical covariance matrix is used. Different ranks had different optimal solver and the UAR for the different solver for species is shown below.



	Order	Family	Species
Solver	Eigen	LSQR	LSQR
shrinkage	0.201	0.401	0.001
n_components	30	140	105
Accuracy	67.0	67.5	81.0
UAR	64.1	68.3	80.0
UAP	60.7	64.1	79.2
Unweighted Average F1 Value	62.4	66.1	79.5

Table 6.6: Optimal Parameters and results on Test set for LDA, where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively, where Eigenvalue decomposition (Eigen) and Least Square solution (LSQR) proved to be the optimal solvers.

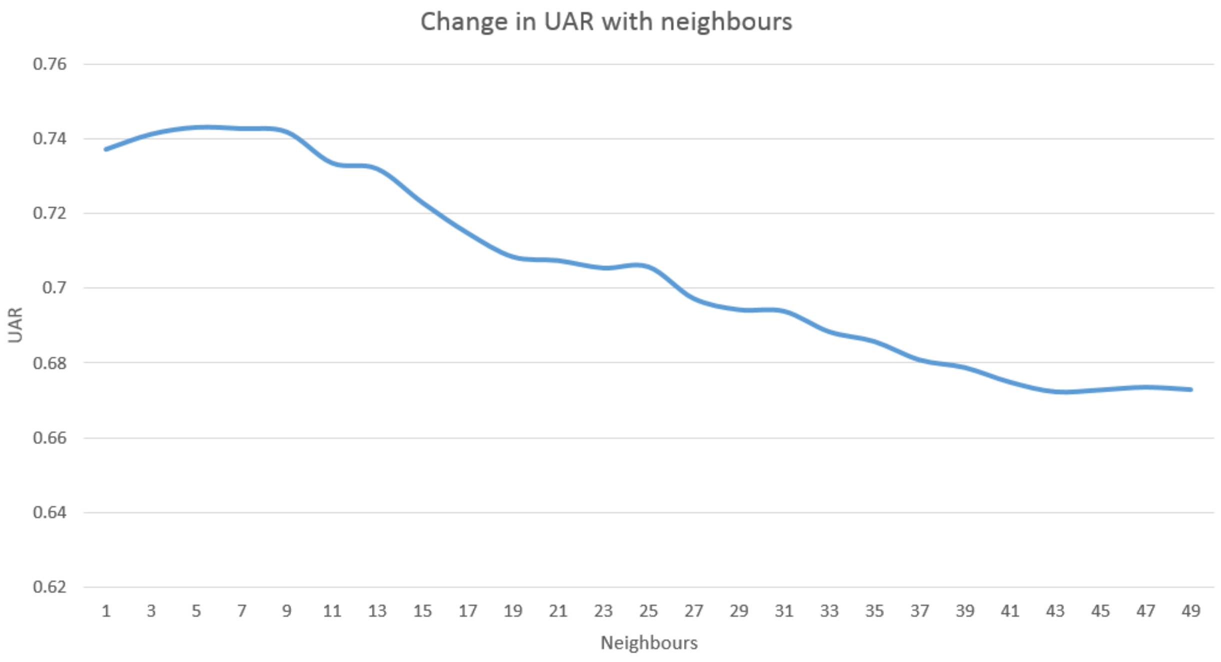
Unlike with SVC, class 1 is confused less often here, but classes with more instances are confused more, this may be due to the fact that the large number of instances may have caused misclassification since the variances between all the instances of the same class could have been quite large.

		Predicted Class							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual Class	**1**	**50.00%**	3.45%	5.17%	10.34%	10.34%	10.34%	6.90%	3.45%
	2	6.12%	**47.96%**	13.27%	0.00%	6.12%	11.22%	8.16%	7.14%
	3	2.40%	4.19%	**79.04%**	1.20%	8.98%	3.59%	0.60%	0.00%
	4	8.87%	1.21%	6.05%	**67.74%**	4.84%	8.47%	2.02%	0.81%
	5	7.36%	1.94%	12.40%	3.49%	**63.57%**	6.59%	3.10%	1.55%
	6	9.40%	4.70%	5.37%	3.36%	6.04%	**66.78%**	1.68%	2.68%
	7	3.69%	5.86%	8.46%	3.90%	3.69%	8.68%	**64.43%**	1.30%
	8	3.43%	5.36%	1.50%	3.00%	3.86%	7.51%	2.36%	**72.96%**

Table 6.7: Confusion Matrix obtained with Linear Discriminant Analysis for Order on Test Set

6.4 K Nearest Neighbour

Since it is a Lazy learning method, the training time was really short as upon training the data is just stored. The important parameters here are the number of neighbours, weights and algorithm. The number of neighbour specifies the number of neighbours that should be considered for each instance for classification. The weights on the other hand determines the function used to calculate the weight of the neighbours, this includes 'distance' which uses the inverse of the distance between the points, '$uniform$' which weighs all neighbours equally. Algorithm is used to find the nearest neighbours; '$brute$' for instance will search for the nearest neighbour using a brute force approach and as expected will be slow, whereas 'kd_tree' stores the data in an ordered manner so that retrieval will be quicker, however suffers from the curse of dimensionality as the number of dimension increases. The number of neighbours was dependent on the category since the different ranks had different optimal number of neighbours. The graph below shows how the UAR varies for family with the number of neighbours.



The confusion matrix for order shown below shows that class 1 is confused quite frequently with class 6, this may be due to the fact that the features extracted for class 1 and class 6 may have high similarities, and could be along the same region, thus even with 1 neighbour it is confused with class 5.

	Order	Family	Species
Neighbours	1	5	1
Weights	distance	distance	uniform
Algorithm	kd_tree	kd_tree	kd_tree
LeafSize	25	20	35
Accuracy	75.9	75.6	78.0
UAR	68.9	74.3	77.2
UAP	68.4	71.9	76.8
Unweighted Average F1 Value	68.6	73.1	77.0

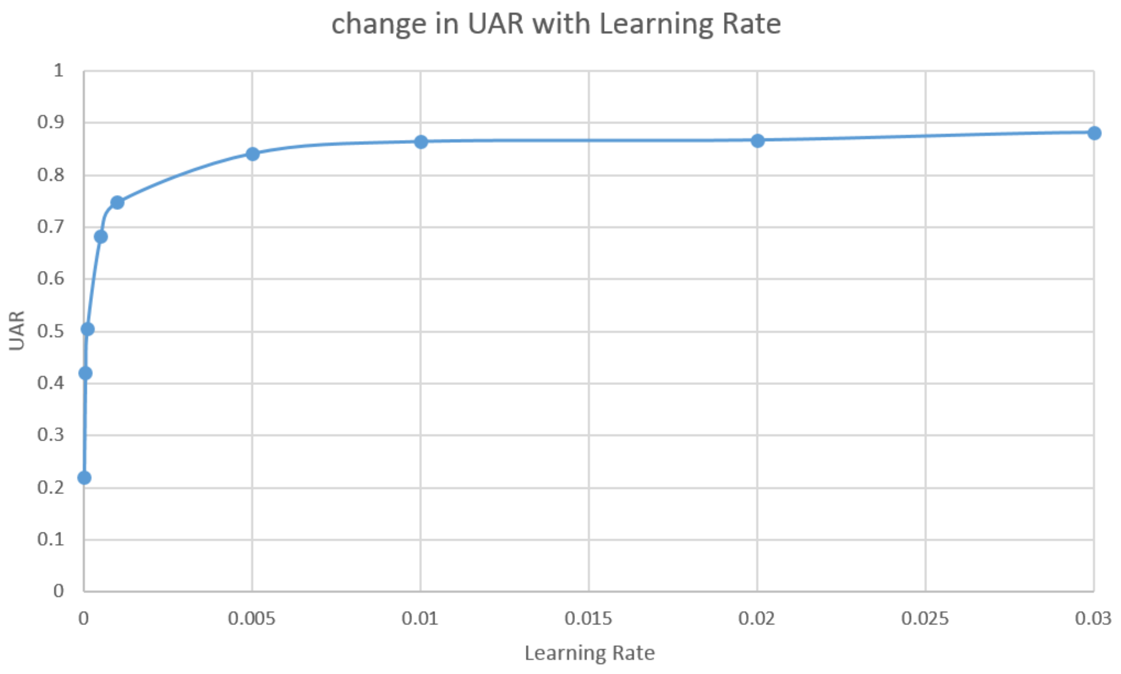
Table 6.8: Optimal Parameters and results on Test set for KNN, where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively, where distance assigns the inverse of the distance as the weight to each neighbour and uniform assigns a uniform value to the weight of each neighbour. K-Dimensional Tree (kd_tree) structure was used as the algorithm for quick retrieval

		Predicted Class							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual Class	**1**	**34.48%**	6.90%	8.62%	0.00%	15.52%	24.14%	6.90%	3.45%
	2	3.06%	**53.06%**	9.18%	2.04%	5.10%	12.24%	8.16%	7.14%
	3	1.20%	2.99%	**67.66%**	3.59%	10.78%	5.39%	7.19%	1.20%
	4	1.21%	0.81%	2.42%	**89.52%**	4.44%	0.00%	1.21%	0.40%
	5	6.20%	1.16%	6.59%	2.33%	**70.93%**	6.98%	2.71%	3.10%
	6	1.68%	2.35%	4.36%	3.69%	4.36%	**73.49%**	5.37%	4.70%
	7	1.08%	2.17%	2.82%	2.17%	3.04%	5.42%	**78.74%**	4.56%
	8	1.50%	0.86%	2.36%	1.50%	2.36%	5.36%	3.00%	**83.05%**

Table 6.9: Confusion Matrix obtained with K-Nearest Neighbour for Order on Test Set

6.5 Artifical Neural Network

Various network topology were investigated here. Up to two hidden layers were used, this is to ensure that the network is able to learn any complex functions while ensuring that it doesn't overfit. The learning rate was kept small so that the network converges, and it was reduced exponentially to find the optimum at a quicker rate, since the training time is much longer than the previous methods discussed here. The number of neurons in the hidden layers was also a hyperparameter that had to be optimised explicitly, it was ensured that the number of neurons at each layer was strictly less than the number of neurons in the previous layers and greater than the number of neurons in the succeeding layer. This was so that at each succeeding layer the number of features is reduced until at the output layer it is reduced to only one node per class. The graph below shows how the UAR changes for different values of learning rate for Family, it can be clearly seen that as the learning rate increases, the UAR also increases and after 0.03 starts to decline at a slower rate although not shown as part of the figure.



The confusion matrix for order shown below shows that there is much less confusion in all classes. Class 1 is recognised almost twice as accurately than in the previous methods, and all the other classes are recognised more accurately and this is reflected in the UAR, since it is the highest UAR obtained so far.

	Order	Family	Species
Layers	[384,192,80,8]	[384,192,15]	[384,192,36]
Learning Rate	0.01	0.03	0.02
Momentum	0	0	0
Accuracy	83.7	88.9	91.2
UAR	79.6	88.2	90.7
UAP	80.2	87.6	90.3
Unweighted Average F1 Value	79.9	87.9	90.5

Table 6.10: Optimal Parameters and results on Test set for ANN, where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively. The layers show the shape of the network where the first entry shows the size of the input layer and the last element shows the size of the output layer, and the remaining elements show the size of the hidden layer(s).

		Predicted Class							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual Class	**1**	**58.62%**	3.45%	0.00%	6.90%	6.90%	13.79%	5.17%	5.17%
	2	5.10%	**65.31%**	4.08%	1.02%	5.10%	8.16%	8.16%	3.06%
	3	0.60%	0.60%	**88.02%**	1.80%	5.39%	1.20%	1.20%	1.20%
	4	0.00%	0.00%	1.61%	**90.32%**	2.82%	1.61%	2.82%	0.81%
	5	1.16%	1.55%	5.81%	2.33%	**78.68%**	3.49%	2.71%	4.26%
	6	1.34%	1.68%	1.68%	3.02%	2.68%	**82.89%**	4.70%	2.01%
	7	0.65%	1.52%	1.74%	2.39%	2.60%	1.95%	**85.25%**	3.90%
	8	0.21%	0.86%	0.21%	3.22%	1.50%	2.79%	3.65%	**87.55%**

Table 6.11: Confusion Matrix obtained with Artificial Neural Network for Order on Test Set

6.6 Convolutional Neural Networks

The key parameters in convolutional neural networks were the architecture and the learning rate. Unlike with ANN the architecture plays a bigger role since different type of layers are available here, and fine tuning the network is a time consuming process particularly with the long training time. The layers that were used include Convolutional Layers, Max Pooling Layers, Dropout Layers and Dense Layers. Dropout layer allows the network to combine several different architectures by "dropping" neurons in a layer with a probability p. The learning rate also plays a key role in the network converging to the global optima. Momentum and Adaptive Learning Rate were used to avoid the network converging to a local optima, this is done by adapting the learning rate depending on the error observed.

6.6.1 Raw Audio

	Order	Family	Species
Network Architecture	input size 1x40000, conv1 filters 30 size 5, pool1_pool_size=(10), conv2 filters 20 size 10, pool2_pool_size=(10), hidden4_num_units=2000, hidden5_num_units=1000, output_num_units=8	input size 1x40000, conv1 filters 10 size 5, pool1_pool_size=(10), conv2 filters 10 size 20, pool2_pool_size=(10), hidden4_num_units=2000, hidden5_num_units=1000, output_num_units=15	input size 1x40000, conv1 filters 10 size 5, pool1_pool_size=(10), conv2 filters 10 size 20, pool2_pool_size=(10), hidden4_num_units=2500, hidden5_num_units=1000, output_num_units=36
Learning Rate	0.04	0.04	0.04
Accuracy	44.2	39.1	37.6
UAR	40.0	38.4	36.9
UAP	40.0	36.8	36.8
Unweighted Average F1 Value	40.0	37.6	36.8

Table 6.12: Optimal parameters and performance on Test set for CNNs with raw audio, where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively

It seems that although the network has learned some characteristics since the performance measure is much higher than the chance levels, it is quite low relative to previous models. In addition previously the performance increased from order to species, since the taxonomic rank was becoming more specific, here however the opposite is observed. There is a lot of confusion with class 1, 2 and 5. Class 5 is predicted as class 3, more often than it is predicted correctly.

		Predicted Class							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual Class	**1**	**18.97%**	12.07%	18.97%	10.34%	12.07%	10.34%	5.17%	12.07%
	2	5.10%	**21.43%**	9.18%	11.22%	5.10%	16.33%	11.22%	20.41%
	3	2.99%	7.78%	**58.08%**	10.78%	10.18%	4.79%	3.59%	1.80%
	4	4.44%	4.84%	11.69%	**52.42%**	1.61%	6.45%	2.42%	16.13%
	5	8.91%	7.36%	36.43%	5.43%	**29.46%**	5.43%	2.33%	4.65%
	6	4.03%	11.41%	7.72%	7.38%	2.35%	**44.63%**	10.40%	12.08%
	7	3.69%	8.24%	11.28%	7.59%	2.82%	10.41%	**37.53%**	18.44%
	8	1.29%	7.94%	3.43%	8.80%	1.72%	6.87%	12.66%	**57.30%**

Table 6.13: Confusion Matrix obtained with Convolution Neural Network and raw audio for Order on Test Set

6.6.2 Spectrogram

	Order	Family	Species
Network Architecture	input size 1x125x125, conv1 filters 64 size 2x2, pool1_pool_size=(2,2), conv2 filters 64 size 1x1, pool2_pool_size=(2,2), conv3 filters 64 size 2x2, pool3_pool_size=(2,2), hidden4_num_units=3000, hidden5_num_units=1500, output_num_units=8	input size 1x125x125, conv1 filters 64 size 2x2, pool1_pool_size=(2,2), conv2 filters 64 size 1x1, pool2_pool_size=(2,2), conv3 filters 64 size 2x2, pool3_pool_size=(2,2), hidden4_num_units=3000, hidden5_num_units=1500, output_num_units=15	input size 1x125x125, conv1 filters 32 size 20x20, pool1_pool_size=(2,2), conv2 filters 32 size 10x10, pool2_pool_size=(2,2), hidden4_num_units=3000, hidden5_num_units=1500, output_num_units=36
Learning Rate	0.045	initially 0.0001 with adam (adaptive learning rate)	0.02
Accuracy	57.0	60.9	65.5
UAR	51.0	55.6	61.7
UAP	49.5	56.1	61.0
Unweighted Average F1 Value	50.2	55.8	61.4

Table 6.14: Optimal parameters and performance on Test set for CNNs with spectrogramson CNNs with where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively

Spectrograms performed the best with CNNs, this may be since spectrograms are 2 dimensional image data while the others were 1 dimensional, and CNN were designed for feature extraction from images. However the performance is still much lower than those obtained using ANN and openSMILE extracted features as input. The confusion matrix shows that only class 1 and 2 are confused quite frequently and the others are predicted with a relatively low confusion. The figures below shows the convolutional layers learned by the network for species, in the first convolutional layer. Figure 6.3 shows the convolutional layer applied to an input appears, a few of the layers in the figure appear black due to the layers not training on the input and due to the random weight initialisations. Figure 6.4, shows the input along with the critical regions respectively, the image shows the input, the critical parts highlights the region that the network believes to be critical in classifying and the super-imposed image shows the two imposed. It can clearly be seen that the regions highlighted are indeed where the unique to that image. Figure 6.5 shows the network architecture that lead to the optimal performance for species.

		Predicted Class							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual Class	**1**	**15.52%**	6.90%	24.14%	12.07%	8.62%	20.69%	1.72%	10.34%
	2	1.02%	**9.18%**	17.35%	13.27%	5.10%	19.39%	6.12%	28.57%
	3	4.19%	1.20%	**57.49%**	16.17%	8.98%	7.78%	1.80%	2.40%
	4	2.42%	2.02%	13.71%	**51.61%**	0.81%	14.11%	0.81%	14.52%
	5	4.65%	3.88%	52.33%	6.59%	**18.99%**	8.91%	0.78%	3.88%
	6	1.68%	2.35%	7.72%	10.07%	2.35%	**62.75%**	1.68%	11.41%
	7	5.42%	5.64%	14.32%	6.94%	4.12%	11.93%	**32.10%**	19.52%
	8	2.58%	4.51%	6.87%	7.51%	0.86%	12.88%	10.30%	**54.51%**

Table 6.15: Confusion Matrix obtained with Convolution Neural Network and spectrograms for Order on Test Set

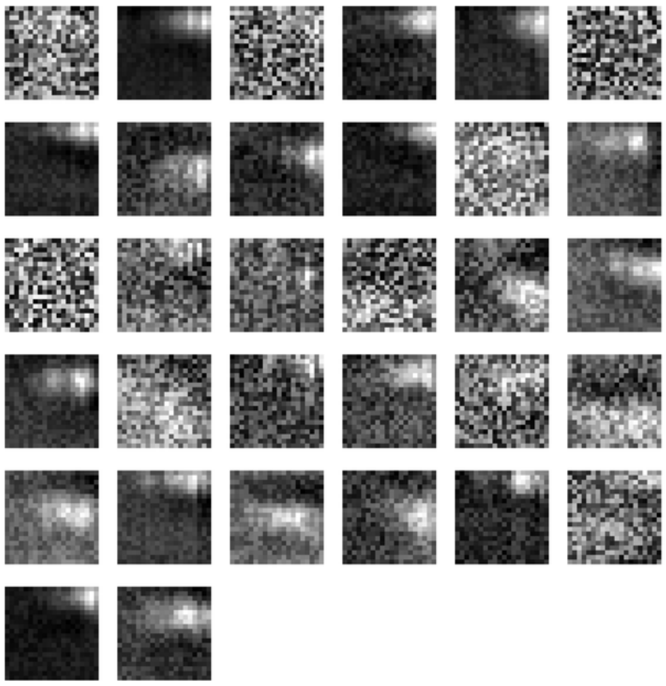


Figure 6.2: Filters of Convolutional Layer learned by the CNN, white pixels show the region that will be extracted and transferred to the next layer. It can be seen that the key features are concentrated around the top right corner, as some filters focus primarily on these region. Also some filters also convolve along the frequency axis too suggesting that combining frequencies and time also helps in the classification.

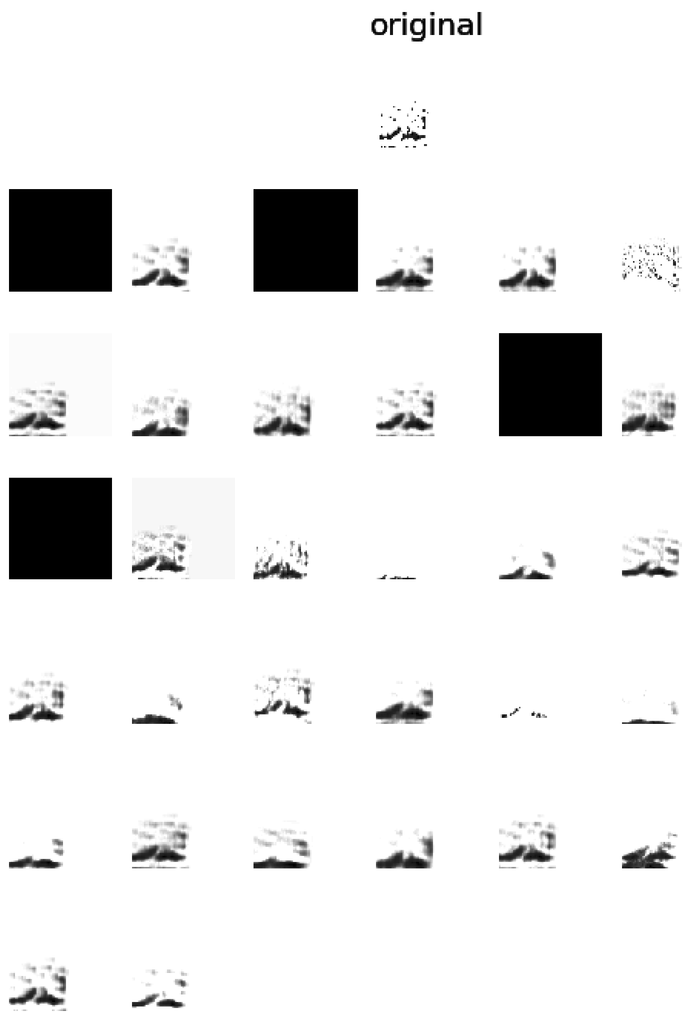


Figure 6.3: Filters of Convolutional Layer learned by the CNN applied to an example input

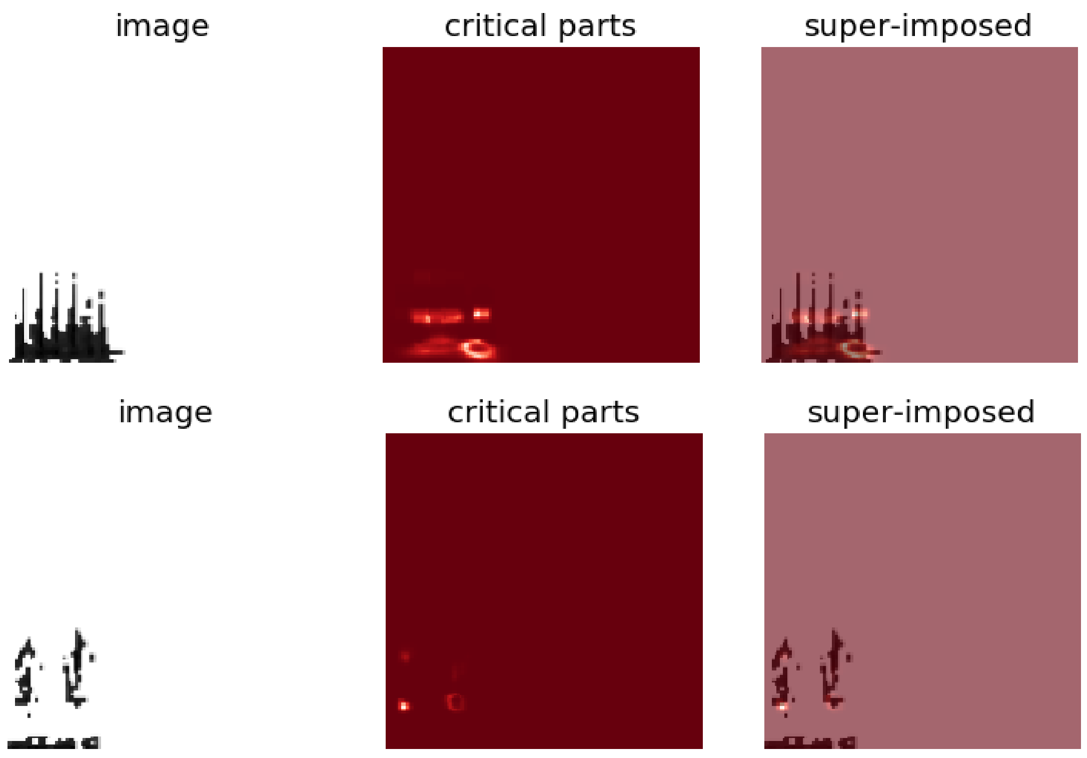


Figure 6.4: Plot showing parts of the image that are particularly important for the network to classify the image correctly.

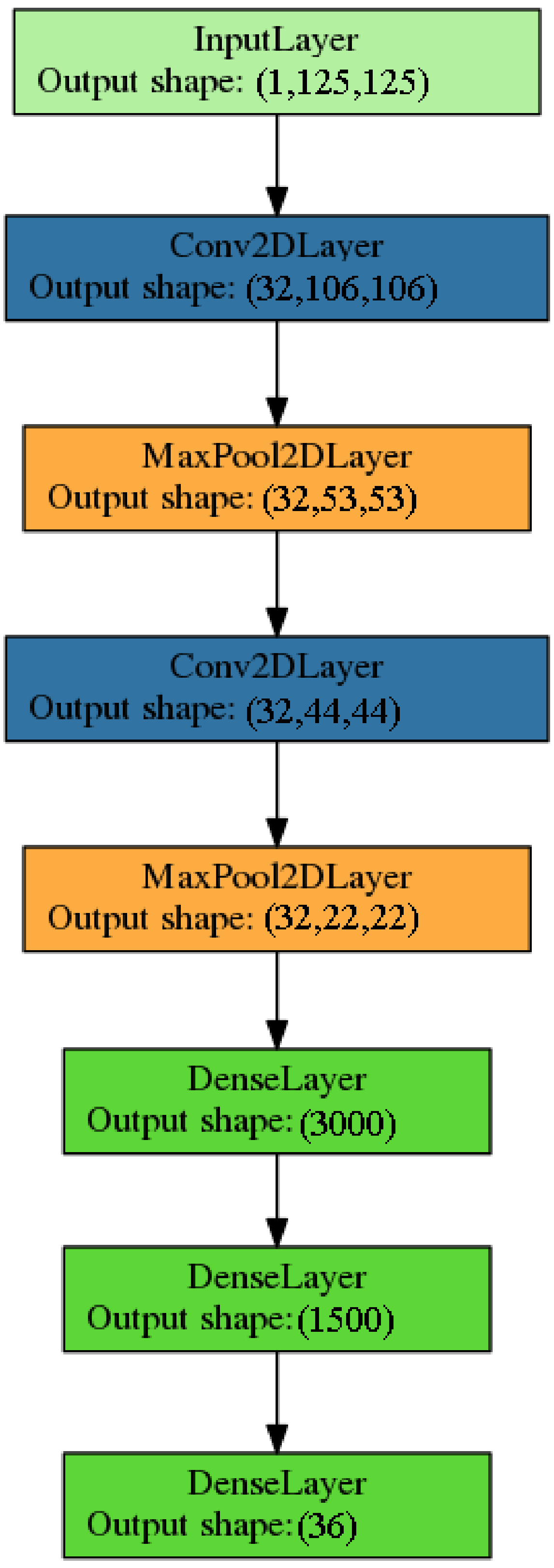


Figure 6.5: Visualisation of the network architecture for Species, one dimensional layers that convolve only across the time axis were also tested but led to performance lower than that obtained for two dimensional layers.

6.6.3 Wavelets

	Order	Family	Species
Network Architecture	input size 1x20009, conv1 filters 10 size 50, pool1_pool_size=(5), conv2 filters 5 size 10, pool2_pool_size=(2), hidden4_num_units=5000, hidden5_num_units=1000, output_num_units=8	input size 1x20009 conv1 filters 10 size 100, pool1_pool_size=(5), conv2 filters 5 size 100, pool2_pool_size=(2), hidden4_num_units=5000, hidden5_num_units=1000, output_num_units=15	input size 1x20009 conv1 filters 100 size 10, pool1_pool_size=(1), conv2 filters 50 size 5, pool2_pool_size=(2), conv3 filters 20 size 3, pool3_pool_size=(3), conv4 filters 10 size 2, pool4_pool_size=(4), conv5 filters 5 size 2, pool5_pool_size=(5), hidden4_num_units=500, hidden5_num_units=1000, output_num_units=36
Learning Rate	0.02	0.02	0.01
Accuracy	42.8	34.7	33.5
UAR	37.8	34.4	32.3
UAP	38.3	34.3	31.0
Unweighted Average F1 Value	38.0	34.3	31.6

Table 6.16: Optimal parameters and performance on Test set for CNNs with wavelets, where chance levels are 12.5%, 6.67% and 2.78% for Order, Family and Species respectively

The performance is very similar to the performance obtained from raw audio although it is higher here. The confusion matrix below for Order shows a similar pattern to that was observed in raw audio. The confusion matrix below shows the findings for order and it can be seen that only classes 3, 4, 6, 7 and 8 are recognised with relatively less confusion, the other classes are subjected to a large amount of confusion, and in fact the actual classes are predicted less often than other classes.

Actual Class	Predicted Class: 1	2	3	4	5	6	7	8
1	**22.41%**	12.07%	12.07%	0.00%	6.90%	17.24%	17.24%	12.07%
2	5.10%	**28.57%**	5.10%	7.14%	8.16%	15.31%	20.41%	10.20%
3	5.39%	1.20%	**66.47%**	2.99%	11.38%	7.78%	2.99%	1.80%
4	1.21%	3.23%	6.05%	**64.52%**	3.63%	8.06%	3.63%	9.68%
5	4.03%	4.44%	23.79%	6.45%	**48.79%**	6.85%	0.40%	5.24%
6	3.69%	5.03%	5.70%	7.38%	7.38%	**51.01%**	4.70%	15.10%
7	1.52%	4.56%	11.93%	3.69%	6.72%	5.64%	**57.70%**	8.24%
8	1.72%	2.79%	2.15%	3.43%	5.58%	4.94%	10.94%	**68.45%**

Table 6.17: Confusion Matrix obtained with Convolution Neural Network and wavelets for Order on Test Set

6.7 Summary of Results

The tables below show the performance of all the classifiers. It can be seen that almost all the classifiers have consistent performance across the different taxonomic rank. Since all the classifiers' performance has exceeded the chance level we can conclude that the features investigated here are indeed capable of differentiating between birds.

Table 6.18: Summary of classifiers for Order, chance level: 12.5%

Classifier	Accuracy	UAR	UAP	Unweighted Average F-1 Value
ANN	83.7	79.6	80.2	79.9
NuSVC	78.4	70.4	70.8	70.6
SVC	75.7	69.9	68.3	69.1
KNN	75.9	68.9	68.4	68.6
LDA	67.0	64.1	60.7	62.4
CNN - spectrograms	57.0	51.0	49.5	50.2
CNN - raw audio	44.2	40.0	40.0	40.0
CNN - wavelets	42.8	37.8	38.3	38.0

Table 6.19: Summary of classifiers for Family, chance level: 6.67%

Classifier	Accuracy	UAR	UAP	Unweighted Average F-1 Value
ANN	88.9	88.2	87.6	87.9
NuSVC	84.9	81.8	82.8	82.3
SVC	82.3	80.5	78.7	79.6
KNN	75.6	74.3	71.9	73.1
LDA	67.5	68.3	64.1	66.1
CNN - spectrograms	60.9	55.6	56.1	55.8
CNN - raw audio	39.1	38.4	36.8	37.6
CNN - wavelets	34.7	34.4	34.3	34.3

Table 6.20: Summary of classifiers for Species, chance level: 2.78%

Classifier	Accuracy	UAR	UAP	Unweighted Average F-1 Value
ANN	91.2	90.7	90.3	90.5
NuSVC	86.9	84.9	85.8	85.3
SVC	84.1	82.8	82.7	82.8
LDA	81.0	80.0	79.0	79.5
KNN	78.0	77.2	76.8	77.0
CNN - spectrograms	65.5	61.7	61.0	61.4
CNN - raw audio	37.6	36.9	36.8	36.8
CNN - wavelets	33.5	32.3	31.0	31.6

Chapter 7

Conclusions and Outlook

This section will reflect on the results obtained from this project and also any further works that could be done.

Extensive research has been done previously in this area with satisfactory results. However they primarily focused on the classification of only species. In addition only a specific feature set or a specific classifier was tested and hence the result obtained was only a good indicator of that specific method applied to bird classification not of bird classification in general. In this project numerous classifiers and feature sets were investigated for classifying bird sounds to its species, family and order. Engineered features were used as the first feature set and this was investigated with classifiers that have proven to work well with small feature sets such as Artificial Neural Network and Support Vector Classification. Subsequently Spectrograms, Wavelets and Timeseries were investigated with Convolutional Neural Network to extract key features from the large feature set. The UAR obtained from the three fold cross validation was used to compare the different classifiers. An optimal UAR of 90.7% was obtained with Artificial Neural Network.

The optimal results obtained in this project has exceeded the results obtained from extreme machine learning techniques. The optimal classification UAR obtained was 90.7% for classification consisting of 36 species whereas in extreme machine learning discussed in Previous works (Section 2.1) an UAR of 89.6% was obtained with 30 species [41]. Although the species used for this project may not be the same as the species used in the Extreme Machine Learning task. This project proved that a strong correlation does indeed exist between the bird sound and its Taxonomic rank and it has also shown that there is a potential of using machine learning techniques in the study of the biodiversity of an area. The results obtained for CNN do not accurately represent the true optimal results that could have been obtained. Optimising a CNN requires a lot of time and since time was a constraint in this project, it could very well be that the optimal CNN model reported in the previous chapter was not optimal. Also the convolutional layers tested for Spectrograms were two dimensional, whereas one dimensional convolutional layer may make more sense in case of Spectrogram, as the network should only convolve over the time axes.

One of the many ways of extending the work carried out here is by using additional metadata in the classification. Currently only the sound data is used but additional data such as the location and altitude data are available which would certainly aid in classification, since some birds will not be found in a certain area or at a certain altitude.

The initial aim of this project was to classify a large number of species (272) from sound. However due to the imbalanced dataset problem, a large number of species had to be discarded. Hence the number of species that were investigated at the end was only a mere 36 species. A real world classification rate would only be obtained if the algorithm was trained on a database consisting of a large number of species with adequate instances.

Although a range of different feature input were tested in this project, there are yet so many that could be tested. openSMILE alone has many different configuration files and can be adapted to extract engineered features that we deem necessary. There could be further research in the features that truly differentiate birds of different species and these could then be extracted and tested.

The project only had audio samples that mostly consisted of some noise and the sound of one individual bird. This however is not a true reflection of the audio samples that may be obtained if recordings were to be actually done in the wild. They may capture multiple bird sounds of different species simultaneously. Some further research could be done here by cutting the audio samples to a short duration and then classifying the individual fragmented audio samples or by applying multilabel classification. Also since species is a more specific classification than order or family, species classification can be used to classify birds into order or family for a multitask classification.

One slightly unrelated yet useful research could be to investigate if individual birds could be identified. This is known as bird banding, where an individual bird will be ringed and upon discovery in the future could provide useful information relating to the understanding of the species. It would prove to be very useful as current methods requires getting physical access to the bird so that it is equipped with a tag for future identification. However in countries such as Australia, laws exist to prevent banding certain species as it may cause damage and distress to the bird. Thus a method to identify birds without potentially harming it would definitely prove to be useful.

Chapter 8

Appendix

Classification Details
Species (Latin,German,English) (e,g, Accipiter gentilis)
Order (Latin,German„English) (e.g. Accipitriformes)
Family (Latin,German,English) (e.g. Accipitridae)

Audio Details
Name (e.g. AccGen00001)
Duration (e.g 2.703s)
Quality ((e.g b)
Bitrate (e.g. 24)
Sampling rate (e.g. 96000Hz)

MetaData
Location (e.g. Hom-Bad Meinberg, OT Belle)
NationalCode (e.g DE)
DegreeOfLatitude (e.g 51.9°)
DegreeOfLongitude (e.g 9.03°)
Date (e.g. 33719)
Equipment (e.g. Uher CR 210)

	Order	Family	Species
Kernel	RBF	RBF	RBF
Gamma	0.1	0.1	0.1
C	30	90	50
Accuracy	78.1	82.0	84.9
UAR	71.5	81.1	83.0
UAP	70.2	79.6	82.9
Unweighted Average F1 Value	70.8	80.3	82.9

Table 8.1: Optimal Parameters and results on Validation set for SVC

	Order	Family	Species
Kernel	RBF	RBF	RBF
Gamma	0.1	0.1	0.1
Nu	0.201	0.001	0.001
Accuracy	81.1	85.6	88.2
UAR	77.6	84.7	86.9
UAP	78.6	84.5	85.2
Unweighted Average F1 Value	78.1	84.6	86.1

Table 8.2: Optimal Parameters and results on Validation set for NuSVC

	Order	Family	Species
Solver	Eigen	LSQR	LSQR
shrinkage	0.201	0.401	0.001
n_components	30	140	105
Accuracy	68.1	69.7	82.3
UAR	68.0	69.4	81.2
UAP	67.6	68.2	81.6
Unweighted Average F1 Value	67.8	68.8	81.4

Table 8.3: Optimal Parameters and results on Validation set for LDA

	Order	Family	Species
Neighbours	1	5	1
Weights	distance	distance	uniform
Algorithm	kd_tree	kd_tree	kd_tree
LeafSize	25	20	35
Accuracy	72.8	75.9	79.6
UAR	67.2	73.6	75.1
UAP	67.0	72.6	75.5
Unweighted Average F1 Value	67.1	73.1	75.3

Table 8.4: Optimal Parameters and results on Test set for KNN

	Order	Family	Species
Layers	[384,192,80,8]	[384,192,15]	[384,192,36]
Learning Rate	0.01	0.03	0.02
Momentum	0	0	0
Accuracy	85.6	89.2	90.3
UAR	81.5	88.6	89.9
UAP	80.6	88.9	89.8
Unweighted Average F1 Value	81.0	88.7	89.8

Table 8.5: Optimal Parameters and results on Validation set for ANN

	Order	Family	Species
Network Architecture	input size 1x40000, conv1 filters 30 size 5, pool1_pool_size=(10), conv2 filters 20 size 10, pool2_pool_size=(10), hidden4_num_units=2000, hidden5_num_units=1000, output_num_units=8	input size 1x40000, conv1 filters 10 size 5, pool1_pool_size=(10), conv2 filters 10 size 20, pool2_pool_size=(10), hidden4_num_units=2000, hidden5_num_units=1000, output_num_units=15	input size 1x40000, conv1 filters 10 size 5, pool1_pool_size=(10), conv2 filters 10 size 20, pool2_pool_size=(10), hidden4_num_units=2500, hidden5_num_units=1000, output_num_units=36
Learning Rate	0.04	0.04	0.04
Accuracy	40.1	38.2	37.9
UAR	39.5	38.6	37.4
UAP	38.7	38.3	3 37.7
Unweighted Average F1 Value	39.1	38.5	37.6

Table 8.6: Optimal parameters and performance on Validation set for raw audio

	Order	Family	Species
Network Architecture	input size 1x125x125, conv1 filters 64 size 2x2, pool1_pool_size=(2,2), conv2 filters 64 size 1x1, pool2_pool_size=(2,2), conv3 filters 64 size 2x2, pool3_pool_size=(2,2), hidden4_num_units=3000, hidden5_num_units=1500, output_num_units=8	input size 1x125x125, conv1 filters 64 size 2x2, pool1_pool_size=(2,2), conv2 filters 64 size 1x1, pool2_pool_size=(2,2), conv3 filters 64 size 2x2, pool3_pool_size=(2,2), hidden4_num_units=3000, hidden5_num_units=1500, output_num_units=15	input size 1x125x125, conv1 filters 32 size 20x20, pool1_pool_size=(2,2), conv2 filters 32 size 10x10, pool2_pool_size=(2,2), hidden4_num_units=3000, hidden5_num_units=1500, output_num_units=36
Learning Rate	0.045	initially 0.0001 with adam (adaptive learning rate)	0.02
Accuracy	58.2	60.2	67.6
UAR	52.3	55.0	62.1
UAP	50.3	56.1	62.3
Unweighted Average F1 Value	51.3	55.6	62.2

Table 8.7: Optimal parameters and performance on Validation set for Spectrograms

	Order	Family	Species
Network Architecture	input size 1x20009, conv1 filters 10 size 50, pool1_pool_size=(5), conv2 filters 5 size 10, pool2_pool_size=(2), hidden4_num_units=5000, hidden5_num_units=1000, output_num_units=8	input size 1x20009 conv1 filters 10 size 100, pool1_pool_size=(5), conv2 filters 5 size 100, pool2_pool_size=(2), hidden4_num_units=5000, hidden5_num_units=1000, output_num_units=15	input size 1x20009 conv1 filters 100 size 10, pool1_pool_size=(1), conv2 filters 50 size 5, pool2_pool_size=(2), conv3 filters 20 size 3, pool3_pool_size=(3), conv4 filters 10 size 2, pool4_pool_size=(4), conv5 filters 5 size 2, pool5_pool_size=(5), hidden4_num_units=500, hidden5_num_units=1000, output_num_units=36
Learning Rate	0.02	0.02	0.01
Accuracy	43.5	39.5	35.2
UAR	38.1	36.2	33.1
UAP	37.3	36.8	32.6
Unweighted Average F1 Value	37.7	36.5	32.8

Table 8.8: Optimal parameters and performance on Validation set on Wavelets

Bibliography

[1] audEERING UG (haftungsbeschränkt) - openSMILE.
http://www.audeering.com/research/opensmile, 2016.
accessed 01-02-2016.

[2] Noramalina Abdullah.
Using labview and a support vector machine to classify normal and abnormal brain images.
http://sine.ni.com/cs/app/doc/p/id/cs-14132, 2016.
accessed 06-05-2016.

[3] Ethem Alpaydin.
Introduction to machine learning.
MIT press, 2014.

[4] P. Alstrom and R. Ranfft.
The use of sounds in avian systematics and the importance of bird sound archives.
BULLETIN-BRITISH ORNITHOLOGISTS CLUB, 123:114–135, 2003.

[5] Justin Johnson Andrej Karpathy.
CS231n Convolutional Neural Networks for Visual Recognition.
http://cs231n.github.io/, 2016.
accessed 06-05-2016.

[6] Miguel B. Araújo and Miska Luoto.
The importance of biotic interactions for modelling species distributions under climate change.
Global Ecology and Biogeography, 16(6):743–753, 2007.

[7] Chris Bagwell et al.
SoX - Sound eXchange.
http://sox.sourceforge.net/, 2016.
accessed 01-02-2016.

[8] Yong Joseph Bakos.
K-nearest neighbors.

`http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/knn.html`, 2016.
accessed 06-05-2016.

[9] Yoshua Bengio and Yves Grandvalet.
No unbiased estimator of the variance of k-fold cross-validation.
The Journal of Machine Learning Research, 5:1089–1105, 2004.

[10] C.J. Bibby.
Putting Biodiversity on the Map: Priority Areas for Global Conservation.
BirdLife International, 1992.

[11] birdwatching-bliss.com.
How do birds sing? Learn about bird song.
`http://www.birdwatching-bliss.com/bird-song.html`, 2016.
accessed 01-02-2016.

[12] Millie Bond.
Animal Classification.
`http://a-z-animals.com/reference/animal-classification/`, 2016.
accessed 01-02-2016.

[13] Michael Buckland and Fredric Gey.
The relationship between recall and precision.
Journal of the American society for information science, 45(1):12, 1994.

[14] CJ Cela-Conde.
The meaning of Hominidae.
Human evolution, 13(3-4):251–264, 1998.

[15] Nitesh V. Chawla.
Data mining for imbalanced datasets: An overview.
In *Data mining and knowledge discovery handbook*, pages 853–867. Springer, 2005.

[16] John B. Colby, Jeffrey D. Rudie, Jesse A. Brown, Pamela K. Douglas, Mark S. Cohen, and Zarrar Shehzad.
Frontiers in Systems Neuroscience.
`http://journal.frontiersin.org/article/10.3389/fnsys.2012.00059/full`, 2016.
accessed 06-05-2016.

[17] Ronan Collobert and Jason Weston.
A unified architecture for natural language processing: Deep neural networks with multitask learning.

In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[18] Jarrad A. Cousin.
The Speciation and Biogeography of Birds.
Pacific Conservation Biology, 10(3):202–203, 2004.

[19] Chris Drummond and Robert C Holte.
C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling.
In *Workshop on learning from imbalanced datasets II*, volume 11. Citeseer, 2003.

[20] Florian Eyben.
Real-time speech and music classification by large audio feature space extraction.
Springer, 2016.

[21] Florian Eyben, Felix Weninger, Stefano Squartini, and Björn Schuller.
Real-life voice activity detection with LSTM recurrent neural networks and an application to hollywood movies.
2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013.

[22] Florian Eyben, Martin Wöllmer, and Björn Schuller.
openSMILE: the munich versatile and fast open-source audio feature extractor.
In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM, 2010.

[23] Seppo Fagerlund.
Bird species recognition using support vector machines.
EURASIP Journal on Applied Signal Processing, 2007(1):64–64, 2007.

[24] W.T. Fitch and David Reby.
The descended larynx is not uniquely human.
Proceedings of the Royal Society of London: Biological Sciences, 268(1477):1669–1675, 2001.

[25] Jerome H Friedman.
On bias, variance, 0/1,Äîloss, and the curse-of-dimensionality.
Data mining and knowledge discovery, 1(1):55–77, 1997.

[26] Amar Gondaliya.
Regularization implementation in R.
`http://pingax.com/regularization-implementation-r/`, 2016.
accessed 06-05-2016.

[27] Kaizhu Huang, Haiqin Yang, Irwin King, and Michael R. Lyu.

Imbalanced learning with a biased minimax probability machine.
Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 36(4):913–923, 2006.

[28] Anil K. Jain, Jianchang Mao, and K.M. Mohiuddin.
Artificial neural networks: A tutorial.
Computer, (3):31–44, 1996.

[29] Pablo A. Jaskowiak and R.J.G.B. Campello.
Comparing correlation coefficients as dissimilarity measures for cancer classification in gene expression data.
In *Proceedings of the Brazilian Symposium on Bioinformatics*, pages 1–8, 2011.

[30] Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano.
Comparing boosting and bagging techniques with noisy and imbalanced data.
Systems, Man and Cybernetics, Part A: Systems and Humans, 41(3):552–568, 2011.

[31] C.V. Krishnaveni and T. Sobha Rani.
On the classification of imbalanced datasets.
IJCST, 2:145–148, 2011.

[32] Balaji Lakshminarayanan, Raviv Raich, and Xiaoli Fern.
A syllable-level probabilistic framework for bird species identification.
In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 53–59. IEEE, 2009.

[33] Mario Lasseck.
Large-scale identification of birds in audio recordings.
In *CLEF (Working Notes)*, pages 643–653, 2014.

[34] Jia Li.
Linear Discriminant Analysis.
`http://sites.stat.psu.edu/~jiali/course/stat597e/notes2/lda.pdf`, 2016.
accessed 01-04-2016.

[35] A. I. Marques, V. Garcia, and J. S. Sanchez.
On the suitability of resampling techniques for the class imbalance problem in credit scoring.
Journal of the Operational Research Society, 64(7):1060–1070, 2012.
accessed 01-02-2016.

[36] MathWorks.
Introduction to wavelet families.

http://mathworks.com/help/wavelet/gs/introduction-to-the-wavelet-families.html, 2016.
accessed 06-05-2016.

[37] Tom M Mitchell.
Machine Learning.
McGraw-Hill, 1997.

[38] Mohamad T. Musavi, Wahid Ahmed, Khue Hiang Chan, Kathleen B. Faris, and Donald M. Hummels.
On the training of radial basis function classifiers.
Neural networks, 5(4):595–603, 1992.

[39] Dennis O'Neil.
Classification of Living Things: Kingdom to Subphylum.
http://anthro.palomar.edu/animal/animal_3.htm, 2016.
accessed 01-02-2016.

[40] Maja Pantic.
Machine Learning Lecture - Instance Based Learning.
http://ibug.doc.ic.ac.uk/media/uploads/documents/courses/lect_slides_2012_2013/ML-lecture4.pdf, 2016.
accessed 01-04-2016.

[41] Kun Qian, Zixing Zhang, Fabien Ringeval, and Björn Schuller.
Bird Sounds Classification by Large Scale Acoustic Features and Extreme Learning Machine.

[42] RE Ricklefs and GW Cox.
Morphological similarity and ecological overlap among passerine birds on st. kitts, british west indies.
Oikos, pages 60–66, 1977.

[43] Björn Schuller and Anton Batliner.
Computational paralinguistics: emotion, affect and personality in speech and language processing.
John Wiley & Sons, 2013.

[44] Arja Selin, Jari Turunen, and Juha T Tanttu.
Wavelets in recognition of bird sounds.
EURASIP Journal on Applied Signal Processing, 2007(1):141–141, 2007.

[45] John Shawe-Taylor and Grigoris Karakoulas.
Optimizing classifiers for imbalanced training sets.
Advances in neural information processing systems, 11:253, 1999.

[46] soundingwell.
Reading and writing wave files.
`https://soundingwell.wordpress.com/`, 2016.
accessed 06-05-2016.

[47] Johan A.K. Suykens and Joos Vandewalle.
Least squares support vector machine classifiers.
Neural processing letters, 9(3):293–300, 1999.

[48] Theano Development Team.
Convolutional Neural Networks (LeNet).
`http://deeplearning.net/tutorial/lenet.html`, 2016.
accessed 06-05-2016.

[49] Kagan Tumer and Joydeep Ghosh.
Analysis of decision boundaries in linearly combined neural classifiers.
Pattern Recognition, 29(2):341–348, 1996.

[50] Dion Walker.
Statistical modelling services.
`http://www.statisticalconsultants.co.nz/services/statistical-modelling-services.html`, 2016.
accessed 06-05-2016.