A Deep Neural Network Approach to the LifeCLEF 2014 Bird task

Hendrik Vincent Koops, Jan van Balen, and Frans Wiering

Utrecht University
Department of Information and Computing Sciences
{h.v.koops,j.m.h.vanbalen,f.wiering}@uu.nl
http://www.cs.uu.nl/staff/cur/AFD/iat.html

Abstract. This paper describes the methods that are used in our submission to the LifeCLEF 2014 Bird task. A segmentation algorithm is created that is capable of segmenting the audio files of the Bird task dataset. These segments are used to select relevant Mel-Frequency Cepstral Coefficients (MFCC) frames from the MFCC dataset. Three datasets are created, 48: containing only the mean MFCC per segment, 96: containing the mean and variance of the MFCCs in a segment, and 240: containing the mean, variance and the mean of three sections. These dataset are shuffled and split in a test and train set to train Deep Neural Networks with several topologies, which are capable to classify the segments of the datasets. It was found that the best network was capable of correctly classifying 73% of the segments. The results of a run from our system placed us 6th in the list of 10 participating teams. In a follow-up research it is found that shuffling the data before splitting introduces overfitting, which can be reduced by not shuffling the datasets prior to splitting, and using dropout networks.

Keywords: Deep Learning, Neural Networks, Feature Learning, Birdsong Recognition, Bioacoustics

1 Introduction

The LifeCLEF 2014 Bird task is a bird identification task based on different types of audio records over 501 species from South America centred on Brazil. This paper will describe the methods used to create the *"Utrecht Univ."* run that was submitted to the task.

Features are predominantly handcrafted in audio information retrieval research. Handcrafting features relies on a significant amount of domain- and engineering knowledge to translate insights into algorithmic methods. Building features on heuristics makes the process of good feature extraction a difficult problem, as it hinges on the assumption that the feature designer can know what a good representation of a signal must be to solve a problem. Feature design is thus constrained by what a designer can conceive and comprehend. Furthermore, manual optimisation of handcrafted features is a slow process, costly in effort.

A research area that tries to solve some of the aforementioned problems in feature design is called *deep learning*, which uses deep architectures to learn *feature hierarchies*. The features that are higher up in deep hierarchies are formed by the composition of features on lower levels. These multi-level representations allow a deep architecture to learn the complex functions that map the input (such as digital audio) to output (e.g. classes), without the need of dependence on manual handcrafted features.

In our submission, we used *deep neural networks* (neural networks with several hidden layers) to facilitate hierarchical feature learning and classification of birdsongs. Our system consists roughly out of tree composed steps:

1. Automatic segmentation of noisy bird sounds
2. Extracting relevant Mel-Frequency Cepstral Coefficients (MFCC) features
3. Feature learning and classification with Deep Neural Networks

The following sections will describe these steps.

2 Automatic Segmentation of Noisy Bird Sounds

A dataset containing audio data normalised to 44.1 kHz, .wav format (16 bits) is provided for the Bird task. In addition to audio recordings, a dataset containing Mel-Frequency Cepstral Coefficients (MFCC) is provided. This MFCC dataset contains comma separated value files corresponding to the audio files, containing 48 coefficients per line, computed on windows of 11.6 ms, each 3.9 ms. The 48 coefficients are the first 16 MFCCs and their speed an acceleration.

To facilitate the learning of features on only the relevant birdsong parts of the audio files, a segmentation algorithm is created that is based on the assumption that the loudest parts of an audio recording are the most relevant. The algorithm consists of three major sections:

1. Decimating and dynamic filtering of the input signal
2. Detection of segments
3. Clustering of segments into higher temporal structures

2.1 Decimating and Dynamic Filtering

The elements of the BirdCLEF2014 dataset are normalised to a bit rate of 16 bits per second and a sample rate of 44100 Hz. This input signal is downsampled by a factor 4, resulting in a signal with a sample rate of 11025 Hz. Decimation of a signal is common practice in speech recognition, as it reduces the amount of information by removing the top part of the spectrum that we know cannot hold the most important information. The spectrum energy in song birds is typically concentrated on a very narrow area in the range of 1 to 6 kHz [2]. By decimating the input signal by a factor 4, we reduce the highest possible frequency to 5512.5 Hz. Although some bird song overtone information could exist beyond 5 kHz, this is never the loudest frequency.

After decimation, the signal is passed through a high pass filter F_1 with a passband frequency of 1kHz, to filter unwanted low frequency noise such as continuous noise from recording equipment, low frequency animal sounds and mechanical sounds. This filter is a 10^{th} order high pass filter with a passband frequency of 1kHz and a stop band attenuation of 80 dB. From this resulting signal, the fundamental frequency is calculated by finding the maximum value of the spectrogram. This value, f_0, is then used in a adaptive filter.

The signal is filtered by another high pass filter that adapts its passband frequency to $0.6 * f_0$. An adaptive filter such as this can account for loud sounds that occur below the bird sound in the spectrogram. This filter is also a 10^{th} order high pass filter, with a passband frequency of $0.6 * f_0$ Hz and a stop band attenuation of 80 dB.

2.2 Detection of Segments

This part of the algorithm uses an energy-based algorithm somewhat similar to [3] to detect the loudest elements in the spectrogram of the decimated and filtered input signal. In the spectrogram $|S(f_n, t_n)|$ of a signal, the maximum value of f_n with t_n is found. From this position t_n, the frequency parameter $\omega_n(0) = f_n$ and amplitude value $a_n(0) = 20\,log_{10}\,|S(f_n, t_n)|$ [dB] is stored. From this position $|S(f_n, t_n)|$, a left and right wise trace from the maximum peak of $S(f, t)$ for $t > t_0$ and for $t < t_0$ until $a_n(t - t_0) < a_n(0) - T$dB is performed. From a visual inspection on a subset of the BirdCLEF2014 dataset, the stopping criteria T of 17 dB for marking the boundary of a section was found to create the best positions. The obtained frequency and amplitude trajectories corresponding to the n^{th} annotated part are stored in functions $\omega_n(\tau)$ and $a_n(\tau)$, where $\tau = t_0 - t_s, \ldots, t_0 + t_e$. By removing the annotated part from starting position t_s until end position t_e $S(f, [t_s, t_s + 1, \ldots, t_e])$ from the spectrogram, a reanalysis of the same area is prevented. These steps are repeated N times until no maximum value in the spectrogram above 17 dB is found, resulting in N annotated segments with corresponding start and end time values.

2.3 Clustering of Segments into Higher Temporal Structures

A problem with the aforementioned segmentation algorithm, especially in recordings with little background noise, is that a lot of small areas of only a few milliseconds in length are detected. Bird songs are better described at a higher temporal level, which is richer in information. The smaller sections are therefore merged into larger chunks, representing larger parts of the bird song.

After experimenting with several clustering techniques, it was found that simple gap-wise merging of smaller sections was the most successful in creating meaningful larger sections. Larger chunks are created by analysing the silences between sections and merging subsequent sections with silences smaller than a m milliseconds into one annotated section. Because the segmentation algorithm in the previous step does not annotate each found segment in rank of loudness, the segments need to be sorted first. From this sorted list of n segments S_n, each gap

between S_i and S_{i+1} is analysed. If the difference between the beginning of S_{i+1} and the end of S_i is smaller than m milliseconds, then let $S_i = S_i + S_{i+1}$ and each subsequent section S_{i+2} is demoted one position in the sorted list of segments, i.e. $S_{i+2} = S_{i+1}$. From an evaluation experiment in which handcrafted segments were compared to automatically generated segments, $m = 800$ milliseconds was found to create good segments.

The start and end boundaries of the segments of each file are stored in comma separated value (csv) files, with each line representing one segment. These annotations will be used to select the part of a file from the BirdCLEF2014 data set.

3 Extracting Relevant Mel-Frequency Cepstral Coefficients (MFCC) Features

The start and end boundaries of the segments created by our algorithm are used to select MFCC features from the MFCC dataset that was included in the BirdCLEF2014 training set. Using the calculated segments, three datasets are created:

- 48: only the mean of the MFCCs of a segment is calculated
- 96: the mean and variance of the MFCCs of a segment are calculated
- 240: the mean, variance, speed, acceleration and each mean of the section divided in three equal parts are calculated

Table 1 depicts an overview features contained in the 48, 96, and 240 dataset. Each dataset contains 46799 segments, which equals to around 4.83 segments per file. The datasets are shuffled and divided into a 75% training and 25% test set, and used as input for several deep neural network architectures.

Table 1. Contents of different network sizes with Mel-Frequency Cepstral Coefficients (MFCC) input.

	48	96	240
Mean (48)	✓	✓	✓
Variance (48)		✓	✓
Speed (48)			✓
Acceleration (48)			✓
Means of three sections (48×3)			✓

4 Feature Learning and Classification With Deep Neural Networks

Artificial Neural Networks (ANNs) are a class of machine learning techniques that take inspiration from the physical structure and workings of the animal

brain. Recent developments in neural network research unveiled ways to train neural networks with more than one hidden layer, which developed a research area now known as *deep learning* [5].

Deep neural networks with several topologies are used in feature learning and classification experiments. All networks are implemented in Python, a widely used general-purpose, high-level programming language, using the numerical computation library Theano [4]. Training and testing errors of these topologies can be found in Table 2.

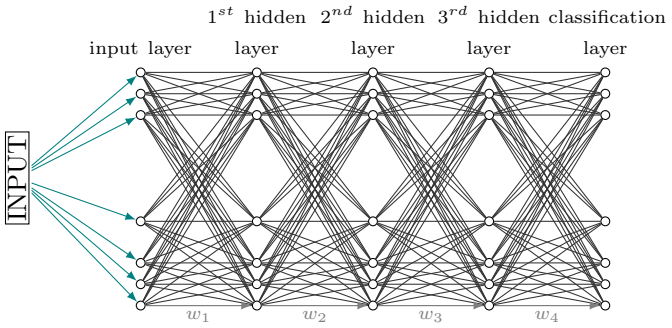

Fig. 1. Abstract depiction of a three-hidden layer Deep Artificial Neural Network used in experiments.

The neural networks take as input the elements from the 48, 96, and 240 datasets, and are trained to classify the segments as one of the 501 species. Figure 1 shows an abstract version of a three-hidden layer Deep Artificial Neural Network used in experiments. The implementations in this experiment use *mini-batch optimisation*, in which the updates of the parameters of the network are based on the summed gradients measured on a rotating subset of the complete training set. To achieve mini-batch optimisation, a selection mechanism is implemented that randomly selects a subset of size n from the data set, in which n is a size that is a parameter to the main call to the network. During early testing and implementation, it was found that a batch size of 250 examples returned favourable results. In all of the experiments in this research, a batch size of 250 is used, which means that at every iteration of training phase, 250 randomly selected training examples are passed though the network, from which a gradient and updates are computed.

The input layers of the network correspond with the dataset that is used: networks with input layer size 48 are trained and tested with the 48-dataset, the same holds for the 96 and 240 input layer sizes. For the hidden layers, two approaches are used in experiments: one in which the hidden layer size is smaller than the input layer, and one in which the hidden layer is larger than the input layer. For the networks with input layer size 48, experiments are carried out with hidden layer sizes of 48 and 40. For the networks with input layer size

96, networks with hidden layer sizes 64 and 84 are used. For networks with input layer size 240, experiments are carried out with hidden layer sizes 128 and 350. All of these networks contain two hidden layers of equal size, except for networks with input layer size 240, two experiments are carried out with three hidden layers. The classification layer is always of size 501 in every network, simply because all the audio files in the BirdCLEF2014 data set belong to one of 501 species.

4.1 Voting

The input to the networks are segments, and the networks therefore perform *segment based* classification. The goal of the LifeCLEF 2014 Bird task is to classify *files*, which means a mechanism is needed to use the segment classification for file classification. Several voting schemes are implemented to accomplish this, of which we will discuss the one with the best performance: *activation-weighted voting*.

In *activation-weighted voting*, the activations of the classification layer of the network is used to create a weighted vector of classes from which the mode is chosen. Let F be segmented into five segments $\{s_1, s_2, s_3, s_4, s_5\}$, and assume there are three classes C_1, C_2 and C_3. Now let $[0.1, 0.1, 0.8]$ be the activations for the classes for s_1. Create a new vector w_1 and add C_1 one time, C_2 one time and C_3 eight times. Repeat for all segments in the file, and concatenate every w_n for n $= 1 \ldots 5$ in w. This results in a weighted vector w from which the mode can be taken, which will be the class of F. The result of this voting mechanism on the output of several neural networks can be found in the third column of Table 2.

The advantage of this approach is the usage of the complete layer activations as probabilities, thereby taking advantage of the network's classification uncertainty. A drawback of this scheme is that it can become slow for a large number of species and a large dataset. Fortunately, the Python package Numpy has great optimisation strategies to deal with this kind of vector processing that minimise computation time.

5 Results and Submission

Table 2 shows training and testing results of experiments with various network topologies using shuffled 48, 96 and 240 Mel-Frequency Cepstral Coefficients (MFCC) datasets. It is found that on average, networks with an input layer of size 240 perform the best on the created MFCC datasets.

The first two rows of Table 2 show that of the networks with an input layer of size 48, the network with a hidden layer size of 40 outperforms the network with a hidden layer of size 48 without voting on the test set.

A big difference can be found in the testing accuracy of the networks with input size 96. The network with hidden layer size 64 outperforms the network

Table 2. Training and testing results of experiments with various network topologies using shuffled 48, 96 and 240 Mel-Frequency Cepstral Coefficients datasets. The first number in the network topology describes the input layer size, the last number describes the output layer size. The numbers in between describe the amount and size of the hidden layers.

Network topology	Training error	Test error	Test error with voting
48-40-40-501	63.8 %	66.16 %	76.91 %
48-48-48-501	64 %	74.9 %	82 %
96-64-64-501	42.6 %	41.56 %	40.0 %
96-84-84-501	25.0 %	56.04 %	55 %
240-128-128-501	0 %	93.6 %	93.7 %
240-350-350-501	0 %	31.1 %	27 %
240-128-128-128-501	0 %	48.5 %	48.1 %
240-350-350-350-501	0 %	38.2 %	38.5 %

with hidden layer size 84, with and without voting. For both the 48 and 96 networks it is found that increasing the hidden layer size decreases the classification accuracy. This is not the case in the networks with input layer size 240, of which four networks are created with different topologies.

In the 240-networks with two hidden layers, the network with hidden layers of size 350 outperforms the network with hidden layers of size 128 with a dramatic increase of 62.5%. The bad performance of the network with two hidden layers of size 128 could be attributed to the relative large difference between input layer and hidden layer size. Then again, in the network with three hidden layers of size 128, a classification accuracy is obtained that is close to the accuracy of the network with three hidden layers of size 350. The reason for this might be that by adding more hidden layers, the network is capable of learning a higher representation of the input data. The networks with three hidden layers again show an increase in accuracy when the hidden layer size is increased.

Overall, it shows that when using voting, the classification accuracy increases on average with 0.02 percent points. If the the big drop in accuracy due to voting in the 48-48-48-501 network is excluded, the average accuracy increases with 4.1 percent points using voting.

The best performing neural network (240-350-350-501) was able to classify the segments with 27 % error (i.e. 73 % accuracy). This was chosen to calculate the submitted run. The network was retrained using the entire segmented training set, and its classification results are transformed into the BirdCLEF comma separated value file format. This run is submitted to the BirdCLEF submission system, as `1400099635524__clef_240350350v` under the team name *"Utrecht Univ. Run 1"*.

Based on the Mean Average Precision (MAP), our run ranks 15th out of 29 submitted total runs. By comparing the best runs from each group, we are reported to be 6th in the list of 10 participating teams. The MAP of our run with background species is 0.123 and the MAP without background species is 0.14.

6 Conclusion and Discussion

After receiving the results, and observing the large difference between our results and the results calculated by the LifeCLEF 2014 Bird task committee, we re-evaluated our system. It is believed that overfitting of the neural network occurs when the obtained segments from the elements from the BirdCLEF dataset are shuffled prior to splitting the dataset in a train and test set. Shuffling before dividing the dataset dramatically raises the probability of segments of one file being present in both the test and train set, which could result in the neural networks learning a wrong underlying relationship (such as microphone characteristics, auditory location characteristics, recorder noise or a particular individual bird) in the training set. This in turn results in a poorer performance in the test set.

Therefore, after receiving the results from all the teams, the datasets are re-created and split without shuffling. In addition to the new datasets, new networks are created that incorporate dropout [1], to prevent overfitting. The obtained results so far are closer to the results obtained by other teams in the LifeCLEF 2014 Bird task, with test set classification accuracies ranging between 10% and 20%.

7 Future Work

At the time of writing this paper, it is not clear where overfitting of the neural networks precisely has occurred. It would be interesting to compare the metadata that was included in the BirdCLEF2014 datasets with classification results by the neural networks. This might show what unwanted relationships are learned by the neural networks, which in turn could help with preventing overfitting in the future.

It would also be interesting to see how the performance of other deep neural network architectures, such as stacked autoencoders or convolutional neural networks compare to our implementation.

References

1. Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, volume abs/1207.0580, 2012.
2. Aki Harma. Automatic identification of bird species based on sinusoidal modeling of syllables. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, pages V–545. IEEE, 2003.
3. Panu Somervuo and Aki Harma. Bird song recognition based on syllable pair histograms. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 5, pages V–825. IEEE, 2004.

4. James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.
5. Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, MIT Press, 2006.
6. Goëau, Hervé and Glotin, Hervé and Vellinga, Willem-Pier and Rauber, Andreas. LifeCLEF Bird Identification Task 2014. *CLEF working notes 2014*, 2014.