Master's Programme in Data Science

Common rosefinch (*Carpodacus erythrinus*), P. Lauha 2020

Improving Template-Based Bird Sound Identification

Patrik Lauha

December 22, 2020

Supervisor: Professor Otso Ovaskainen

Examiner: Professor Jarno Vanhatalo

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Degree programme Master's Programme in Data Science

Tekijä — Författare — Author

Patrik Lauha

Työn nimi — Arbetets titel — Title

Improving Template-Based Bird Sound Identification

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year December 22, 2020	Sivumäärä — Sidantal — Number of pages 55

Tiivistelmä — Referat — Abstract

Automatic bird sound recognition has been studied by computer scientists since late 1990s. Various techniques have been exploited, but no general method, that could even nearly match the performance of a human expert, has been developed yet. In this thesis, the subject is approached by reviewing alternative methods for cross-correlation as a similarity measure between two signals in template-based bird sound recognition models. Template-specific binary classification models are fit with different methods and their performance is compared. The contemplated methods are template averaging and procession before applying cross-correlation, use of texture features as additional predictors, and feature extraction through transfer learning with convolutional neural networks. It is shown that the classification performance of template-specific models can be improved by template refinement and utilizing neural networks' ability to automatically extract relevant features from bird sound spectrograms.

ACM Computing Classification System (CCS):
Applied computing → Arts and humanities → Sound and music computing
Computing methodologies → Machine learning → Machine learning approaches → Neural networks

Avainsanat — Nyckelord — Keywords

automated species identification, bird song, spectrogram, convolutional neural networks

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Acknowledgements

Besides my official supervisor Otso Ovaskainen, who has supported me and granted me the opportunity to write my master's thesis about a topic that interests me both academically and personally, I want to thank several people, whose work has promoted and enabled my thesis. Panu Somervuo has guided me in selecting and applying the different methods considered in the thesis. Petteri Lehikoinen has shared his ornithological expertise and annotated majority of the data in addition to defining the letters from the original sound files together with Aki Aintila. The sound files have been collected by Ulisses Moliterno de Camargo, and Kerttu application, which was used to annotate the data, has been built by Meeri Rannisto. Finally, I have produced my master's thesis as a part of international LIFEPLAN [56] venture and this project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 856506).

Contents

1. Introduction

Birds are a common object of interest for both scientific community and numerous bird-watcher hobbyists around the world. Since many bird species are primarily detected by their vocalization, there is a great demand for a system that could identify birds automatically by their sound. Automatic identification of bird vocalizations would offer useful applications for public use, as well as a powerful tool for ecologists for monitoring the distribution and behavior of bird species. Reliable automatically produced identifications would enable data collection on an entirely different scale compared to methods that rely on identifications produced by human experts.

Automated bird song recognition has been studied for a couple of decades and significant process in the classification ability has been made. However, the challenge of automatically detecting bird species from continuous field recordings is still far from being completely solved. The earliest attempts towards automated bird sound classification were conducted in the late 1990s after the prominent development in the field of automated speech recognition [4, 31]. These applications utilized dynamic time warping and hidden Markov models to classify vocalizations of indigo bunting *(Passerina cyanea)* and zebra finch *(Taeniopygia guttata)* by comparing them to pre-selected templates. It was observed that these methods can achieve excellent performance for distinct song types under specific conditions but might not immediately extend to a greater number of species and wider range of sounds.

Other works have applied various kinds of methods such as backpropagation neural networks on time-frequency statistics [36], decision trees with support vector machine classifiers applied on mel-cepstrum and low-level signal parameters [12], sinusoidal modeling [21] and classifying singular vector representations of spectrograms [18]. Most methods can produce good or even excellent results for a certain subset of bird species. However, the more species are included, the more complicated the task gets, and it is extremely challenging to develop a solution that would generalize well to a wide range of species [43].

The process of recognizing bird vocalizations usually contains roughly following stages, even though the division between different steps might sometimes be rather unclear:

1. Preprocessing the data by converting raw field recordings into more easily processable form and performing some type of noise reduction.
2. Detecting and segmenting individual vocalizations.
3. Extracting important features from the representations of vocalizations.
4. Building classifiers, which operate on the selected set of features, using normally human-labelled annotations for training.

A common solution for audio preprocessing is converting sound files into spectrogram images with short-time Fourier transform (STFT), even though alternative options, such as wavelet transform, exist as well. For feature extraction and selection of classifier, there are numerous options. Mel-frequency cepstral coefficients (MFCC) are a popular choice as features, but also other descriptive statistics and autonomous feature extraction by artificial neural networks are used. The list of possible recognition methods is long as well: for example, hidden Markov models (HMM), support vector machines (SVM), decision trees, random forests, k-nearest neighbor -classifiers (k-NN) and neural networks have been used in various implementations.

A good overview of the progress in the field of automated bird song recognition can be obtained by observing the results of annual BirdCLEF competition [9]. The main task in the early years of the competition was to build a model, that can identify the most actively vocalizing birds from audio files that focus on a single foreground species [27]. The objective has subsequently been modified into a more challenging task of localizing and identifying all audible birds within the provided soundscape test set. Initially, the best results in BirdCLEF and earlier similar competitions [32] were provided by supervised classifiers built on matching probabilities and some low-level statistics of short sound segments [27, 28]. First convolutional neural networks (CNN) were introduced in the challenge in 2016 and have dominated the competition ever since [25, 26, 14, 47]. Already in 2017 all submitted solutions were based on CNNs [26].

The results from BirdCLEF competitions indicate, that it is much more difficult to recognize a variety of birds from real-life soundscapes, where different individuals and species vocalize from varying distances possibly overlapping with each other, than to classify single species from targeted records. The latter problem can be very efficiently solved by convolutional neural networks and is already considered as solved in the aftermath of BirdCLEF 2018 and 2019 [14, 47]. The former problem remains mainly unsolved, at least compared to the classification ability of reasonably qualified human experts [43].

Although neural networks have advanced the development towards automatic expert-level bird sound classification, one major challenge related to them is the need of data. Compared to many other machine learning methods, neural networks require huge amount of data [15], which is not always available for all species in a suitable form. There are some libraries of bird recordings such as Xeno-canto (circa 580 000 recordings of over 10 000 species) [58], Macaulay Library (circa 590 000 recordings of almost 10 000 species) [52], and Tierstimmenarchiv ("animal sound archive", circa 120 000 recordings of 1 800 bird species and other animals) [38]. However, many of the recordings are long and the targeted species are not vocalizing all the time. The recordings might also contain vocalizations of other birds in addition to the target species. Therefore, the data as such might not be directly suitable to be used as training data and could possibly require significant pre-processing effort by human experts.

There are currently few existing softwares for automatic bird sound recognition, such as BirdNET [51], Kaleidoscope Pro [57] and monitoR [16]. Neural network based BirdNET by the Cornell Lab of Ornithology and Chemnitz University of Technology is perhaps the most well-known among these, and it is also available as a mobile phone app for public use. In a small test with 205 recordings containing 23 different species, BirdNET was observed to reach accuracy of 91.5% [5]. Also according to my own experience, BirdNET classifies manually recorded and targeted single-species recordings fairly well in practice, as long as the vocalizations are loud enough for the microphone of a mobile phone. However, the practical performance of automatic recognition softwares can vary a lot depending e.g. on the number of species tested and the quality of the data [43].

One attempt to enable automatic bird sound classification is constructed by Ovaskainen, de Camargo and Somervuo [40]. Their Animal Sound Identifier (ASI) uses cross-correlation between spectrograms to build a set of species-specific models, which predict the presence or absence of different species in an audio sample. The key idea of ASI is that no previously existing sound templates for animal species are needed. Instead, the user of the software produces and classifies the training data from raw field recordings theirself with a moderate amount of work.

In this master's thesis I aim to improve the cross-correlation based classification ability of ASI by investigating alternative predictors and classification methods for bird sound. Since the field of signal processing and audio classification is extensive and there is a variety of possible approaches and applications, the purpose of this work is not to provide a comprehensive review of all possible solutions. Instead, the object is to select a few promising techniques, and based on them build practical models with as high performance as possible.

Improvement for current methods is searched in three different ways. The templates, on which the classification is originally based, are processed by extending them in varied ways, averaging over several samples and performing noise reduction. The texture features of bird sound spectrograms are studied and tested as additional predictors alongside cross-correlation. Finally, a fundamentally different deep learning technique, transfer learning with convolutional neural networks, is applied. After initial experiments, best performing models are compared with a new dataset. It is shown that template averaging and extensions as well as the employment of neural networks improve the performance of classification models.

The outline of this thesis is as follows. The operation of ASI and the data that are used in practical analysis are described in Chapter 2. Different methods for sound classification are introduced in Chapter 3 and the performance of these methods compared in Chapter 4. Finally, concluding observations and proposals for feature development are considered in Chapter 5.

2. Animal Sound Identifier

Animal Sound Identifier [40] is a MATLAB software designed for automatic identification of animal vocalizations. ASI provides a method for building species-specific models, which identify species automatically from field recordings. The process does not require an external database of pre-annotated training data. Instead, user provides ASI with a set of recordings and performs the identification of target species vocalizations themself in a semi-automated manner. User's task is to define a few model examples of bird vocalizations and classify training samples extracted from the recordings to positive or negative matches. Once the training is executed and species-specific models are fit and validated, ASI is ready to produce probabilistic classification of animal sounds from existing and new recordings. ASI pipeline consists of six steps, which are described below.

1. **Letter candidate identification**

   The identification process is based on defining a few representative, species-specific model examples of vocalizations, referred as "letters", for each target species, and comparing the similarity between these letters and new audio data. The first task is to locate these letters from the field recordings. To improve the efficiency of this process, ASI implements unsupervised search of potential letter candidates and locates from the audio segments short signals, which are repeated in the recordings and are therefore likely to be recognizable bird vocalizations. In the process of finding repeating patterns, cross-correlation (Equation 3.1) works as a measure of similarity.

2. **Letter selection and refinement**

   The final selection of letters is conducted by the user, who chooses the letters from the set of candidates suggested by ASI, by fine-tuning the starting and ending points of the letter along with the frequency band of interest, and identifies the vocalizing species. Alternatively, the letters can also be chosen freely from any part of the field recordings or imported from existing libraries. For optimal performance, multiple letters representing different types of vocalization should be defined for each species.

3. **Fitting letter-specific models**

Letter-specific models are fit to assess the probability of different letters occurring in an audio segment. Highest cross-correlations between letters and audio recordings are calculated for all letters and audio segments. For each letter, several audio samples with varying cross-correlations are played to the user, who classifies them according to whether the target species is present or not. The classified training data are used to fit probit regression models to predict the probability of a letter being present in an audio segment, using highest cross-correlation between the letter and the audio segment as predictor.

4. **Constructing species-level predictors by merging multiple letters**

   The letter-specific occurrence probabilities are combined to build species-level predictors for species-specific models. Raw predictors are first constructed directly from the letter-specific probabilities and then developed into more sophisticated predictors with modified principal component analysis.

5. **Fitting species-specific models**

   Species-specific probit models are fit to assess the probability of different species occurring in an audio segment. User can select the optimal explanatory variables from the set of predictors created in previous step.

6. **Validation of species-specific models**

   To evaluate the performance of ASI, species-specific models are validated by adopting a set of audio segments, which were not used in the training phase. These samples are submitted to the user, who identifies, which species are present in the segments. The classifications made by the user are then compared to those of ASI to evaluate the performance of the system in terms of recall and precision.

The focus of this master's thesis is in improving the performance in step 3. Even though the song of a certain bird species is highly stereotyped and often easily identified by an experienced observer, significant variation still exists in vocalizations of certain species and even individuals [42, 24, 46]. In the context of cross-correlation based similarity measurements, this variation forms a challenge. For example, the variation in the duration of silent fragments between the signals in two otherwise similar audio samples can dramatically affect cross-correlation, even though the samples might sound almost exactly similar to a human observer. As shown in Chapter 4, cross-correlation alone is often not sufficient predictor for species classification and more flexible models are required for better performance.

2.1 Data

The audio data used in this thesis were collected by Ulisses Moliterno de Camargo in ten different locations across region of Uusimaa in Southern Finland between May and August 2018. The whole dataset contains 1,810,194 one-minute .wav audio files recorded with autonomous recording units. The audio files were converted into normalized spectrograms, which can be considered as images with resolution of 5999 x 129 pixels or 129 x 5999 matrices with values between 0 and 1. One column of 129 pixels represents the amplitudes within frequencies 0-8 kHz during a period of ten milliseconds in the audio. By picking segments from these spectrograms, ornithologists Petteri Lehikoinen and Aki Aintila have defined letters that represent different vocalizations of several bird species. Figure 2.1 shows an example letter that represents a part of a song of tree pipit *(Anthus trivialis)*.

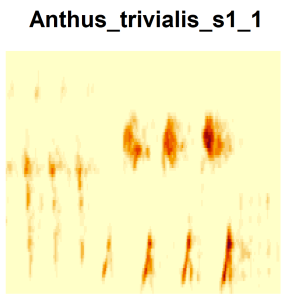


Figure 2.1: Example of a letter belonging to tree pipit *(Anthus trivialis)*.

The annotated data were produced in Kerttu online application [48] mostly by Petteri Lehikoinen and me. Kerttu is a citizen science platform with a purpose to collect human-annotations from ornithologists and amateur birders. The user selects, from the list of Finnish birds, all species that they can identify by sound. The letters, for which the samples are annotated, are then one by one randomly chosen among the letters belonging to those species. The audio samples to be annotated are selected by finding the best matches for the letter from all one-minute sound files and extracting samples with same size as the letter. From this set, 100 samples with cross-correlations (between the sample and the letter) ranging from 0 to 1 are annotated by the user. The selection of these 100 samples is weighted towards samples of such cross-correlation, that are equally likely to either contain or not contain the target species of the letter and thus provide the best information for the letter-specific model fitting. The user can classify the target species to be present or not present or declare that they are uncertain. In this thesis, uncertain samples were rejected. The data thus consists of short audio samples, each of which is related to a specific letter, and binary labels, which indicate, whether the sample contains the target species of the corresponding letter or not.

Initially, the data produced in Kerttu contained 6,469 samples for 60 letters of 37 different species, and later after complementary annotations 25,354 samples for 211 letters of 46 species. The letters with less than 10 positive samples actually containing the target species were excluded. A training set of letters was constructed from those data that were available at the end of September 2020 for experiments with the different classification methods. An additional test set was formed at the end of November 2020 from new letters, which had not belonged to the first set. The final data used in this thesis therefore contains two sets of letters and corresponding annotated samples. In the first phase alternative methods were explored using the initial set of 47 letters for 30 species and in total 4,765 annotated samples. In the second phase, the most promising models developed during the first phase were tested with a new set of 102 letters for 33 species and in total 10,200 annotated samples. A complete list of letters used in the analysis is provided in Appendix A.

3. Methods for improved bird sound identification

In this chapter, I introduce the functioning of current letter-specific models of ASI and the techniques that were tried while searching for better performing models. The analysis was conducted in R, MATLAB and Python with roughly following division. Data preprocessing and fitting of baseline models was conducted in MATLAB using Statistics and Machine Learning [55], Signal Processing [54] and Image Processing Toolboxes [53]. Model fitting and visualizations were performed in R using packages pROC [44] and tuneR [35]. Convolutional neural networks were built in Python using most importantly Keras library [8].

3.1 Baseline models

Currently in ASI, the similarity between audio signals is measured with *cross-correlation*, also known as *sliding dot product*. ASI uses MATLAB function normxcorr2 [3, 34, 20], which calculates the normalized cross-correlation between letter template and a spectrogram image. Normalized cross-correlation is defined as

$$\gamma(u,v) = \frac{\sum_{x,y}\left(f(x,y)-\bar{f}_{u,v}\right)\left(t(x-u,y-v)-\bar{t}\right)}{\sqrt{\sum_{x,y}\left(f(x,y)-\bar{f}_{u,v}\right)^2 \sum_{x,y}\left(t(x-u,y-v)-\bar{t}\right)^2}}, \tag{3.1}$$

where, in this case, f is the spectrogram image, t is the letter template positioned at location (u, v) and $\bar{f}_{u,v}$ is the mean of $f(x, y)$ in the region under the template. The correlation between a letter and a spectrogram image is calculated across all locations (u, v) of the spectrogram from the frequency band corresponding to that of the letter allowing a small variation. The highest correlation obtained is selected to represent the similarity between the letter and the sample.

The letter-specific models of ASI work as a baseline reference for the models of this thesis. As described in Chapter 2, the letter-specific models are simply probit regression models with one independent variable fitted separately for each letter. Probit model is a generalized linear model with the inverse of a standard normal cumulative distribution function as link function. In a probit model

$$P(Y = 1|X) = \Phi(X^T\beta), \tag{3.2}$$

where Φ is the cumulative distribution function of standard normal distribution, X is the vector of regressors, in this case the intercept and the cross-correlation, and β are the regression parameters. $P(Y = 1|X)$ is here the probability of the target species being present in an audio sample, given the highest cross-correlation between the letter and the sample.

3.2 Letter processing

Before investigating entirely new methods, an obvious question was, whether the letters can be improved to perform better with current techniques. During the data annotation, it was observed that some of the letters represent syllables that occur in vocalizations of many different species. For example, a short letter that contains only a rapid thrill might be difficult to identify even for a human expert without information about the following or preceding syllables. In addition, some letters seemed to be cropped too tightly and were missing part of the frequencies where the vocalization of the target species could be observed. Therefore, I experimented the effect of extending the letters and the corresponding samples. The initial trials included merely doubling the length of the letter and repeating similar model fitting process as was conducted with the baseline models. Letter extension had varying effects to the performance of the models. In some cases it was useful to use a longer period of sound as a letter and in some cases the performance declined dramatically, mostly because of including long fragments of silence in the letters or producing very long letters, for which it was hard to find matches. Therefore, a slightly more sophisticated approach was applied and a few different extension options in both x- and y-direction were tested. The choices that produced the best results were used to define new, optimally extended letters.

The optimal length for the letter was defined by setting the left side of the letter to begin from the initial starting point, 1/2 times the original length of the letter before the initial starting point and 1 original length of the letter before the initial starting point. Respectively, the right side of the letter was set to end at the initial ending point, 1/2 times the original length of the letter after the initial ending point and 1 original length of the letter after the initial ending point. By combining each starting point with each ending point, nine different croppings for one letter were produced, one of which corresponds to the original letter. Cross-correlations between the audio samples and these new letters were calculated, and the cropping that yielded the highest AUC (Equation 3.17) for the particular letter model was selected. Similar procedure was followed to obtain also the optimal height for the letter.

It is widely recognized, that even though the training data of automated recognition systems should contain also natural field noise [43], using high-quality training templates that are recorded from a close distance and contain as little background noise as possible, produce good results [30]. Therefore, another interesting experiment was to examine the effect of using letters that are constructed by averaging spectrograms over multiple samples instead of relying on a single sample extracted from one audio file alone. The hypothesis was that averaging over multiple samples eliminates background noise and other redundant signals as well as reshapes letter spectrograms towards a more general form, that represents the vocalization of the species in general instead of a single individual.

I tested two different approaches and formed both mean and median representations of each letter. Instead of using original letters, training set samples were used to calculate mean and median letters, and cross-correlations between these averaged letter representations and the samples were calculated. Mean letter was calculated simply by taking all positive instances of the training set and averaging every pixel of the letter over the whole set:

$$\bar{f}(x,y) = \frac{\sum_{i=1}^{n} f_i(x,y)}{n}, \tag{3.3}$$

where f_i, $i \in 1 \ldots n$ is the set of all n instances in the training set, where the target species is present, including the letter itself, and $f_i(x,y)$ is the intensity of pixel (x,y) in the spectrogram image of the ith instance in the training set. Median letter was formed in a similar manner, taking medians instead of means:

$$f_{\text{median}}(x,y) = \text{median}(f_i(x,y)) \qquad i \in 1 \ldots n. \tag{3.4}$$

Figure 3.1 demonstrates the effect of averaging on a letter belonging to Eurasian wren *(Troglodytes troglodytes)*. Averaging removes noise and smoothens the signal. Median letter preserves original texture better, whereas in mean letter the smoothing effect is stronger.

If the main target of letter averaging is noise reduction, there is also another alternative technique, which does not require many samples. While transforming the audio signal to image, *Gaussian blur*, also known as *Gaussian smoothing*, is applied to reduce detail and noise and enhance spectrogram structures. This is performed by convolving the spectrogram image with Gaussian function. The blurred image

$$f' = (f \circledast g), \tag{3.5}$$

where f is the original image and $g(x,y)$ is Gaussian function $\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$. In ASI, Gaussian blur is executed with MATLAB function imgaussfilt [2] with default value $\sigma = 1$

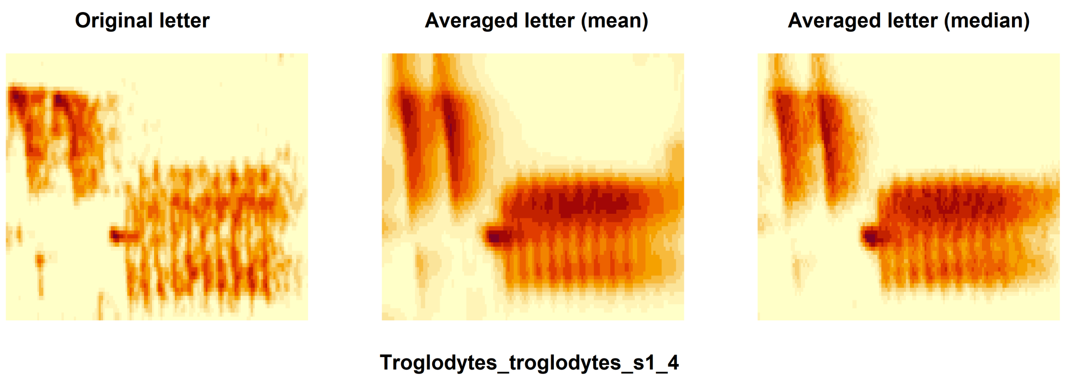


Figure 3.1: Example of averaged letters of Eurasian wren *(Troglodytes troglodytes)*. The figure shows from left to right the original letter, mean letter and median letter calculated from all samples that contain the target species.

as the standard deviation of 2-dimensional Gaussian smoothing kernel. As an alternative to letter averaging, all letters and audio samples were transformed into spectrograms with different values of σ and cross-correlations between letters and corresponding samples were re-calculated. Figure 3.2 demonstrates, how the parameter choice affects the spectrogram images. The greater the standard deviation, the stronger is the blurring.

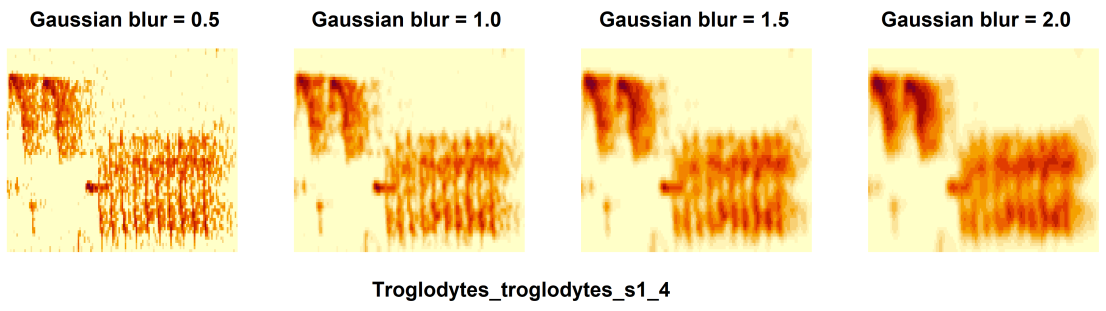


Figure 3.2: The effect of standard deviation parameter (σ) choice in Gaussian blur on a letter of Eurasian wren *(Troglodytes troglodytes)*. Default value in ASI is 1.0.

3.3 Texture features

In addition to mere letter processing, also additional predictors were studied. This was conducted by inspecting the performance of texture features of letter images as predictors. Different methods for measuring the texture and shape features of an image have been presented already since 1960s by eg. Haralick et al. [19], Tamura et al. [50] and Hu [23]. Hu's moment invariants are designed to produce measures of shape features, that are invariant to rotation, scaling, and translation. These were not considered relevant in the context of spectrograms, where for example a rotated curve does not represent a similar sound anymore.

Inspecting the data revealed, that even though the positive and negative samples were fairly easy to classify as sound files, in the spectrogram images there was no obvious difference between the two classes visible to the human eye. Figure 3.3 shows an example case of positive and negative samples for a letter belonging to common crane *(Grus grus)*. Since the textural features presented by Tamura et al. are designed to correspond to human visual perception, whereas those presented by Haralick et al. provide more abstract measures of texture, the latter were chosen to be applied in experiments.

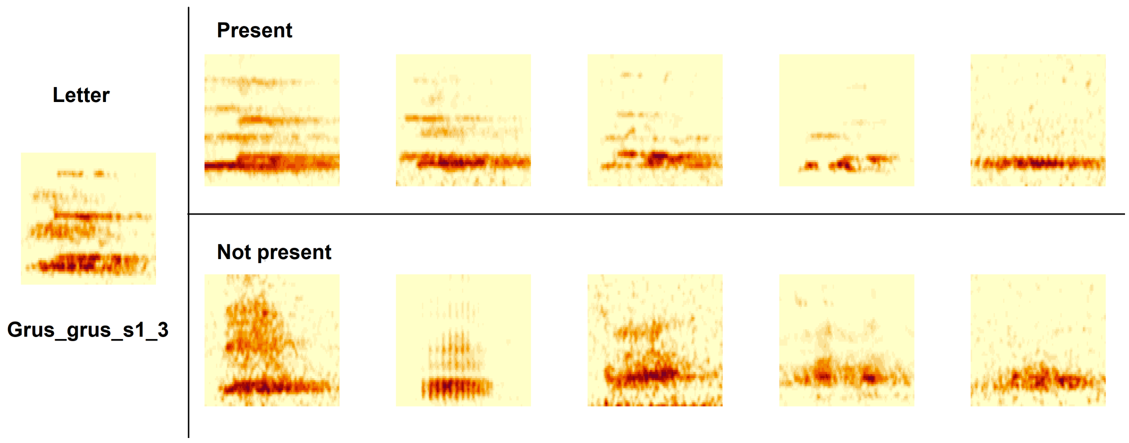


Figure 3.3: Examples of positive and negative spectrogram samples for a letter of common crane, *Grus grus*). The letter is shown on the left panel. Top row shows samples that contain vocalizations of common crane and bottom row shows samples that contain some other signals or noise.

The textural features of Haralick et al. are based on *gray-level co-occurrence matrix* (GLCM), also known as *gray-tone spatial dependency matrix*. The GLCM of an image is formed by quantizing the pixels of the image into N_g distinct gray-level classes and calculating, how often these different gray-levels occur in the neighborhoods of each other. GLCM P is a N_g x N_g matrix, where each cell indicates, how many times pixels of gray-level j occur in some fixed neighborhood of all pixels of gray-level i:

$$P(i,j|\alpha) = \#\text{ pixels taking value } j \text{ in the neighborhoods } \alpha \text{ of pixels taking value } i. \tag{3.6}$$

The 14 features calculated from the GLCM are *angular second moment, contrast, correlation, variance, inverse difference moment, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, information measures of correlation I & II* and *maximal correlation coefficient*. Equations for these features are given in Appendix B. GLCMs of the letters and sample images were calculated with MATLAB function graycomatrix [1, 19] with two different parameterizations for neighborhood α and number of gray levels N_g. In addition to the default setup $N_g = 8$, α : *next pixel on the* *same row*, increased neighborhood and number of gray-levels $N_g = 10$, α : *all pixels that are at most 2 rows and 2 columns away from target pixel*, were applied. Since modifying these parameters had only minor effect to the results, only the results produced with the default setup are presented in Section 4.3. The textural features were calculated from the GLCMs with MATLAB function haralickTextureFeatures [37].

All 14 texture features were calculated for each letter and the corresponding samples, after which the distance between the sample and the letter was calculated for all features. In addition to these distances, an overall Euclidian distance based on distances of all other features was calculated. All distances were first used as only predictor of the letter models separately. Two features, contrast and maximal correlation coefficient, which seemed to perform relatively well compared to other texture features, were also used as additional predictors in the models together with cross-correlation.

3.4 Transfer learning with convolutional neural networks

Since convolutional neural networks are currently the most commonly used and best performing systems in the field of automatic bird sound identification, it was natural to try to apply them for letter-specific classification, and study how they perform compared to cross-correlation. Instead of manually searching for useful features, neural networks can automatically detect them directly from annotated spectrogram images.

A neural network consists of several *layers*, which contain a number of *hidden units* (also known as *neurons*), that apply transformations on the data to produce predictions. The predictions are compared to true target values with a *loss function* to produce *loss score*, which is passed on to the *optimizer*, which updates the weights that affect the operation of network layers. Updated weight parameters are obtained by an iterative method called *stochastic gradient descent*, where weight parameters are updated by moving towards the negative gradient of the loss function:

$$w_{t+1} = w_t - \eta \nabla L\Big(y, g(x, w_t)\Big), \tag{3.7}$$

where w_t are the weights at timepoint t, $L\Big(y, g(x, w_t)\Big)$ is the loss function for given data x, correct output y and weights w_t, and η is a parameter that controls the step size in the updates, also known as *learning rate.* The process is repeated until the weights are adjusted so, that they produce correct predictions preferably for both training data and validation data, which has not been exploited for weight updates.

The different layer types utilized in the models of this thesis are introduced below following the descriptions of Keras documentation [8] and François Chollet's book *Deep learning with Python* [7].

- **Dense layer**, also known as *fully connected layer* creates a weight matrix called kernel, whose size is determined by the dimension of the input and the number of neurons in the layer. The dense layer projects the input data onto a lower dimensional space by applying a (non-linear) piecewise activation function to the dot product of the input data and the kernel:

  $$f(X \bullet W + b), \tag{3.8}$$

  where f is the activation function, X is the input feature map (data) received from the previous layer, W is the kernel weight matrix and b is a bias vector. Since dense layer is merely a transformed matrix dot product between the input and the kernel of the layer, all neurons of a dense layer interact with all neurons of previous layer, which allows the dense layer to learn global patterns from the input image or feature map.

- **Convolutional layer** creates convolution kernel, which is convolved with the layer input to produce the layer output. The convolution operation extracts small patches from the input image or feature map and applies the same transformation to all these patches, producing an output feature map, which is a representation of the features of the input. The neurons of convolutional layer are only connected to a smaller subset of elements in the previous layer. This process allows convolution layers to learn local patterns from the input image.

- **Pooling layer** combines the outputs of a cluster of neurons in one layer into a single neuron in the next layer and therefore reduces the dimension of the data. The pooling is applied by taking the maximum value over a window of a few neurons and passing it on to the next layer.

- **Flatten layer** flattens the multidimensional inputs into vectors that can be passed on to a dense layer.

- **Dropout layer** randomly sets a given fraction of all input values to 0 in the training phase to reduce overfitting.

The activation functions used by convolutional and dense layers were in my networks rectified linear unit (ReLU)

$$f(x) = \max(0, x) \tag{3.9}$$

and sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{3.10}$$

which are very common choices for activation functions in neural networks. ReLU activation was used in the *hidden layers* in the middle of the network and sigmoid activation in the output layer to produce the final classification.

Since training a huge number of parameters requires a lot of data, a common technique used also in bird sound identification is *transfer learning* [33, 60]. The term transfer learning refers to transferring some form of knowledge obtained in a certain task, such as low-dimensional representations of images, to be utilized in another partially similar, yet different problem [41]. In the context of convolutional neural networks, it is possible to load an existing network with pretrained model weights and exploit it for building a new network. Several models, such as VGG16 [45], InceptionV3 [49] and ResNet152V2 [22] are available in Keras. These models have been trained on images from ImageNet database [11], which contains more than 14 million annotated images.

By loading a pretrained model and removing the final classification layer, it is possible to extract a set of interesting features from the input images. Instead of a final prediction, the output of the pretrained model for a given input image is now a set of features, on which the classification produced in the final layer was initially based. A new classifier is now built on top of the pretrained model and trained with new data. The idea behind this technique is, that the pretrained network has initially learned, how to extract general features, which are useful in almost any kind of visual classification [7]. The pretrained convolutional base extracts these features from the new data and passes them on to the new classifier, which is trained to use them to produce predictions for the new data. Once the extraction of generally useful features has already been learned with a comprehensive dataset, particularly important features for the new task can be learned more efficiently from the smaller dataset.

Since the number of training instances for each letter was limited, I utilized transfer learning with pretrained networks to build the letter-specific models. Networks were trained separately for each letter with cross-validation using both positive and negative samples of a letter and the letter itself as training data. The initial trials were conducted following the textbook example of Chollet [7] for using pretrained VGG16 network for feature extraction. The results from first these trials (which are not shown in this paper) were promising, even though there was notable variation in the performance between different letters. Due to small number of training data, the results were somewhat arbitrary, and some letter models were overfitting heavily. However, it seemed evident, that with neural networks improvement was achieved for some letters, for which the previous methods had not worked well. More stable and better performing models were built by tuning the network structure and optimizing network hyperparameters based on trials on few selected letters. The final solution contains two different network structures, one for large and one for small letters. Both are shown in Figure 3.4.

The first network structure was applied for letters with both width and height of at least 32 pixels. This is the minimum input size of images in the VGG16 network. VGG16 contains two input layers, six pooling layers and 13 convolutional layers with pretrained weights and it was used as the convolutional base in the network structure for large letters. The weights of this convolutional base were frozen to preserve the pretrained information from ImageNet data except for the last block of three convolutional layers and two pooling layers. The weights of these layers were trained with the bird sound data to fine-tune the feature representations obtained from VGG16 to be especially suitable for the bird sound classification task. A classifier that consists of two dense layers, a flatten layer and a dropout layer was added on top of the VGG16 to produce binary classifications from the feature representations. The size of the training data was artificially increased with data augmentation in order to prevent overfitting, which tends to be a problem in neural networks with too little training data [15]. The augmentation was executed by creating 4 replicas of each training set image and adding Gaussian noise ($\mu = 0, \sigma = 0.01$) to them. A larger training set size and increased diversity of training data was thus achieved via manipulated samples that mimic vocalizations recorded in noisy environments.

In the trials, it turned out, that transfer learning is not necessary for letters smaller than 32x32 pixels, probably due to smaller number of parameters to be learned. For these letters, I built another network without a pretrained convolutional base, mainly following the examples of Chollet [7]. The network contains an input layer, three convolutional layers, two dense layers, a flatten layer and a dropout layer. The first pooling layer is only applied for letters, with both sides greater than 17 pixels. Since there was no pretrained weight matrix available, all weights of the network were trained during the training phase. Again, intensive data augmentation was applied to avoid overfitting and the size of the training set was increased by a factor of 10 in a similar manner as described above.

All models were trained with a Root Mean Square Propagation (RMSprop) optimizer using binary cross-entropy as a loss function to be minimized. For a single observation and prediction, binary cross-entropy is defined as

$$L(y, \hat{y}) = y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}), \tag{3.11}$$

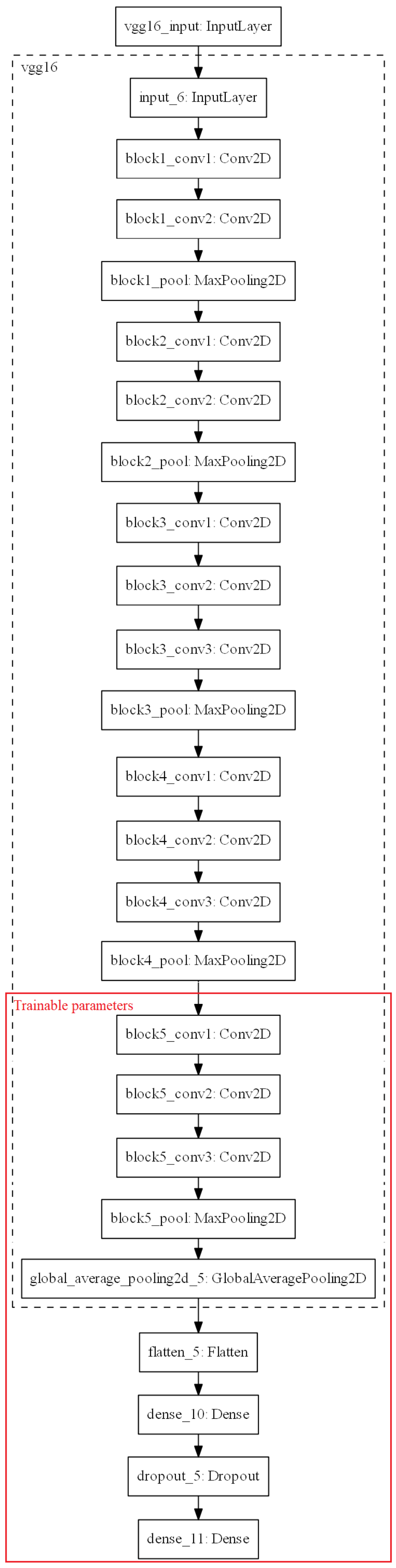
where y is the correct binary label and $\hat{y}$ is the predicted probability $P(Y = 1)$. In RMSprop, the learning rate is adapted separately for the weight parameters by dividing the learning rate by a running average of the magnitudes of previous gradients for the same weight. The running average is first calculated as

$$v(w_t) = \rho v(w_{t-1}) + (1 - \rho)\Big(\nabla L\big(y, g(x, w_t)\big)\Big)^2, \tag{3.12}$$

where ρ is the *discounting factor* and other terms are as in Equation 3.7. The weight parameters are then updated as follows:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v(w_t)}} \nabla L\Big(y, g(x, w_t)\Big). \tag{3.13}$$

Learning rate was set to $\eta = 0.00002$ in the models for large letters and $\eta = 0.0001$ in the models for small letters, and discounting factor was set to Keras default value $\rho = 0.9$. Different optimizers were not explored, since RMSprop is presented as a good enough choice for almost any task [7]. The number of epochs to be run in the training phase was set to 10, since this produced smallest losses in the trials with a few letters. An epoch is an iteration over all training set instances, during which weights are updated and loss function evaluated possibly multiple times. With too few epochs, the values of the weight matrix do not have enough time to converge, but with too many epochs the network starts to overfit. Other adjustments, such as using a different convolutional base instead of VGG16 or more complex data augmentation with existing methods of Keras, were also tested. These were not included in the final models, since they did not seem to enhance the performance of the models.



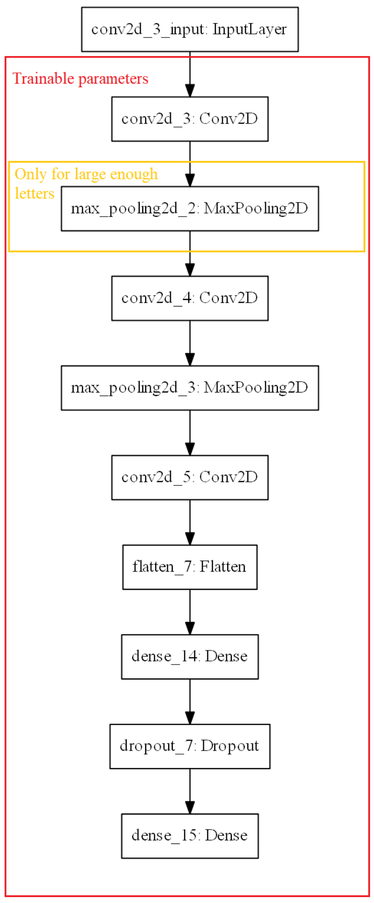


Figure 3.4: Structures of final letter-specific convolutional neural networks for large and small letters. Red squares indicate the parts of the network that were trained with the bird sound data. The model structure on the left side was used for letters with size of at least 32x32 pixels and the model structure on the right side for letters smaller than that.

3.5 Methods for testing model performance

In this section I introduce the methods that were used in model comparisons. Different methods were first experimented with the initial dataset (results shown in Sections 4.1 - 4.4) and the most promising methods were selected for more careful comparison (Section 4.5). This final comparison was executed with a new dataset and complementary measures of performance to evaluate the performance of improved letter models reliably and more extensively.

As a measure of performance for all models, I used the *area under the ROC curve* (AUC) [17]. In binary classification, the classification performance can be measured with *true positive rate* (TPR), also known as *sensitivity* or *recall*, and *false positive rate* (FPR), also known as *fall-out*. TPR indicates among all positive samples, the share of those instances, that are correctly classified as positive:

$$TPR = \frac{TP}{TP + FN}, \tag{3.14}$$

where TP is the number of positive samples classified correctly as positive and FN is the number of positive samples classified incorrectly as negative. Correspondingly, FPR indicates among all negative samples, the share of those instances, that are incorrectly classified as positive:

$$FPR = \frac{FP}{FP + TN}, \tag{3.15}$$

where FP is the number of negative samples classified incorrectly as positive and TN is the number of negative samples classified correctly as negative. If all instances are classified correctly, $TPR = 1$ and $FPR = 0$.

Let $TPR(t)$ and $FPR(t)$ denote TPR and FPR, when an instance is classified as positive, if its prediction score (model's prediction for the probability of the instance being positive) is greater than t. When $TPR(t)$ and $FPR(t)$ are plotted against each other with different values of t, we obtain a *receiver operating characteristic curve* (ROC curve). The area under the ROC curve summarizes model's capability to separate the two classes from each other and is defined as

$$AUC = \int_0^1 TPR(FPR^{-1}(x))\ dx. \tag{3.16}$$

A more reasonable probabilistic interpretation for AUC can be deduced as follows. Let X_1 and X_0, with respective probability densities f_1 and f_0 and cumulative distribution functions F_1 and F_0, be random variables, that represent model output probabilities for a positive and a negative instance drawn from the data. Now, for a given threshold t,

$FPR(t) = 1 - F_0(t)$ and $TPR(t) = 1 - F_1(t)$, and we can write AUC as

$$\begin{aligned} AUC &= \int_0^1 TPR(FPR^{-1}(x))\ dx \\ &= \int_{FPR^{-1}(0)}^{FPR^{-1}(1)} TPR(t)FPR'(t)\ dt \\ &= \int_{\infty}^{-\infty} TPR(t)(-f_0(t))\ dt \\ &= \int_{-\infty}^{\infty} TPR(t)f_0(t)\ dt \\ &= \int_{-\infty}^{\infty}\int_{t^*}^{\infty} f_1(t^*)\ dt^*\ f_0(t)\ dt \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbf{1}_{t^*>t} f_1(t^*)f_0(t)\ dt^*\ dt \\ &= P(X_1 > X_0). \end{aligned} \tag{3.17}$$

In other words, given a random negative sample and a random positive sample, AUC can be interpreted as the probability of the positive sample having higher prediction value (output probability from the model) than the negative sample. Figure 3.5 shows an example of a ROC curve with AUC 0.880. If the predictions for positive and negative samples were linearly separable, and the model could classify all samples correctly, AUC would be 1 and the area under the ROC-curve would cover the whole plot. In a random classifier, the true positive rate would increase linearly with the false positive rate and AUC would be 0.5.

All models described in previous sections were fit with 4-fold cross-validation for each letter separately, except the neural networks, which were evaluated with 5-fold cross-validation. The number of partitions was chosen as a compromise between having as large training datasets as possible and tolerable run times for the model fitting codes. With neural networks the training set sizes were slightly increased by using 5 folds.

For each letter, the samples corresponding to the letter were split to partitions. Each of these partitions were used in turn as a validation set for calculating the AUC for the model fitted with the remaining three (four) partitions. The final AUC for a specific letter model was then calculated as mean of these four (five) scores. The samples of the validation sets were not used in model construction in any way. For example, in letter averaging, the averaged letter was always formed using only samples of the training set.

In letter extension, various models were fit and the final extension strategy for each letter was chosen by applying the extension, which yielded to highest AUC. When several equally good models are compared in a setup, where randomness is involved in the evaluation, some of the models would most likely still perform better than others just by chance [6]. With different data or different cross-validation partitions, the results might be different. Selecting the model with the highest AUC from a set of possible models might therefore have yielded to an overly optimistic confidence about the performance of

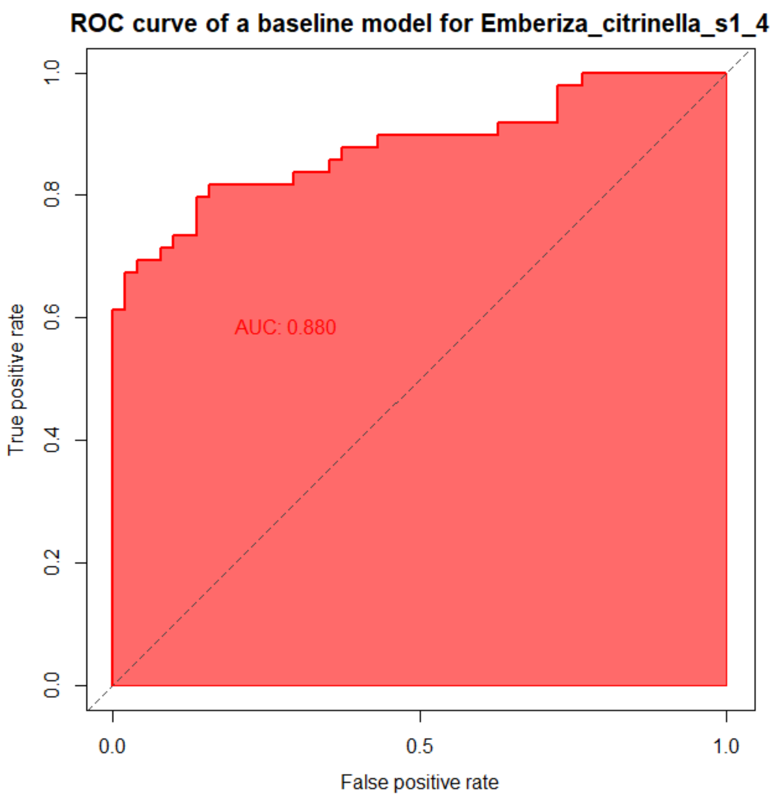


Figure 3.5: Example of a ROC curve of a baseline model for a letter of yellowhammer (*Emberiza citrinella*). Dashed line indicates the performance of a random classifier.

the model. This phenomenon is known as *model selection bias*. However, the results of letter extension can still be used to reflect, how well very carefully defined letters might perform in optimal scenario. The risk of model selection bias is also reduced by the fact that the model space was quite limited: only nine different options for x- axis cropping plus eight new options for y-axis cropping. In addition, the cross-validation codes were run twice, so that the first run was only performed to define the optimal cropping for each letter. After the croppings for each letter were fixed based on the results of the first run, the letter models were fit and evaluated again with different partitions to cross-validation folds. This was done to ensure obtaining a realistic image about the model performance with the selected cropping and not just reporting the same AUC, based on which the cropping was selected in the first place.

While constructing the models described in previous sections, the hyperparameters of the models (such as parameters of GLCM construction and neural network hyperparameters) were selected to produce as high AUCs as possible for those letters that were currently available. This means, that the model structures were optimized to perform well with the first set of letters, and it was therefore necessary to employ a test dataset to obtain an unbiased picture of the performance of different methods. A test dataset of 102 new letters was used to compare best performing methods with each other with previously unseen data. The last trials were conducted between two methods: neural networks and cross-correlation for optimized letters. Optimized letters were built using (mean) letter averaging and executing x- and y-axis extensions described in Section 3.2. Texture features were not included in these final trials, since they did not provide any improvement to probit models with averaged letters. Convolutional neural networks were constructed as explained in Section 3.4.

Letter optimization and neural networks were also combined in two different ways by:

1. Applying neural networks on the extended samples obtained from letter optimization. These were produced together with the extended letters, when in addition to the letter itself, also all corresponding samples were extended identically in both horizontal and vertical directions in order to make the cross-correlation comparison between letters and samples meaningful.

2. Fitting a probit regression model with two regressors: cross-correlation with optimized letters and a predictor value obtained from the neural networks. The neural network-based predictors were formed by passing the CNNs' predictions through the inverse of cumulative distribution function of standard normal distribution Φ^{-1}, that is, the link function of probit regression.

Thus, in total four different methods were compared in the final experiments.

All models were now evaluated with 5-fold cross-validation with same partition to folds for every method. This way the results from different methods are more comparable with each other. For each letter, the data were divided into five folds, four of which were used to build the models, which then produced predictions for the last fold. By repeating the procedure five times, predictions for the probability of the sample containing the target species were obtained for every instance in the dataset. The letter-specific AUCs, as well as all other metrics presented in this section, were then calculated based on these predictions. To avoid model selection bias, the letter extension codes for letter optimization were again run two times with different partitions to cross-validation folds, the latter of these being the same partition which was used with other methods.

All previous evaluations were conducted using AUC as the only performance metric. AUC is a useful tool for measuring model's discrimination ability between positive and negative class, but it does not directly indicate anything about the accuracy of the model predictions. Therefore, *classification accuracy* was included as an additional measure of performance for the test set letter models. The threshold was set to 0.5 and all samples with greater predicted probability were classified as positive and all samples with smaller predicted probability as negative. Accuracy was then calculated for each letter model as the proportion of correctly classified samples out of all samples:

$$\text{accuracy} = \frac{\#\ \text{correctly classified samples}}{\#\ \text{all samples}}. \tag{3.18}$$

The probability predictions of the method that yielded highest AUCs and accuracies were further inspected by evaluating the *sharpness* and *calibration* of the predictions. Sharpness refers to the concentration of a probability distribution, which is used to predict a certain outcome and calibration to the consistency between the prediction and the observed outcome. The predictions are ideal, when they are both well-calibrated and sharp [13].

In binary setup, the predicted probability distribution for a single data instance is concentrated and sharp, when the predicted probability for a given sample is either very high or very low. This means, that the model is highly certain about the true class of the sample. In this thesis, the sharpness of letter model predictions was assessed visually by plotting the distributions of predictions for positive and negative samples of each letter separately. Optimally, the distributions of predictions for positive samples should be concentrated close to 1 and for negative samples respectively close to 0. Narrow distributions close to zero and one would mean that for most data instances the predictions are sharp, and when encountering a positive or a negative sample, the model is usually able to classify it correctly with high certainty.

In a probabilistic classifier, the empirical probability of a data instance x belonging to class c, given the probability prediction $p(x) = p$ produced by the classifier, can be calculated as the proportion of instances belonging to c out of all instances with prediction p:

$$P(c|p(x) = p) = \frac{\#\ \text{samples with prediction } p \text{ belonging to class } c}{\#\ \text{samples with prediction } p}. \tag{3.19}$$

A classifier is called well-calibrated, if

$$P(c|p(x) = p) \to p, \qquad \text{as number of data instances } n \to \infty. \tag{3.20}$$

In other words, among samples that are assigned to be positive with probability p, the proportion of samples that actually are positive should in the long run approach p [10, 59].

In addition to the sharpness assessment, reviewing the calibration of probabilistic predictions provides useful information about the quality of predictions. Even if the model predictions are uncertain, they can still be useful, if the reported probabilities are well-calibrated. On the other hand, reporting sharp but poorly calibrated predictions is harmful, if the model predictions are interpreted as the probability of samples belonging to certain classes. For example, when employing the letter-specific predictions in species-specific models, the performance of species-specific models would probably be damaged if letter-specific prediction reported, that the probability of the target species being present in an audio segment (based on a certain letter) is 95%, when it was in reality just 75%.

The calibration of the letter model predictions with the best performing method was evaluated with *binning*. All samples corresponding to different letters were split to ten classes according to the assigned predictions. The proportion of positive samples, that contain the target species, was calculated for each class. If the model predictions were reliable, the proportion of positive samples should correspond to the predicted probabilities in each class. For example, in the class of predictions ranging from 0.0 to 0.1, the proportion of positive samples should be between 0% and 10% and optimally close to 5%.

4. Results

In this chapter, I introduce the results of different methods studied in this master's thesis. Sections 4.1 - 4.4 show the results of method inspection with the first set of letters and Section 4.5 shows the final comparison between the best methods.

4.1 Performance of baseline models

For some letters, cross-correlation based baseline models perform fairly well, while for some others they barely beat random guessing. The highest AUCs were obtained for letters of blue tit (letter: Cyanistes_caeruleus_c2_2, AUC: 0.972), common chaffinch (Fringilla_coelebs_c1_2 : 0.960), common redstart (Phoenicurus_phoenicurus_s1_1 : 0.958) and Eurasian wren (Troglodytes_troglodytes_s1_6 : 0.952). Lowest AUCs belong to barnacle goose (Branta_leucopsis_c1_1 : 0.592) and Eurasian robin (Erithacus_rubecula_c2_2 : 0.636, Erithacus_rubecula_s1_4 : 0.653). The mean of the AUCs of 47 letter models is 0.848 and median 0.860. In following sections these letter-specific AUCs are presented as a reference point for new models.

4.2 The effect of letter processing

All results presented in this section were obtained with similar probit regression models as the baseline models. Only the letters against which the cross-correlations were calculated had been modified in the process. First, I present the results from letter extensions.

Cropping	Min.	Q_1 (25%)	Median	Mean	Q_3 (75%)	Max.
x- and y-axis optimized	0.569	**0.872**	**0.904**	**0.895**	0.945	**0.995**
x- axis optimized	0.534	0.854	0.901	0.884	**0.948**	**0.995**
Original	**0.592**	0.817	0.860	0.848	0.907	0.972

Table 4.1: Summary statistics of the distributions of letter model AUCs with different letter extension strategies. AUCs of six letter models are shown for each method. These include the worst letter model, the best letter model, median, mean and lower and upper quartiles.

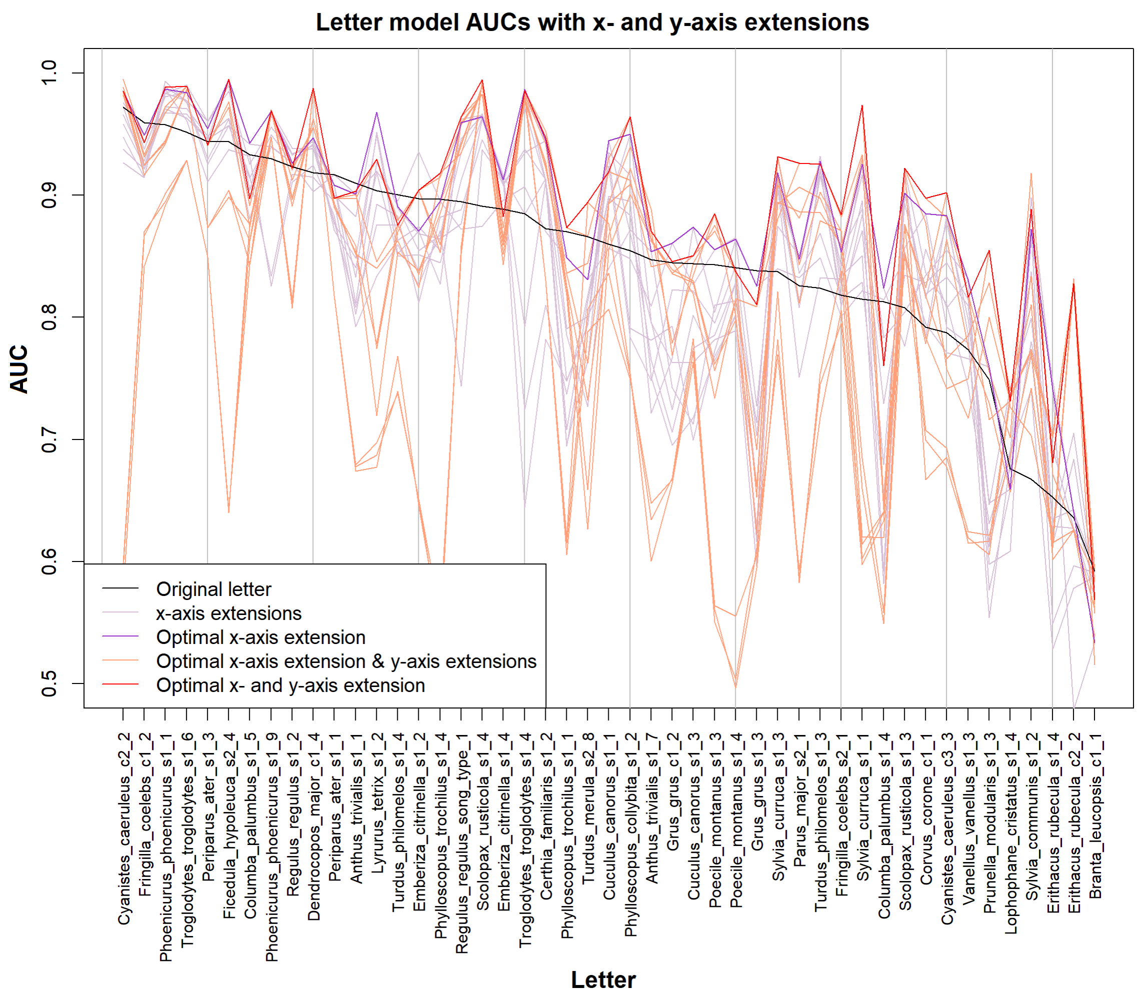


Figure 4.1: AUCs of letter-specific models for differently cropped letters. The figure contains results from nine different choices for x-axis extension and nine different choices for y-axis extension.

Table 4.1 shows summary statistics of the distributions of AUCs of letter-specific models for different methods. Figure 4.1 shows, how letter extension affects the performance of letter-specific models. Since the x-axis-optimized letter were selected by picking the cropping with the highest AUC of all alternative x-axis croppings and the x- and y-optimized letters were selected by picking the cropping with the highest AUC of all alternative y-axis-croppings with the selected x-axis-cropping, the AUCs of x- and y-optimized letters should be higher than any AUCs received from alternative extensions. However, because of random fluctuation originating from the cross-validation and fitting the models two times, this is not always the case.

Overall, the results show that the cropping of the letter plays an important role in terms of model performance. Good letters should be long enough to contain enough information about the vocalizing species, but not too long to prevent failing due to variation in the vocalizations. Based on these trials, it can be assumed that it is important to define both short and long letters for each species to obtain good performance for species-specific models, which are later constructed by combining the information from letter-specific models.

The most promising letter processing technique turned out to be letter averaging. Table 4.2 and Figure 4.2 demonstrate the effect of letter averaging to the performance of the models. For most letters, the AUCs of average letter models are higher than those of baseline models.

Letters	Min.	Q_1 (25%)	Median	Mean	Q_3 (75%)	Max.
Mean	0.568	**0.826**	**0.900**	**0.881**	0.944	0.982
Median	0.537	0.824	0.879	0.878	**0.952**	**0.993**
Original	**0.592**	0.817	0.860	0.848	0.907	0.972

Table 4.2: Summary statistics of the distributions of letter model AUCs with and without letter averaging.

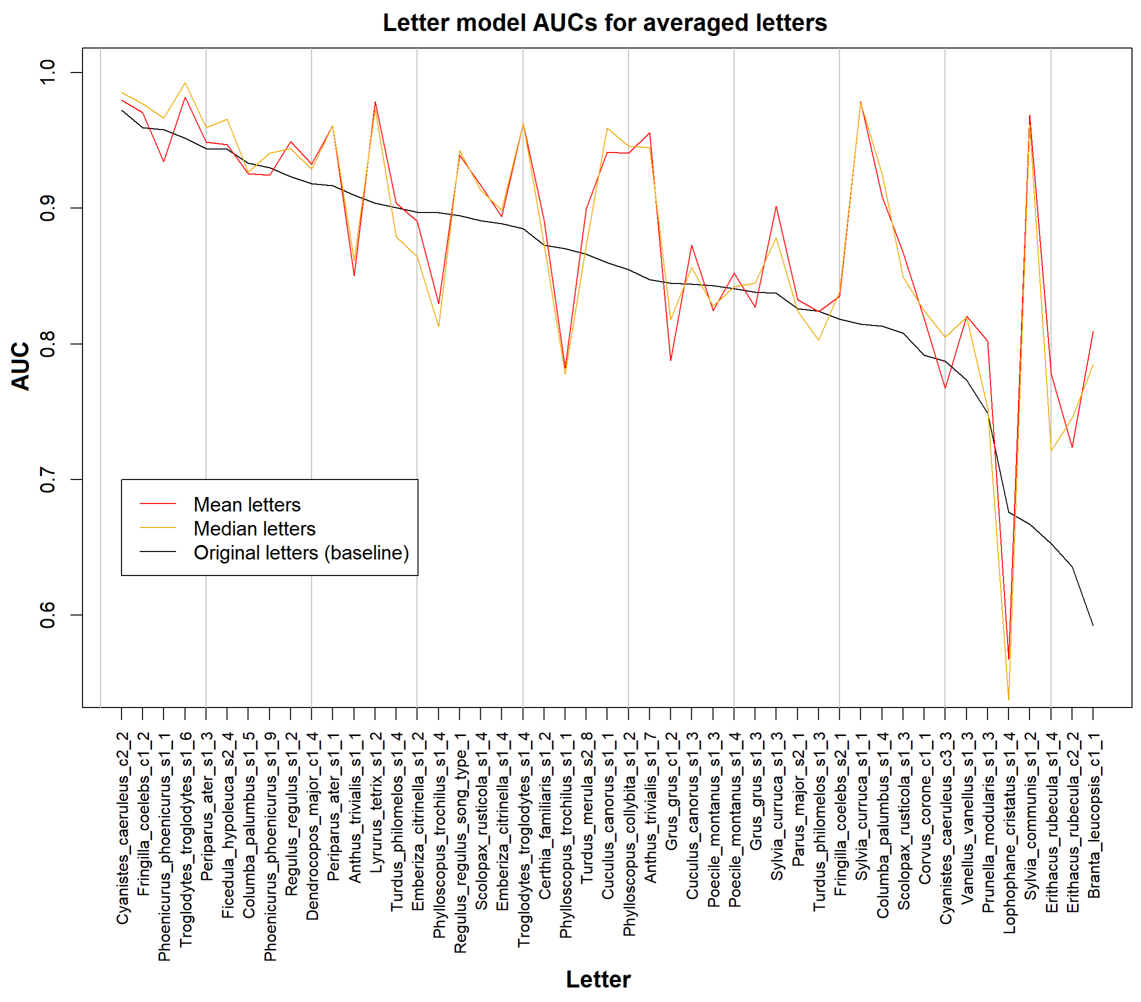


Figure 4.2: AUCs of letter-specific models for original and averaged letters.

The initial hypothesis was, that median letters would perform better than mean letters, since using medians would effectively remove noise while still preserve the texture and contrast of the letter image. However, the results show that the difference between performances of mean and median letter models is virtually nonexistent. There are only a few instances, where letter averaging has impaired the performance of the model. A closer review of averaged letter images reveals, that in these cases the averaging has obscured the signal and removed also some important notes in addition to the noise. In the well-performing letters, averaging has efficiently removed noise and transformed letters into a more general representation of a vocalization of the target species. Examples of well and poorly performing letters are shown in Figure 4.3.

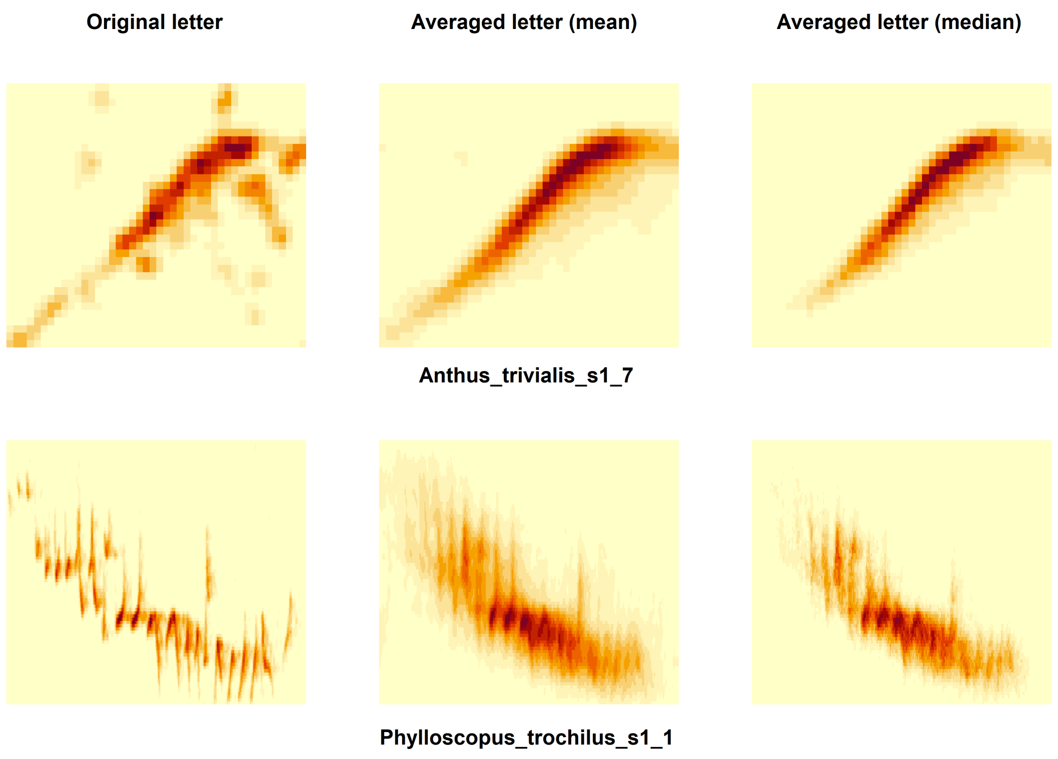


Figure 4.3: Examples of letters that perform well (Tree pipit, *Anthus trivialis*) and poorly (Willow warbler, *Phylloscopus trochilus*) after averaging.

Another alternative for removing noise from letter images was increasing the value of Gaussian blur parameter, which affects the process of transforming sound to images in ASI. This method was not as useful as letter averaging. It can be observed, that modifying the Gaussian blur indeed affects the cross-correlations and the performance of models, but no better alternative for the blurring parameter, than the current default value, was found. The results are presented in Table 4.3 and Figure 4.4.

Gaussian blur	Min.	Q_1 (25%)	Median	Mean	Q_3 (75%)	Max.
0.5	0.473	0.763	0.844	0.821	0.898	**0.975**
1.0 (original)	0.592	**0.817**	**0.860**	**0.848**	0.907	0.972
1.5	**0.608**	0.803	0.858	0.842	**0.916**	0.966
2.0	0.577	0.766	0.842	0.828	0.913	0.968

Table 4.3: Summary statistics of the distributions of letter model AUCs with different parameterizations of Gaussian blur.

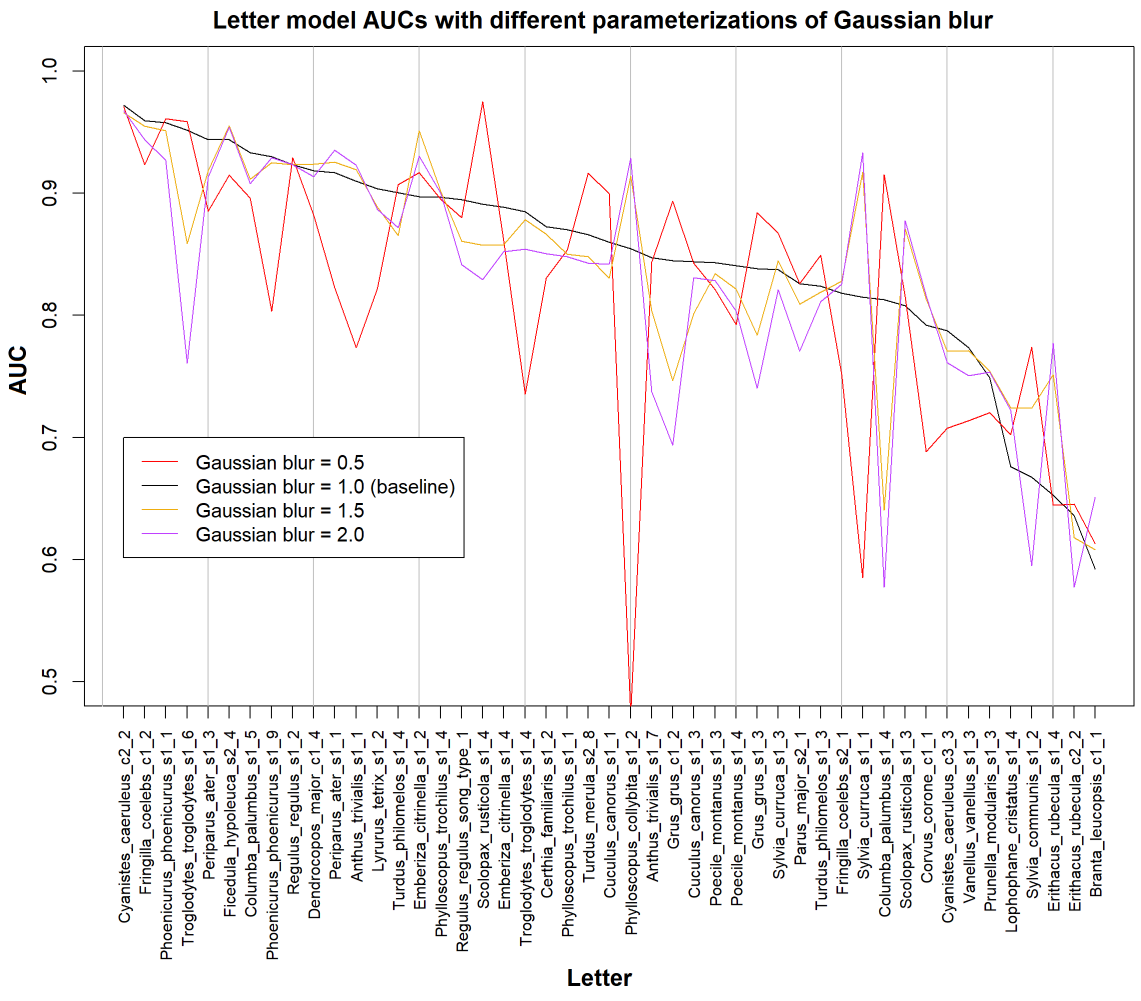


Figure 4.4: AUCs of letter-specific models with different kinds of letter image preprocessing.

4.3 Texture features as additional predictors

Texture features were inspected as additional or alternative predictors in the letter models. The models presented in this section are similar probit regression models as in the previous sections with either one or two predictors. The models with one texture feature as a predictor are generally better than a random classifier but cannot compete with the performance of cross-correlation-based baseline models. It can be deduced from Figure 4.5, that even though there were many texture features to select from, many of these features are virtually measuring the same thing in the context of bird sound spectrograms. For most of the letters, the gray curves that indicate the performance of these single-predictor models closely follow each other, which means that it is essentially irrelevant, which one of the features is used as a predictor.

Probit model predictors	Min.	Q_1 (25%)	Median	Mean	Q_3 (75%)	Max.
Cross-correlation & Contrast	0.550	**0.825**	**0.887**	**0.869**	**0.925**	0.973
Cross-correlation & MCC	0.544	0.822	**0.887**	**0.869**	**0.925**	**0.978**
Cross-correlation (original)	**0.592**	0.817	0.860	0.848	0.907	0.972

Table 4.4: Summary statistics of the distributions of letter model AUCs with and without additional predictors.

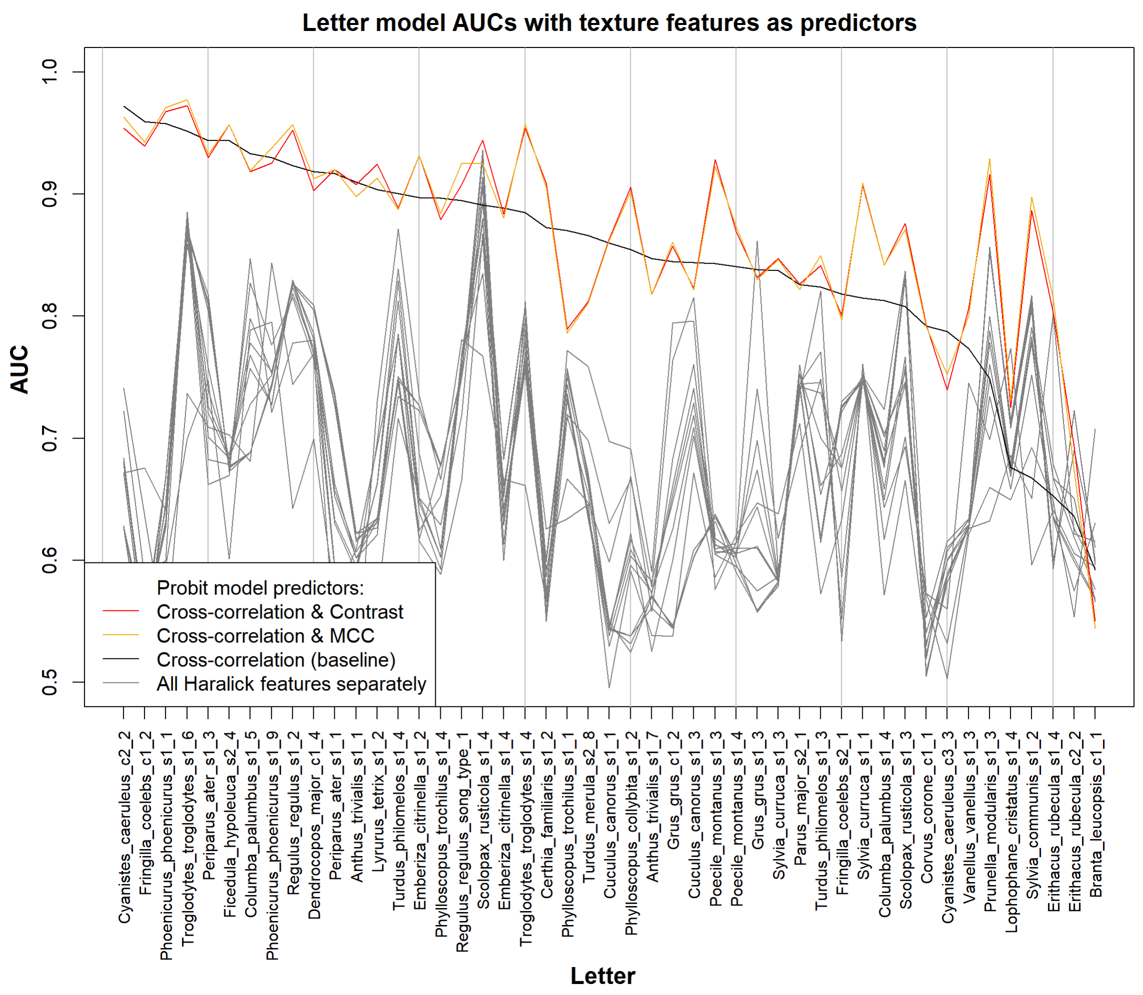


Figure 4.5: AUCs of letter-specific models with different predictors.

Contrast and maximal correlation coefficient, which seemed to perform slightly better than other texture features, were also used as additional predictors in the models together with cross-correlation. They may slightly enhance the performance of letter models, especially for short letters of small passerine birds with high-pitched vocalizations, but the improvement is not prominent. Therefore, even though it seems by rule of thumb, that the texture feature values differ between positive and negative samples, apparently this difference is already captured by cross-correlation and texture features do not bring much additional information. The results of using texture features as model predictors are presented in Table 4.4 and Figure 4.5.

4.4 Performance of convolutional neural networks

As an entirely different approach to the problem, neural networks were also included in the comparison of different methods. In this section, the concept of building several letter-specific models remains unchanged, but the models presented are no longer probit regression models, but neural networks trained to recognize vocalizations that match to a certain letter. The results shown in Table 4.5 and Figure 4.6 indicate, that CNNs clearly outperform the baseline models and are the best method so far. Model AUCs are increased for almost all letters and the impact is prominent especially in the lower end. Based on the trials with the first dataset, transfer learning and CNNs can be considered as the most eligible method.

Method	Min.	Q_1 (25%)	Median	Mean	Q_3 (75%)	Max.
Letter-specific CNNs	**0.797**	**0.886**	**0.952**	**0.933**	**0.974**	**1.000**
Baseline models	0.592	0.817	0.860	0.848	0.907	0.972

Table 4.5: Summary statistics of the distributions of letter model AUCs of letter-specific networks.

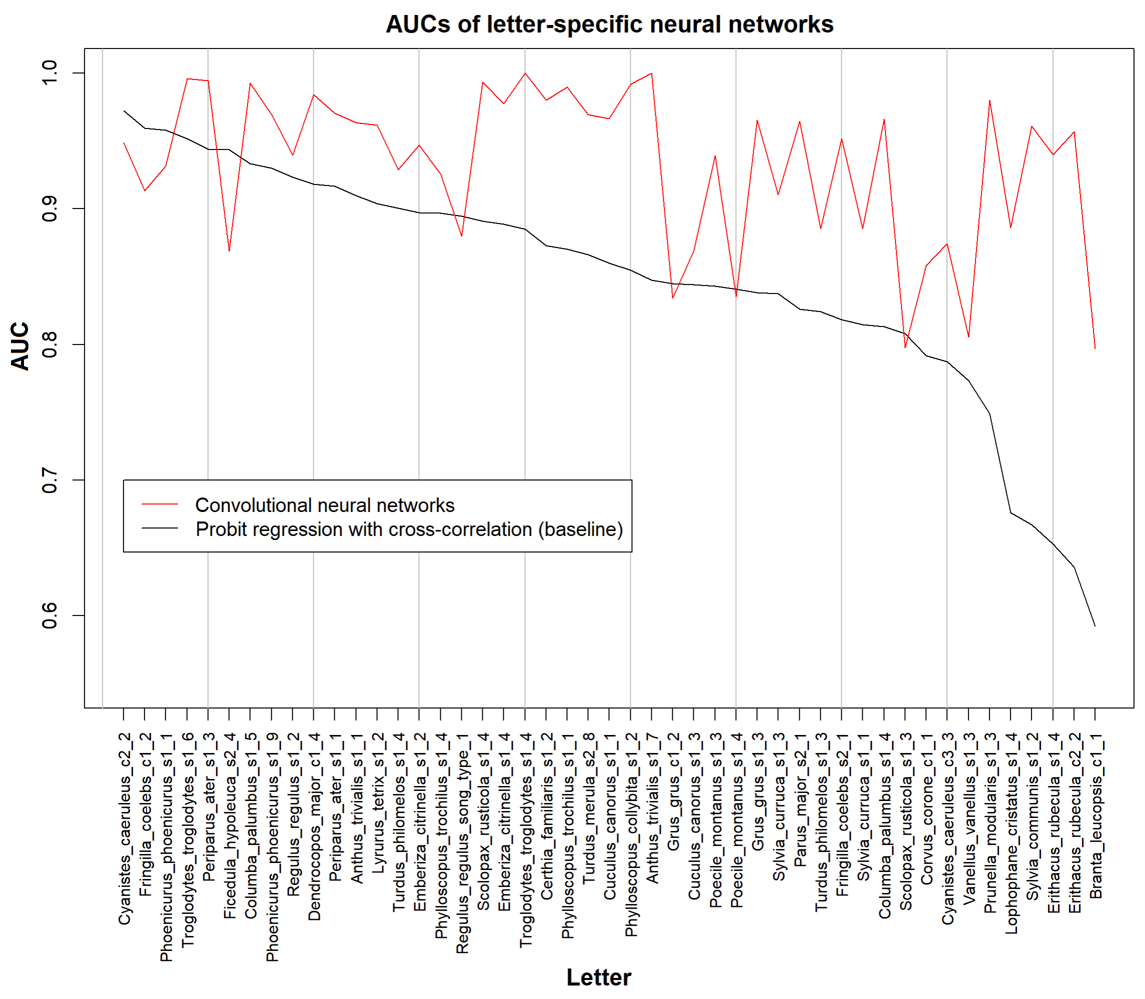


Figure 4.6: AUCs of baseline models and letter-specific CNNs.

4.5 Comparison between cross-correlation with optimized letters and transfer learning with convolutional neural networks

The final results from comparing cross-correlation with optimized letters and neural networks with the new set of letters are presented in this section. Table 4.6 and Figure 4.7 show the AUCs obtained with both of these methods and their two combinations. The AUCs of each letter model with different methods are plotted on the upper panel of the figure. For readability purposes, scientific names in the letter names on the x-axis are here replaced with widely used six-letter alpha codes, that is, the three first characters from the genus and the species of the bird. The more easily interpretable image on the lower panel shows same AUCs arranged separately in decreasing order for each method.

The findings from the test set letters follow closely those from the first set of letters.

Both letter optimization and neural networks provide improvement compared to baseline models and combinations of these methods enhance the performance even further. Optimized letters perform clearly better than original ones with cross-correlation-based probit regression. CNNs seem to work slightly better than cross-correlation with optimized letters and using extended samples/letters instead of original ones in the CNNs yields subtle further improvement. The best performing method is probit regression that combines cross-correlation for optimized letters and predictions from CNNs.

Method	Min.	Q_1 (25%)	Median	Mean	Q_3 (75%)	Max.
CNN & cross-corr. with opt. letters	**0.522**	**0.850**	**0.930**	**0.890**	**0.963**	**1.000**
CNN with extended samples	0.438	0.837	0.900	0.876	0.953	0.995
CNN with original samples	0.477	0.820	0.888	0.860	0.955	**1.000**
Cross-corr. with optimized letters	0.510	0.799	0.887	0.852	0.930	0.999
Cross-corr. with original letters	0.466	0.702	0.790	0.763	0.837	0.979

Table 4.6: Summary statistics of the distributions of letter model AUCs with different methods.

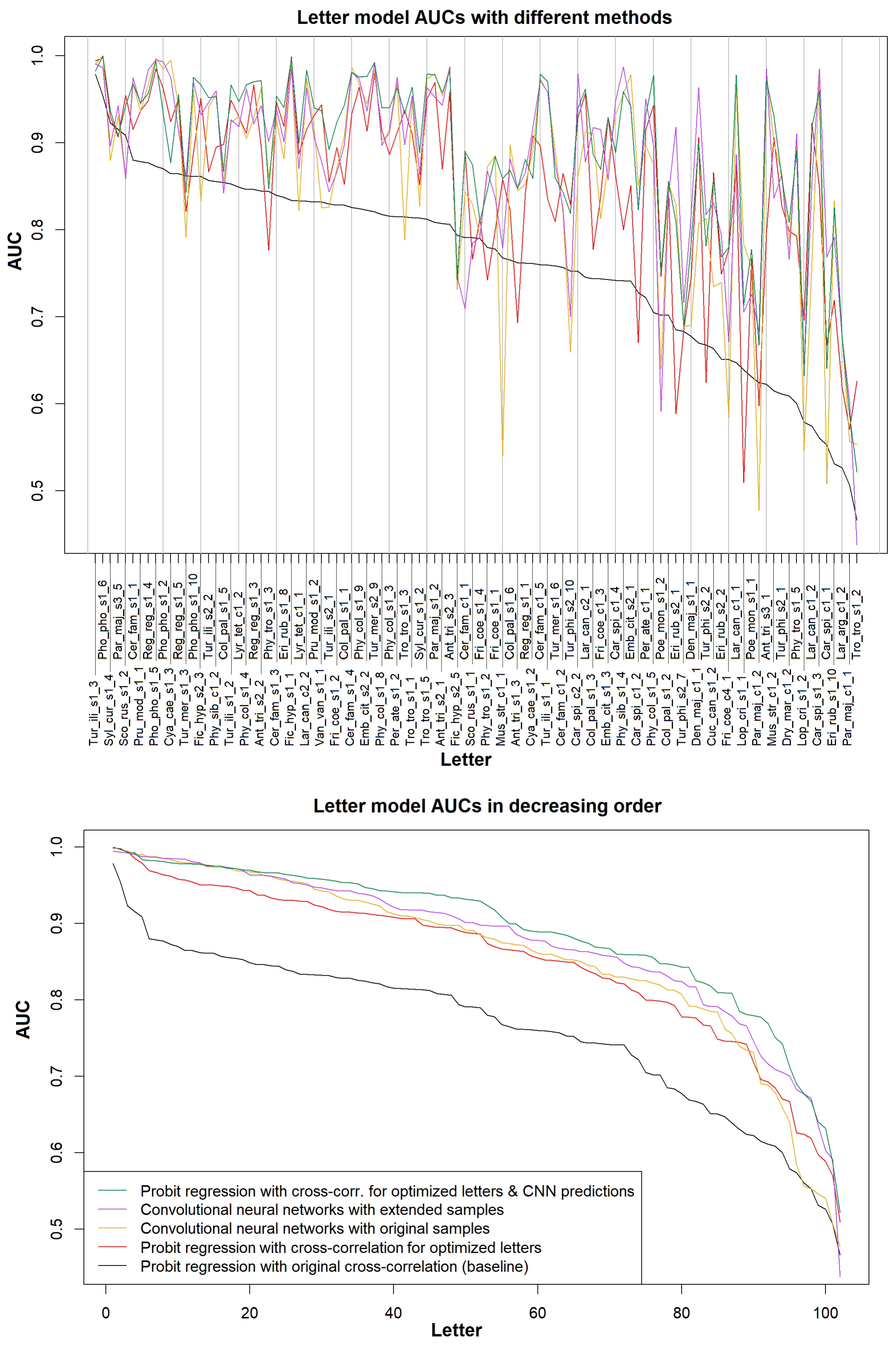


Figure 4.7: AUCs of letter-specific models with different methods for the test set letters. All tested methods clearly beat the baseline models.

Model accuracies for different methods are provided in Table 4.7 and Figure 4.8. The relative order of the methods is the same as it was when comparing the AUCs. All methods clearly beat the baseline method and the best method still seems to be probit regression that combines cross-correlation with optimized letters and CNN predictions. The difference between the best performing models and baseline models is notable. Prediction accuracy of at least 90% was achieved for 48 out of 102 letters with the best method but for only 4 letters with the baseline method. If the limit of interest is set to 80%, the corresponding numbers are 88 and 43 out of 102 letters. On average, the accuracies of letter-specific models were improved by 10.1 percentage points.

Method	Min.	Q_1 (25%)	Median	Mean	Q_3 (75%)	Max.
CNN & cross-corr. with opt. letters	**0.570**	**0.830**	**0.890**	**0.872**	**0.928**	**1.000**
CNN with extended samples	0.490	0.803	0.860	0.850	0.910	0.980
CNN with original samples	0.490	0.793	0.850	0.839	0.900	0.990
Cross-corr. with optimized letters	0.540	0.770	0.840	0.827	0.880	0.980
Cross-corr. with original letters	0.550	0.703	0.765	0.771	0.838	0.980

Table 4.7: Summary statistics of the distributions of letter model accuracies with different methods.

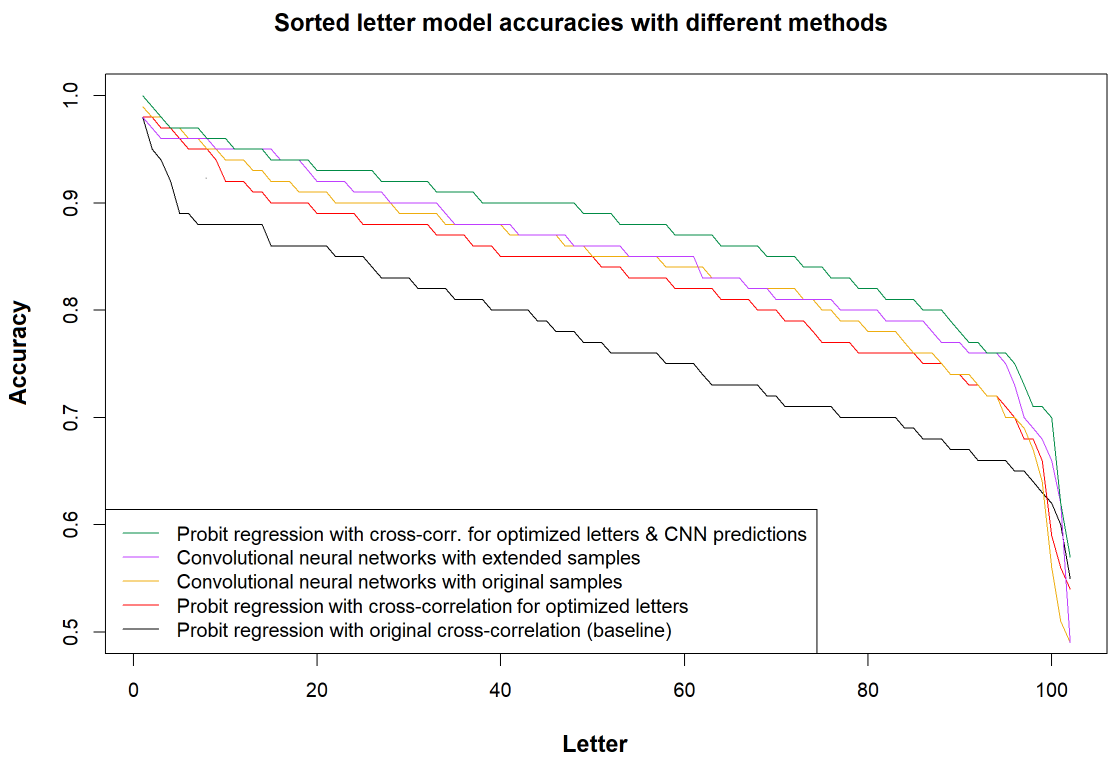


Figure 4.8: Accuracies of letter-specific models with different methods for the test set letters.

The performance of the best model, the combination of cross-correlation and CNNs, is further inspected in Figure 4.9, which shows the distributions of model predictions for positive and negative samples corresponding to each letter. In an optimal model, the distribution of predicted probabilities for positive samples would be concentrated close to 1 and for negative samples close to 0.

The letters are arranged in decreasing order according to letter model accuracies starting from the top of the leftmost image. For most letters, located primarily on the leftmost panel, predictions are reasonably good, the classes are fairly well separated, and majority of both positive and negative classes are classified correctly. Distributions of predictions for positive and negative samples overlap severely for only a few letters, Troglodytes_troglodytes_s1_2 (Eurasian wren) being the worst case with virtually undistinguishable classes and modest accuracy of 57%. Most of the poorly performing letters are short or very short and as images rather ambiguous. In some cases, the (original) letters were recognizable when listened, but their averaged image representations seem vague and nondescript. Many of the best performing letters are also very short, but they typically contain a clear structure of few consecutive sounds or one very distinct voice typical to the target species.

It is also notable, that for several letters a considerable proportion of positive samples receives predictions lower than 0.5 and is therefore incorrectly classified. For example, for Parus_major_c1_2 (great tit), Larus_argentatus_c1_2 (herring gull) and Poecile_montanus_s1_2 (willow tit), a high accuracy is achieved, even though all instances are classified as negative. This is due to the small number of positive samples, which may be caused by two reasons. Either the letter itself is poor, at least in the sense of cross-correlation, and no matches were therefore found when annotating the data, or the particular species or song type occurs in the data only rarely. It is indeed important to notice, that even though well performing models can for many letters be fit with only 100 annotated samples, it is crucial to have enough positive samples in the training data in order to construct well-functioning models.

It is obvious, that the best models which give sharp and accurate predictions for both positive and negative classes provide valuable information and are useful in species recognition. However, neither the outputs from those models, where predictions for positive and negative class partially overlap with each other, are worthless. Even if an individual letter-specific model can't alone quite distinguish between the classes and produce sharp predictions with high certainty, the produced predictions can still be useful in final species-level classification as long as these predictions are well-calibrated.

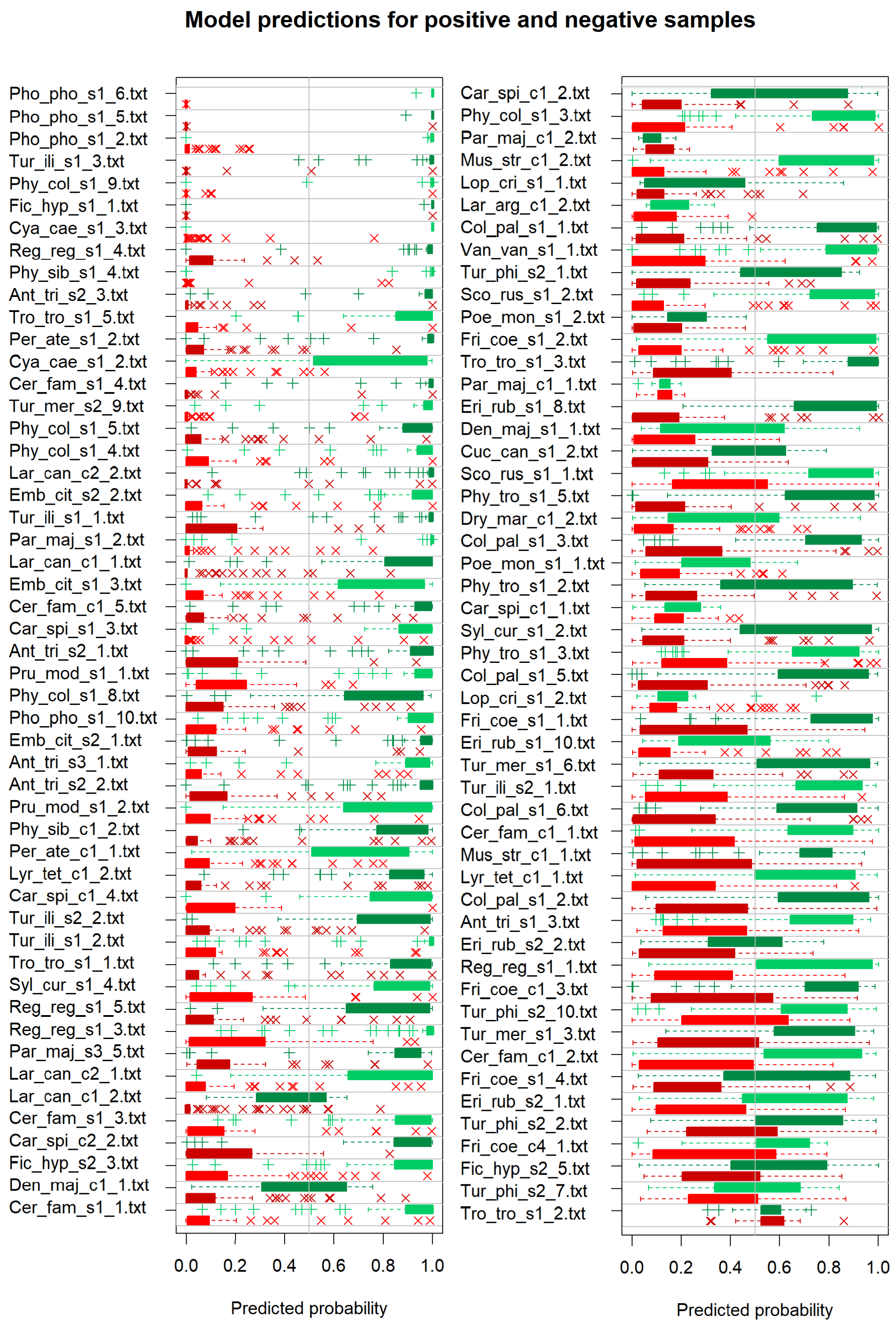


Figure 4.9: Distributions of predictions for positive and negative classes for each letter from the probit model of two regressors. Green whisker-boxes indicate the predictions for positive samples where the target species is present, and red whisker-boxes for negative samples, where the target species is not present. The box is drawn around the interquartile range from the first quartile to third quartile and contains 50% of the data points. The plot whiskers extend to the minimum and maximum values, except for observations that lie very far from the interquartile range and are considered as outliers and denoted with separate crosses. The image is split into two panels in order to fit it in one page.

The calibration of the predictions is evaluated in table 4.8. The proportions of positive samples are close to expected proportions in almost every class, which means that the predictions are mostly well-calibrated. Samples with predictions between 0.4 and 0.5 seem to contain the target species slightly less frequently than they should, but otherwise the results are satisfactory. These results are visualized in Figure 4.10, where the observed relative frequencies are plotted against the predicted class probabilities. The fact that observed frequencies increase almost linearly with the class midpoints demonstrates, that the predictions are fairly well-calibrated.

Model prediction	Positive samples / All samples	Positive samples (%)	Expected (%)
0.0 - 0.1	182 / 3996	4.6	5.0
0.1 - 0.2	130 / 993	13.1	15.0
0.2 - 0.3	133 / 623	21.3	25.0
0.3 - 0.4	145 / 431	33.6	35.0
0.4 - 0.5	128 / 337	38.0	45.0
0.5 - 0.6	210 / 396	53.0	55.0
0.6 - 0.7	235 / 380	61.8	65.0
0.7 - 0.8	317 / 419	75.7	75.0
0.8 - 0.9	443 / 515	86.0	85.0
0.9 - 1.0	2022 / 2110	95.8	95.0

Table 4.8: Calibration statistics for the predictions of two regressor probit letter models (all letter models combined). Second and third column show the proportion of positive samples out of all samples in different classes. Fourth column shows the expected proportion of positive samples for each class if the models were well-calibrated.

Overall, the results from the new set of letters indicate, that letter optimization and neural network -based approaches improve the performance of letter-specific models of ASI. Combination of these two methods produced promising results, even though some room for improvement still remains. More discussion and interpretation of these results is provided in the following chapter.

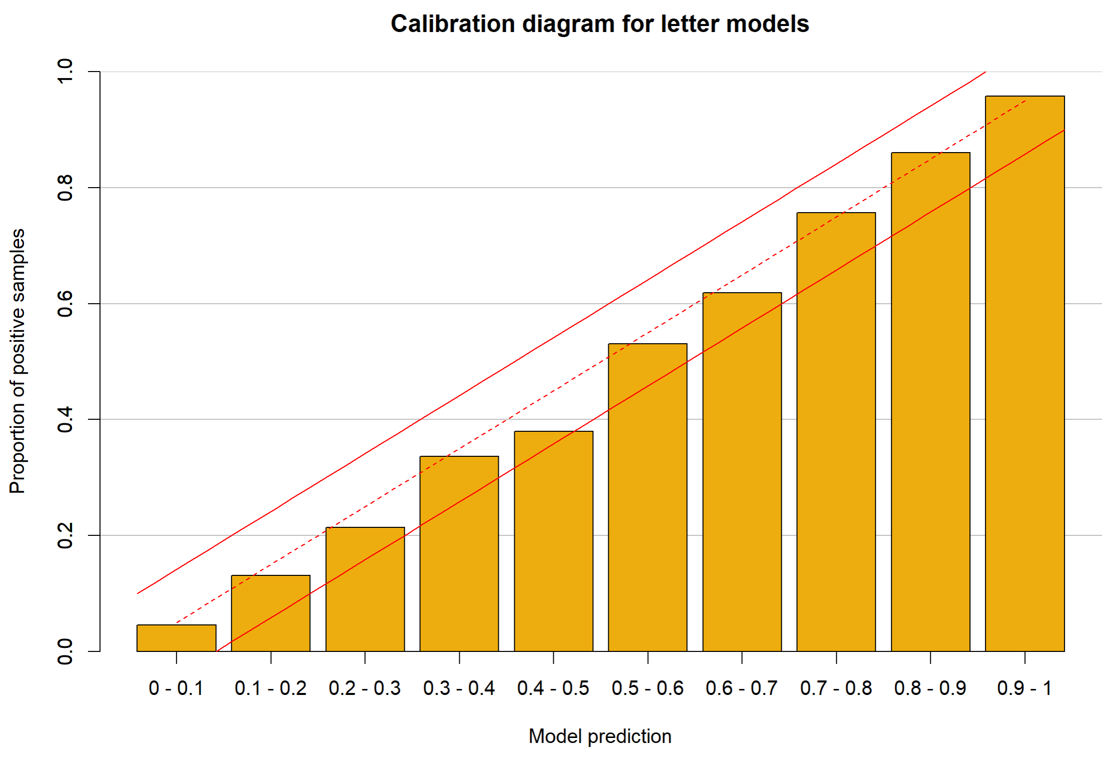


Figure 4.10: Calibration diagram for the two regressor probit model predictions for all letter models combined. With well-calibrated predictions, the proportion of positive samples in each class should correspond to the predicted probabilities. The tops of the bars should therefore be located between solid red lines, which indicate the class limits.

5. Conclusions

In this master's thesis, I have shown that the performance of letter-specific bird sound classification models can be enhanced by letter processing and involving deep learning methods, such as convolutional neural networks. Both techniques provided improvement already by themselves, and when combined, they clearly outperformed the baseline models that rely on cross-correlation between the samples and an example signal picked from the data. The best-performing method presented in this work was probit regression with two predictors; cross-correlation between the samples and an optimally cropped and averaged representation of the vocalization, and a prediction produced by a convolutional neural network, which was powered by transfer learning. This chapter is divided into two sections. I will first discuss the most important aspects regarding the interpretation of these results in Section 5.1. Possible topics to be considered in the future development are proposed in Section 5.2 in the context of the entire four-step process of automatic bird sound recognition introduced in Chapter 1.

5.1 Discussion

In this section, I list three important aspects worth noticing, while considering the results of this thesis. Firstly, because of the selection method used while annotating the data in Kerttu, the annotated data consists of samples that are especially difficult for the baseline cross-correlation models. This is also observed by the developers of ASI [40]. For each letter, most of the samples are located in a range of such cross-correlations, where the models are uncertain. In new data, the proportion of samples which have very low or very high cross-correlations with the letters would be much larger than in the current data. This would yield to higher AUCs and accuracies for the models, since a greater proportion of the samples would now be easy to classify. However, this is not necessarily the case with alternative methods. Unlike with the baseline method, the currently available annotated data is not explicitly selected to contain samples that are challenging for the new methods. If the selection of the data was weighted towards samples that are difficult for alternative models, in a similar manner as it was for baseline models, alternative models would most likely have lower AUCs and accuracies than they do with this data. Also, there is a possibility that selecting the extensions for optimized letters from a set of several options based on the model performance in different scenarios, yields to an overly optimistic picture of the actual performance of optimized letters with new data, even though an attempt was made to avoid this by using repeated cross-validation and very coarse extension techniques. Anyhow, it is possible, that the difference between improved methods and the baseline is in reality not as significant as it appears in these results.

Secondly, the models presented here are solely letter-specific models, which are meant to be combined in a later phase into species-specific models. Therefore, it is not necessarily harmful, that some of the letter-specific models are insufficient to distinguish positive and negative classes from each other or that they have low classification accuracies by themselves. There is no need for all letter models to work well by themselves, since they might still have an essential role in providing complementary information for the final classification task in the species-specific models.

Thirdly, and most importantly, as stated above, the only thing inspected here were the individual letters and the models' ability to divide a given set of samples into two classes. The final objective is to recognize different species from continuous field recordings and the main question is, how accurately can the models detect all vocalizing species. With these data and results, no conclusions about the performance of these methods in the final task can be made. It is not even certain, that the letter-based approach is suitable for identification of varied vocalizations of different bird species. To answer these questions, here presented methods should be tested with longer recordings which contain annotations for all vocalizing species.

As a concluding remark, it can be stated, that apparently better methods than the ordinary cross-correlation were found, but the journey towards a general and global solution for adequately reliable automatic bird sound identification is only at the beginning. To evaluate the actual performance of the presented methods in the final identification task, longer clips of annotated data with possibly several birds are needed.

5.2 Suggestions for future development

In Chapter 1, the process of automated bird song recognition was divided into four steps. These steps were data preprocessing (1), detection of individual vocalizations (2), feature extraction (3) and classification model building (4). Alternative strategies for steps one and two were mostly not considered in this thesis. Still, for future development it is essential to notice, that also the implementation of these steps could possibly be improved. For example in step one, different signal processing techniques could be applied.

It is evident that some information is currently lost during the conversion from sound files to images. The most widely used method for transforming signal into spectrogram is the short-time Fourier transform, which is also applied in ASI. However, an alternative technique would be wavelet transform, which has been observed to outperform STFT in mosquito detection and bird species classification tasks [29]. Spectrogram representations of sounds are also not explicit. A very common choice for the spectrogram scale is the mel scale, which is obtained from the Hertz scale by applying logarithm with base 10 and some additional coefficients. The same logarithmic base is utilized also in ASI, but different choices for the parameterization of the log-transform and the base of the logarithm might produce spectrograms with different, possibly useful, properties.

The main efforts of this thesis were concentrated on the third and fourth step of automated bird song recognition, that is, feature extraction and classification model building. Alternatively, all steps from two to four could possibly be executed within a convolutional neural network. Instead of letter-specific binary classification networks that were now used, a general network could be built for multiclass classification between bird species. A further, maybe more preferable development would be an ensemble of few such networks, in the manner of the most successful efforts in BirdCLEF 2018 [14]. Such CNNs could perhaps provide a solution that is less dependent on carefully defined letters, but more data and efficient data augmentation would certainly be needed in the effort.

Having enough data, especially a sufficient number of positive instances, and making the most out of them by applying data augmentation effectively is crucial, no matter what the final number or structure of the networks is. Several augmentation strategies are available, and it is important to choose the ones that are reasonable with bird sounds [39]. To guarantee the adequate number of annotated high-quality training data, existing libraries such as the Macaulay Library could possibly be utilized in the future.

References

[1] MATLAB code graycomatrix. https://www.mathworks.com/help/images/ref/graycomatrix.html, [3.11.2020].

[2] MATLAB code imgaussfilt. https://www.mathworks.com/help/images/ref/imgaussfilt.html, [3.11.2020].

[3] MATLAB code normxcorr2. https://www.mathworks.com/help/images/ref/normxcorr2.html, [3.11.2020].

[4] S. E. Anderson, A. S. Dave, and D. Margoliash. Template-based automatic recognition of birdsong syllables from continuous recordings. *The Journal of the Acoustical Society of America*, 100(2):1209–1219, 1996.

[5] M. Arif, R. Hedley, and E. Bayne. Testing the accuracy of a birdNET, automatic bird song classifier. 2020.

[6] G. Cawley and N. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107, 2010.

[7] F. Chollet. *Deep learning with Python.* Manning Publications Co., Shelter Island, New York, 2018.

[8] F. Chollet et al. Keras, 2015. https://github.com/fchollet/keras, [10.11.2020].

[9] Cross Language Evaluation Forum. BirdCLEF bird song identification challenge. https://www.imageclef.org/, [28.9.2020].

[10] A. P. Dawid. The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379):605–610, 1982.

[11] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[12] S. Fagerlund. Bird species recognition using support vector machines. *EURASIP Journal on Advances in Signal Processing*, 2007.

[13] T. Gneiting, F. Balabdaoui, and A. E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[14] H. Goëau, K. Stefan, H. Glotin, R. Planqué, W. P. Vellinga, and A. Joly. Overview of BirdCLEF 2018: monospecies vs. soundscape bird identification. In *CLEF: Conference and Labs of the Evaluation Forum*, 2018.

[15] K. Gurney. *An introduction to neural networks*. UCL Press, London, 1997.

[16] S. D. Hafner and J. Katz. monitoR: Acoustic template detection in R, 2018. https://CRAN.R-project.org/package=monitoR, [29.10.2020].

[17] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[18] M. Hansson-Sandsten. Classification of bird song syllables using singular vectors of the multitaper spectrogram. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 554–558, 2015.

[19] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

[20] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume II. Addison-Wesley, 1992.

[21] A. Härmä. Automatic identification of bird species based on sinusoidal modeling of syllables. *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. (ICASSP-88)*, 5, 2003.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016*, pages 630–645, 2016.

[23] M. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[24] E. D. Jarvis, C. Scharff, M. R. Grossman, J. A. Ramos, and F. Nottebohm. For whom the bird sings: Context-dependent gene expression. *Neuron*, 21(4):775–788, 1998.

[25] A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W. P. Vellinga, J. Champ, R. Planqué, S. Palazzo, and H. Müller. LifeCLEF 2016: Multimedia life species identification challenges. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. CLEF 2016*, pages 286–310, 2016.

[26] A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W. P. Vellinga, J. C. Lombardo, R. Planqué, S. Palazzo, and H. Müller. LifeCLEF 2017 lab overview: Multimedia species identification challenges. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. CLEF 2017*, pages 255–274, 2017.

[27] A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W. P. Vellinga, R. Planqué, A. Rauber, R. Fisher, and H. Müller. LifeCLEF 2014: Multimedia life species identification challenges. In *Information Access Evaluation. Multilinguality, Multimodality, and Interaction. CLEF 2014*, pages 229–249, 2014.

[28] A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W. P. Vellinga, R. Planqué, A. Rauber, S. Palazzo, R. Fisher, and H. Müller. LifeCLEF 2015: Multimedia life species identification challenges. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. CLEF 2015*, pages 462–483, 2015.

[29] I. Kiskin, D. Zilli, Y. Li, M. Sinka, K. Willis, and S. Roberts. Bioacoustic detection with wavelet-conditioned convolutional neural networks. *Neural Computing and Applications*, 32:915–927, 2020.

[30] E. C. Knight and E. M. Bayne. Classification threshold and training data affect the quality and utility of focal species data processed with automated audio-recognition software. *Bioacoustics*, 28(6):539–554, 2019.

[31] J. A. Kogan and D. Margoliash. Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden markov models: a comparative study. *The Journal of the Acoustical Society of America*, 103(4):2185–2196, 1998.

[32] M. Lasseck. Bird song classification in field recordings: winning solution for NIPS4B 2013 competition. In *Neural Information Processing Scaled for Bioacoustics- from Neurons to Big Data*, pages 176–181, 2013.

[33] M. Lasseck. Audio-based bird species identification with deep convolutional neural networks. 2018.

[34] J. P. Lewis. Fast normalized cross-correlation. *Industrial Light & Magic*, 10, 2001.

[35] U. Ligges, S. Krey, O. Mersmann, and S. Schnackenberg. *tuneR: Analysis of Music and Speech*, 2018. https://CRAN.R-project.org/package=tuneR, [4.11.2020].

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

[37] R. Monzel. MATLAB code haralickTextureFeatures, 2020. https://www.mathworks.com/matlabcentral/fileexchange/58769-haralicktexturefeatures, [3.11.2020].

[38] Museum für Naturkunde Berlin. Tierstimmenarchiv. https://www.tierstimmenarchiv.de/, [26.10.2020].

[39] L. Nanni, G. Maguolo, and M. Paci. Data augmentation approaches for improving animal audio classification. *Ecological Informatics*, 57, 2020.

[40] O. Ovaskainen, U. M. de Camargo, and P. Somervuo. Animal sound identifier (ASI): software for automated identification of vocal animals. *Ecology Letters*, 21(8):1244–1254, 2018.

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

[42] J. Podos and P. S. Warren. The evolution of geographic variation in birdsong. *Advances in the Study of Behavior*, 37:403–458, 2007.

[43] N. Priyadarshani, S. Marsland, and I. Castro. Automated birdsong recognition in complex acoustic environments: a review. *Journal of Avian Biology*, 49(5), 2018.

[44] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J. C. Sanchez, and M. Müller. pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, 12:77, 2011.

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] S. J. Sober, M. J. Wohlgemuth, and M. S. Brainard. Central contributions to acoustic variation in birdsong. *Journal of Neuroscience*, 28(41):10370–10379, 2008.

[47] K. Stefan, S. F. R., H. Goëau, H. Glotin, R. Planqué, W. P. Vellinga, and A. Joly. Overview of BirdCLEF 2019: Large-scale bird recognition in soundscapes. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum*, 2019.

[48] Suomen lajitietokeskus. Kerttu-application for bird sound annotation with citizen science. https://laji.fi/theme/kerttu/instructions, [30.10.2020].

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[50] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(6):460–473, 1978.

[51] The Cornell Lab of Ornithology. BirdNET. https://birdnet.cornell.edu/, [29.10.2020].

[52] The Cornell Lab of Ornithology. Macaulay library. https://www.macaulaylibrary.org/, [26.10.2020].

[53] The MathWorks, Inc. Image processing toolbox. https://www.mathworks.com/products/image.html, [4.11.2020].

[54] The MathWorks, Inc. Signal processing toolbox. https://www.mathworks.com/help/signal/index.html, [4.11.2020].

[55] The MathWorks, Inc. Statistics and machine learning toolbox. https://www.mathworks.com/products/statistics.html, [4.11.2020].

[56] University of Helsinki. Project LIFEPLAN - a planetary inventory of life. https://www.helsinki.fi/en/projects/lifeplan, [10.12.2020].

[57] Wildlife Acoustics. Kaleidoscope Pro. https://www.wildlifeacoustics.com/products/kaleidoscope-pro, [29.10.2020].

[58] Xeno-canto Foundation and Naturalis Biodiversity Center. Xeno-canto. https://www.xeno-canto.org/, [26.10.2020].

[59] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

[60] M. Zhong, J. LeBien, M. Campos-Cerqueira, R. Dodhia, J. L. Ferres, J. P. Velev, and T. M. Aide. Multispecies bioacoustic classification using transfer learning of deep convolutional neural networks with pseudo-labeling. *Applied Acoustics*, 166, 2020.

Appendix A. List of letters

Following letters were used in the analysis. Table shows the names of letters, width and height of the letter spectrogram in pixels, number of annotated samples for the specific letter, and how many of them contain the target species, as well as the dataset to which the letter belonged.

Letter name	Width	Height	Positive samples	Dataset
Anthus_trivialis_s1_1	196	74	55 / 100	Train
Anthus_trivialis_s1_3	44	69	52 / 100	Test
Anthus_trivialis_s1_7	43	34	37 / 100	Train
Anthus_trivialis_s2_1	49	57	48 / 100	Test
Anthus_trivialis_s2_2	122	54	58 / 100	Test
Anthus_trivialis_s2_3	45	39	24 / 100	Test
Anthus_trivialis_s3_1	66	45	39 / 100	Test
Branta_leucopsis_c1_1	27	42	28 / 100	Train
Carduelis_spinus_c1_1	23	15	16 / 100	Test
Carduelis_spinus_c1_2	47	36	27 / 100	Test
Carduelis_spinus_c1_4	68	74	42 / 100	Test
Carduelis_spinus_c2_2	25	32	46 / 100	Test
Carduelis_spinus_s1_3	67	38	21 / 100	Test
Certhia_familiaris_c1_1	41	15	49 / 100	Test
Certhia_familiaris_c1_2	54	18	46 / 100	Test
Certhia_familiaris_c1_5	87	41	43 / 100	Test
Certhia_familiaris_s1_1	201	44	49 / 100	Test
Certhia_familiaris_s1_2	119	47	36 / 100	Train
Certhia_familiaris_s1_3	115	55	55 / 100	Test
Certhia_familiaris_s1_4	58	52	48 / 100	Test
Columba_palumbus_s1_5	80	11	44 / 100	Test
Columba_palumbus_s1_1	105	9	50 / 100	Test
Columba_palumbus_s1_2	251	8	55 / 100	Test
Columba_palumbus_s1_3	84	12	54 / 100	Test
Columba_palumbus_s1_4	143	11	55 / 100	Train
Columba_palumbus_s1_5	211	15	28 / 100	Train
Columba_palumbus_s1_6	105	10	45 / 100	Test
Corvus_corone_c1_1	31	27	48 / 100	Train

Cuculus_canorus_s1_1	83	9	43 / 100	Train
Cuculus_canorus_s1_2	36	5	23 / 100	Test
Cuculus_canorus_s1_3	37	5	42 / 100	Train
Cyanistes_caeruleus_c2_2	46	38	42 / 100	Train
Cyanistes_caeruleus_c3_3	52	32	36 / 100	Train
Cyanistes_caeruleus_s1_2	56	16	12 / 100	Test
Cyanistes_caeruleus_s1_3	115	44	14 / 100	Test
Dendrocopos_major_c1_1	22	80	16 / 100	Test
Dendrocopos_major_c1_4	21	106	31 / 100	Train
Dendrocopos_major_s1_1	26	17	18 / 100	Test
Dryocopus_martius_c1_2	68	11	17 / 100	Test
Emberiza_citrinella_s1_2	35	54	12 / 100	Train
Emberiza_citrinella_s1_3	32	64	17 / 100	Test
Emberiza_citrinella_s1_4	73	17	49 / 100	Train
Emberiza_citrinella_s2_1	134	32	49 / 100	Test
Emberiza_citrinella_s2_2	56	23	47 / 100	Test
Erithacus_rubecula_c2_2	35	41	36 / 100	Train
Erithacus_rubecula_s1_10	99	40	18 / 100	Test
Erithacus_rubecula_s1_4	94	67	23 / 100	Train
Erithacus_rubecula_s1_8	121	58	50 / 100	Test
Erithacus_rubecula_s2_1	137	19	47 / 100	Test
Erithacus_rubecula_s2_2	52	40	31 / 100	Test
Ficedula_hypoleuca_s1_1	100	39	15 / 100	Test
Ficedula_hypoleuca_s2_3	35	48	47 / 100	Test
Ficedula_hypoleuca_s2_4	57	60	41 / 100	Train
Ficedula_hypoleuca_s2_5	56	43	46 / 100	Test
Fringilla_coelebs_c1_2	35	31	52 / 100	Train
Fringilla_coelebs_c1_3	46	26	54 / 100	Test
Fringilla_coelebs_c4_1	12	28	44 / 100	Test
Fringilla_coelebs_s1_1	272	101	60 / 100	Test
Fringilla_coelebs_s1_2	90	59	39 / 100	Test
Fringilla_coelebs_s1_4	100	25	38 / 100	Test
Fringilla_coelebs_s2_1	60	31	76 / 165	Train
Grus_grus_c1_2	144	13	36 / 100	Train
Grus_grus_s1_3	118	58	40 / 100	Train
Larus_argentatus_c1_2	65	20	12 / 100	Test
Larus_canus_c1_1	61	15	25 / 100	Test
Larus_canus_c1_2	46	17	11 / 100	Test
Larus_canus_c2_1	22	33	34 / 100	Test
Larus_canus_c2_2	30	29	51 / 100	Test

Lophophanes_cristatus_s1_4	43	58	14 / 100	Train
Lophophanes_cristatus_s1_1	130	80	12 / 100	Test
Lophophanes_cristatus_s1_2	34	32	14 / 100	Test
Lyrurus_tetrix_c1_1	95	11	38 / 100	Test
Lyrurus_tetrix_c1_2	135	24	48 / 100	Test
Lyrurus_tetrix_s1_2	151	6	37 / 100	Train
Muscicapa_striata_c1_1	26	36	48 / 100	Test
Muscicapa_striata_c1_2	20	44	32 / 100	Test
Parus_major_c1_1	21	65	14 / 100	Test
Parus_major_c1_2	15	58	12 / 100	Test
Parus_major_s1_2	95	41	50 / 100	Test
Parus_major_s2_1	106	35	41 / 100	Train
Parus_major_s3_5	47	64	43 / 100	Test
Periparus_ater_c1_1	37	19	19 / 100	Test
Periparus_ater_s1_1	147	72	48 / 100	Train
Periparus_ater_s1_2	37	78	37 / 100	Test
Periparus_ater_s1_3	35	70	18 / 100	Train
Phoenicurus_phoenicurus_s1_1	68	30	47 / 100	Train
Phoenicurus_phoenicurus_s1_10	64	36	44 / 100	Test
Phoenicurus_phoenicurus_s1_2	35	35	24 / 100	Test
Phoenicurus_phoenicurus_s1_5	46	60	17 / 100	Test
Phoenicurus_phoenicurus_s1_6	63	99	11 / 100	Test
Phoenicurus_phoenicurus_s1_9	70	94	18 / 100	Train
Phylloscopus_collybita_s1_2	56	46	49 / 100	Train
Phylloscopus_collybita_s1_3	19	43	46 / 100	Test
Phylloscopus_collybita_s1_4	17	55	55 / 100	Test
Phylloscopus_collybita_s1_5	10	34	35 / 100	Test
Phylloscopus_collybita_s1_8	19	68	30 / 100	Test
Phylloscopus_collybita_s1_9	21	67	46 / 100	Test
Phylloscopus_sibilatrix_c1_2	68	34	46 / 100	Test
Phylloscopus_sibilatrix_s1_4	28	36	23 / 100	Test
Phylloscopus_trochilus_s1_1	343	81	41 / 100	Train
Phylloscopus_trochilus_s1_2	107	36	34 / 100	Test
Phylloscopus_trochilus_s1_3	93	46	51 / 100	Test
Phylloscopus_trochilus_s1_4	77	27	54 / 100	Train
Phylloscopus_trochilus_s1_5	70	42	37 / 100	Test
Poecile_montanus_s1_1	30	34	16 / 100	Test
Poecile_montanus_s1_2	20	18	13 / 100	Test
Poecile_montanus_s1_3	47	40	33 / 100	Train
Poecile_montanus_s1_4	41	34	26 / 100	Train

Prunella_modularis_s1_1	138	50	51 / 100	Test
Prunella_modularis_s1_2	46	20	24 / 100	Test
Prunella_modularis_s1_3	51	30	23 / 100	Train
Regulus_regulus_s1_1	75	26	51 / 100	Test
Regulus_regulus_s1_2	44	20	53 / 100	Train
Regulus_regulus_s1_3	154	31	75 / 100	Test
Regulus_regulus_s1_4	148	48	43 / 100	Test
Regulus_regulus_s1_5	65	25	35 / 100	Test
Regulus_regulus_song_type_1	48	19	48 / 100	Train
Scolopax_rusticola_s1_1	139	23	67 / 100	Test
Scolopax_rusticola_s1_2	27	24	40 / 100	Test
Scolopax_rusticola_s1_3	32	27	15 / 100	Train
Scolopax_rusticola_s1_4	19	96	33 / 100	Train
Sylvia_communis_s1_2	36	33	15 / 100	Train
Sylvia_curruca_s1_1	20	42	54 / 100	Train
Sylvia_curruca_s1_2	40	41	33 / 100	Test
Sylvia_curruca_s1_3	61	34	61 / 100	Train
Sylvia_curruca_s1_4	19	38	58 / 100	Test
Troglodytes_troglodytes_s1_1	103	24	43 / 100	Test
Troglodytes_troglodytes_s1_2	107	40	56 / 100	Test
Troglodytes_troglodytes_s1_3	142	25	64 / 100	Test
Troglodytes_troglodytes_s1_4	146	64	50 / 100	Train
Troglodytes_troglodytes_s1_5	85	36	46 / 100	Test
Troglodytes_troglodytes_s1_6	104	41	47 / 100	Train
Turdus_iliacus_s1_1	146	47	64 / 100	Test
Turdus_iliacus_s1_2	104	53	55 / 100	Test
Turdus_iliacus_s1_3	85	28	47 / 100	Test
Turdus_iliacus_s2_1	42	34	50 / 100	Test
Turdus_iliacus_s2_2	98	38	34 / 100	Test
Turdus_merula_s1_3	106	23	55 / 100	Test
Turdus_merula_s1_6	87	28	47 / 100	Test
Turdus_merula_s2_8	209	35	41 / 100	Train
Turdus_merula_s2_9	42	47	28 / 100	Test
Turdus_philomelos_s1_3	38	57	39 / 100	Train
Turdus_philomelos_s1_4	81	41	42 / 100	Train
Turdus_philomelos_s2_1	88	28	27 / 100	Test
Turdus_philomelos_s2_10	82	42	57 / 100	Test
Turdus_philomelos_s2_2	36	109	55 / 100	Test
Turdus_philomelos_s2_7	41	39	42 / 100	Test
Vanellus_vanellus_s1_1	30	23	57 / 100	Test
Vanellus_vanellus_s1_3	34	21	52 / 100	Train

Appendix B. Textural features by Haralick et al.

Following the notation of the original paper by Haralick et al. [19], the textural features calculated from GLCMs are defined by following equations.

- **Angular Second Moment**:

$$f_1 = \sum_i \sum_j p(i,j)^2 \tag{B.1}$$

- **Contrast**:

$$f_2 = \sum_{n=0}^{N_{g-1}} n^2 \Big(\sum_{i=1}^{N_g} \sum_{j=1, |i-j|=n}^{N_g} p(i,j) \Big) \tag{B.2}$$

- **Correlation**:

$$f_3 = \frac{\sum_i \sum_j (ij) p(i,j)^2 - \mu_x \mu_y}{\sigma_x \sigma_y}, \tag{B.3}$$

  where $\mu_x, \mu_y, \sigma_x, \sigma_y$ are the means and standard deviations of p_x and p_y.

- **Variance**:

$$f_4 = \sum_i \sum_j (i - \mu)^2 p(i,j) \tag{B.4}$$

- **Inverse Difference Moment**:

$$f_5 = \sum_i \sum_j \frac{1}{1 + (i-j)^2} p(i,j) \tag{B.5}$$

- **Sum Average**:

$$f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i) \tag{B.6}$$

- **Sum Variance**:

$$f_7 = \sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i) \tag{B.7}$$

- **Sum Entropy**:

$$f_8 = - \sum_{i=2}^{2N_g} p_{x+y}(i) \log(p_{x+y}(i)) \tag{B.8}$$

- **Entropy**:

$$f_9 = - \sum_i \sum_j p(i,j) \log(p(i,j)) \tag{B.9}$$

- **Difference Variance**:

$$f_{10} = \mathrm{Var}(p_{x-y}) \tag{B.10}$$

- **Difference Entropy**:

$$f_{11} = -\sum_{i=0}^{N_{g-1}} p_{x-y}(i)\log(p_{x-y}(i)) \tag{B.11}$$

- **Information Measure of Correlation I**:

$$f_{12} = \frac{f_9 - HXY1}{\max(HX, HY)}, \tag{B.12}$$

where HX and HY are entropies of p_x and p_y,

$HXY1 = -\sum_i \sum_j p(i,j)\log(p_x(i)p_y(j))$ and

$HXY2 = -\sum_i \sum_j p_x(i)p_y(j)\log(p_x(i)p_y(j))$.

- **Information Measure of Correlation II**:

$$f_{13} = \sqrt{1 - e^{-2(HXY2 - f_9)}} \tag{B.13}$$

- **Maximal Correlation Coefficient**:

$$f_{14} = \text{2nd largest eigenvalue of } Q^{1/2}, \tag{B.14}$$

where $Q(i,j) = \sum_k \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)}$,

and

$p(i,j)$ denotes the (i,j)th entry in a normalized GLCM,

$p_x(i)$ denotes the ith entry in the marginal-probability matrix obtained by summing the rows of $p(i,j)$,

N_g denotes the number of distinct gray levels in the quantized image,

$\sum_i = \sum_{i=1}^{N_g}$ and $\sum_j = \sum_{j=1}^{N_g}$,

$p_y(j) = \sum_{i=1}^{N_g} p(i,j)$,

$p_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1, i+j=k}^{N_g} p(i,j) \qquad k = 2, 3, \ldots, 2N_g$ and

$p_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1, |i-j|=k}^{N_g} p(i,j) \qquad k = 0, 1, \ldots, N_g - 1$.