Audio-based Bird Species Identification with Deep Convolutional Neural Networks

Mario Lasseck

Museum fuer Naturkunde Berlin, Germany
Mario.Lasseck@mfn.berlin

Abstract. This paper presents deep learning techniques for audio-based bird identification at very large scale. Deep Convolutional Neural Networks (DCNNs) are fine-tuned to classify 1500 species. Various data augmentation techniques are applied to prevent overfitting and to further improve model accuracy and generalization. The proposed approach is evaluated in the BirdCLEF 2018 campaign and provides the best system in all subtasks. It surpasses previous state-of-the-art by 15.8 % identifying foreground species and 20.2 % considering also background species achieving a mean reciprocal rank (MRR) of 82.7 % and 74.0 % on the official BirdCLEF Subtask1 test set.

Keywords: Bird Species Identification, Biodiversity, Deep Learning, Deep Convolutional Neural Networks, Data Augmentation.

1 Introduction

A system for audio-based bird identification has proven to be particularly useful for biodiversity monitoring and education. It can help professionals working with bioacoustic sounds by processing large audio collections in a fast and automated way. Or help amateurs to identify birds in self-made recordings. The LifeCLEF bird identification task challenges participants to identify different bird species in a large collection of audio recordings provided by Xeno-Canto [1]. The number of training files as well as the number of species to identify constantly grew over the last few years. Starting with 501 species in 2014 the number was doubled in 2015. Since 2016 there are 1500 different species to identify in the test sets. An overview and further details about the BirdCLEF tasks are given in [2]. They are among others part of the LifeCLEF 2018 evaluation campaign [3].

Distinguishing between so many different birds is a challenging fine-grained classification problem. Until 2016, the best system used template matching of small sound elements combined with a large number of acoustic features and a random forest ensemble learning method for classification [4],[5]. With the rise of deep learning, since 2016, the best systems are based on convolutional neural networks [6],[7]. The approach described in this paper also uses neural networks and deep learning. It is building on the work of previous solutions to the task and combines proven techniques with new methods, especially regarding data preparation and augmentation.

2 Implementation Details and Model Training

Data Preparation

Audio files provided in the training and test sets are not only of different duration and quality, they also have different formats regarding sample rate, bit depth and number of channels. In a first step, two data sets are formed with homogeneous file properties. For the first set all files are resampled to 44.1 kHz followed by normalization to a maximum signal amplitude of -3 dB. For a second set all files are first high pass filtered at a frequency of 2 kHz (Q = 0.707), resampled to 22050 Hz, mixed to mono and finally also normalized to -3 dB. The training set is augmented by additionally extracting 381 audio files using the time coded annotation of species in the metadata of the newly provided soundscape validation set. All files belonging to the training set are further processed to create additional data sets with different content described more detailed below:

- *BirdsOnly*
- *NoiseOnly*
- *AtmosOnly*
- *LowQuality*

Via segmentation, the audio content of each training file is separated in signal and noise parts. Segmentation is done in frequency domain applying image processing methods like median clipping [8] and further morphological operations on the spectrogram image to extract individual sound events. The method was already successfully used in previous bird identification tasks amongst others to extract segments for feature engineering via template matching [4,5,9]. The position of audio segments in time can be used to divide an audio file into signal parts, representing bird call or song elements, and noise parts, representing background sounds or noise. One data set is formed by concatenating all frames of a file belonging to bird sounds (*BirdsOnly*) and another one by concatenating all frames belonging to background sounds (*NoiseOnly*). Since the noise files are created by concatenating small parts, sometimes only a few milliseconds in duration, taken from various positions of the original source file, they can have rapid changes in amplitude and frequency. To get a more realistic set of background atmosphere a third data set (*AtmosOnly*) is formed using and concatenating only parts consisting of longer sequences of successive frames related to noise with an overall duration greater than one second. This set is significantly smaller compared to the other sets since most files don't have parts without birds singing for a longer period of time.
Many recordings from the community-based online platform Xeno-Canto are created by amateurs with non-professional equipment. To simulate low quality or highly compressed recordings a fourth data set (*LowQuality*) has been created by encoding all training files to MP3 with a low target bit rate (lame parameter: -V7) and decoding them back to WAV format afterwards.

For model evaluation the training data is split into training and validation sets. This is done by creating 8 stratified folds. For model training only three of them are actually used with individual validation sets of ca. 10 to 20 percent of the training set. To get a more realistic estimation of model performance validation sets are created with respect to bird individuals. An *IndividualID* was assigned to each audio file in order to achieve all files belonging to the same bird are either part of the training or the validation set. Files got the same unique *IndividualID* if they belong to the same class and were recorded by the same author at the same day.

Training Setup

For audio-based bird identification DCNNs are trained via PyTorch [10]. Different models pre-trained on the ImageNet data set are fine-tuned with spectrogram images representing short audio chunks. For audio file reading and processing the PySoundFile and librosa [11] python packages are utilized. The basic data loading pipeline can be summarized as follows:

- extract audio chunk from file with a duration of ca. 5 seconds
- apply short-time Fourier transform
- normalize and convert power spectrogram to decibel units (dB) via logarithm
- convert linear spectrogram to mel spectrogram
- remove low and high frequencies
- resize spectrogram to fit input dimension of the network
- convert grayscale image to RGB image

In each training epoch all training files are processed in random order to extract audio chunks at random position. Training is done with a batch size of ca. 100 - 200 samples using either 2 or 3 GPUs (Nvidia Geforce 1080 and 1080 Ti). Categorical cross entropy is used as loss function considering only foreground species as ground truth targets. Stochastic gradient descent is used as optimizer with Nesterov momentum of 0.9, weight decay of 1e-4 and an initial learning rate of 0.01. Learning rate is decreased during training by ca. 10^{-1} in 3 to 4 steps until 0.0001 whenever performance on the validation set stopped improving.

Validation and post-processing for submission. For the validation and test set audio chunks are extracted successively from each file and channel (if not mono) with an overlap of 10 % for validation files during training and 80 % for files in the test set. Predictions are averaged for each file (Subtask1: monophone recordings) and time interval (Subtask2: soundscape recordings) by taking the mean over all chunks and channels. To emphasize chunks with higher prediction values, suggesting a more confident identification, all predictions are squared before taking the mean. A similar technique called mean exponential pooling is employed in the baseline system provided by [12] this year.
If species composition is known in advance, e.g. based on knowledge about recording habitat or time, removing predictions of unlikely species can further improve identifi-

cation performance. To exploit this fact, predictions of species, not part of data sets in previous years, are removed for corresponding files of the validation and test sets. Other metadata besides year of recording or data set membership is not utilized and none of the provided metadata is used for model training.

Data Augmentation

Various data augmentation techniques are employed to prevent overfitting and improve model accuracy. The following augmentation methods are applied in time domain regarding audio chunks:

- use files from different data sets (Original, *BirdsOnly*, *LowQuality*)
- extract chunks from random position in file (wrap around if end of file is reached and continue reading from beginning)
- apply jitter to duration (ca. ± 0.5 s)
- add 2 audio chunks from random files of the *NoiseOnly* set
- add 2 audio chunks from random files of the *AtmosOnly* set
- add 2 audio chunks from random files belonging to the same bird species
- apply random factor to signal amplitude of all chunks before summation
- apply random cyclic shift
- skip random number of samples (time interval dropout)
- reconstruct audio signal by summing up individual sound elements

Some of the above mentioned techniques were used before in previous BirdCLEF tasks for example adding background noise or sounds from other files belonging to the same bird class with random intensity [6,13,14]. Applying cyclic shift to the sample array by a random amount was also employed by [6] and has the same effect as cutting the audio chunk in two pieces at random position and switching their order [14]. However, some techniques from the above list were not used previously:

Time interval dropout. With a chance of ca. 30 % a random number of samples corresponding to a time interval between zero and the duration of an entire chunk are skipped at random time when reading from the audio file.

Reconstruction via sound elements with dropout. With a chance of ca. 30 % the audio chunk is (re)constructed using the information extracted in the segmentation preprocessing step. Single sound elements are cut from the audio file, multiplied by a short fade in and out envelope, band pass filtered and summed up to recreate the original audio signal. With this procedure it is possible to vary position and length of each sound event individually by adding a small offset to its starting time or by applying a small amount of time stretching to it. Furthermore *sound element dropout* can be implemented by choosing some elements to be skipped and not added to the audio signal sum.

All above described operations are performed in time domain. The final audio chunk is than transformed to frequency domain by applying a short-time Fourier transform. Different FFT parameters are used depending on the sample rate and whether frequencies are chosen to be linear or mel scaled. Normalization and logarithm is applied yielding a dynamic range of approximately 100 dB. Low and high frequencies are removed to get a spectrogram representing a frequency range of about 150 to 10500 Hz. Furthermore the spectrogram is resized to fit the input dimension of the DCNN used for training (e.g. 299x299 pixels for InceptionV3 [15]). Since all networks were pre-trained with RGB images the grayscale image is duplicated to all three color channels. Further augmentation is applied in frequency domain to the spectrogram image:

- pitch shift & frequency stretch by removing additional high and low frequency bands (random number of the first 10 and last 6 rows of the image are cut)
- piecewise time stretch by resizing random number of columns at random position
- piecewise frequency stretch by resizing random number of rows at random position
- use different interpolation filters for resizing
- apply color jitter (brightness, contrast, saturation, hue)

Pitch or frequency shifting and stretching was previously applied in similar ways by e.g. [7] and [16].

Piecewise time stretch. Besides manipulating the duration or speed of an entire audio chunk, time stretching is also applied piecewise with a 50 % chance to change speed at multiple times within a chunk. This is accomplished by dividing the spectrogram image in several vertical pieces, each having a width randomly chosen between 10 and 100 pixel. Then, pieces are resized individually by a factor randomly chosen between 0.9 and 1.1 along the horizontal (time) axis.

Piecewise frequency stretch. The same procedure is applied in an analogous way to the frequency axis with a 40 % chance and a stretch factor between 0.95 and 1.15.

Random choice of interpolation filter. For resizing, the high-quality Lanczos filter of the Python Imaging Library is used by default. However in 15 % of the cases a different resampling filter is chosen from the library with equal chance: Nearest, Box, Bilinear, Hamming and Bicubic.

As a last augmentation step color jitter is applied to the final spectrogram image. This is a very common method during training for image classification (e.g. ImageNet). Brightness, contrast and saturation of the spectrogram image are slightly varied (factor 0.3) as well as hue (factor 0.01). Color jitter was also used by [7].

Table 1 demonstrates the effect of augmentation on model performance. All models in the table are trained using the same parameter settings (if not mentioned otherwise) with a learning rate of 0.01 until performance on the validation set stopped improving (ca. 200 epochs). The first two models in the table show a MRR gain of almost 10 %

for using no augmentation at all compared to applying all above described techniques. For the other models augmentation methods are separately turned off to show their individual influence. The greatest impact on identification performance is gained by adding background noise from the *NoiseOnly* and *AtmosOnly* sets. The least influence is achieved by randomly choosing chunks from the *LowQuality* set leading even to a better MRR score if this method is turned off.

Table 1. Influence of augmentation methods on identification performance.

Settings	MRR [%]
Without augmentation	65.538
Complete augmentation	74.466
Without adding *NoiseOnly* and *AtmosOnly* chunks	67.893
Without adding *AtmosOnly* chunks	72.696
Without adding *NoiseOnly* chunks	73.186
Without piecewise time and frequency stretch	73.681
Without time interval dropout	73.825
Without using *BirdOnly* chunks	73.848
Without reconstruct via sound elements	73.853
Without color jitter	73.984
Without adding same class chunks	74.098
Without duration jitter	74.105
Without using *LowQuality* chunks	74.533

Table 2 shows the impact of using different post-processing methods to get a file-based (or for Subtask2 a file and time interval based) prediction for each species. The default method uses 5 seconds chunks with an overlap of 10 %. Predictions are summarized for each file by simply averaging all chunk predictions. Better identification results are achieved by squaring chunk predictions before taking the mean, using more chunks with larger overlap and removing predictions of species known to be not present in a recording.

Table 2. Influence of post-processing methods on identification performance.

Settings	MRR [%]
Default post-processing	74.466
Predictions squared before averaging	75.788
Pred. squared & 80 % overlap of chunks	75.904
Pred. squared & 80 % overlap & unlikely species excluded	77.540

3 Results

For the first run different models were used for Subtask1 and Subtask2 whereas for runs 2, 3 & 4 the same models or ensemble of models were used for both subtasks. The main properties of models for the first 3 runs are listed in table 3. Results on the official BirdCLEF test sets are summarized in table 4.

Table 3. Main properties of models used for submitted runs 1-3.

Model ID	M1	M2	M3
Included in run	1, 2, 3, 4	1, 2, 3, 4	3, 4
High pass filtered	Yes	No	Yes
Sample rate [Hz]	22050	44100	22050
Number of channels	mono	multi	mono
Chunk duration [s]	5	5	6
Duration jitter [s]	0.45	0.45	0.5
FFT size [samples]	1024	4096	1536
FFT hop size [samples]	256	512	384
Lowest frequency [Hz]	160	160	160
Highest frequency [Hz]	10300	11000	1100
Frequency scaling	Mel	Mel	Mel
Number of mel bands	256	256	310
Network architecture	InceptionV3	InceptionV3	InceptionV3
Image input size [pixel]	299x299	299x299	299x299
Validation fold	7	7	6
Validation MRR [%]	77.63	77.58	76.37

Run 1
A single model was trained for each subtask (M1 for Subtask1, M2 for Subtask2) by fine-tuning pre-trained InceptionV3 networks with mel spectrograms of 5 seconds audio chunks as inputs. Models M1 and M2 mainly differ in the pre-processing of audio files and choice of FFT parameters (see table 3). For Subtask1 pre-filtered, mono audio files were used with a sample rate of 22050 Hz. For Subtask2 audio files were not filtered or mixed down to mono.

Run 2
For the second run an ensemble of the two models from the first run was used for both subtasks by averaging their predictions. Furthermore predictions of new species were removed for recordings belonging to older BirdCLEF data sets.

Run 3
For this run a third model (M3) was added to the ensemble of models from the previous runs. M3 used 6 seconds audio chunks, a larger number of mel bands and slightly different FFT parameters (see table 3). It was trained on a different training/validation split than M1 and M2. Furthermore predictions of the two best snapshots per model (regarding performance on the validation set) were chosen for averaging instead of just one, resulting in 6 predictions per species and file for Subtask1 (or file and time interval for Subtask2). For this and the next run, unlikely species were also removed for older recordings.

Run 4
Again all models and snapshots from the previous runs were reused plus two additional snapshots per model. Furthermore snapshots of 4 models created in an earlier phase were added to the ensemble. Those early models were trained using spectrogram images with linear instead of mel frequency scaling and did not have the later developed piecewise time and frequency stretch augmentation. One of the models was trained by fine-tuning a SENet154 network [17] with input dimension of 224x224 pixel. Although performing worse on the validation set (MRR: 72.7 – 75.8 %) compared to the models used in the first three runs, adding those early models to the ensemble still helped to raise identification performance in both subtasks.

Table 4. Official scores on the BirdCLEF 2018 test sets.

Run	#Models	#Snapshots	MRR Subtask1 [%] (main species only)	MRR Subtask1 [%] (+ background species)	c-mAP Subtask2 [%] (soundscapes)
1	1	1	78.0 (M1)	69.6 (M1)	14.3 (M2)
2	2	2	81.5	72.8	18.1
3	3	6	82.3	73.7	18.7
4	7	18	82.7	74.0	19.3

Figure 2 and 3 compare scores of all submissions to the BirdCLEF 2018 subtasks. The above described approaches provided best systems for both, identifying birds in monophone and soundscape recordings. More results and evaluation details can be accessed via the BirdCLEF 2018 webpage [18] and the crowdAI leaderboards [19], [20].

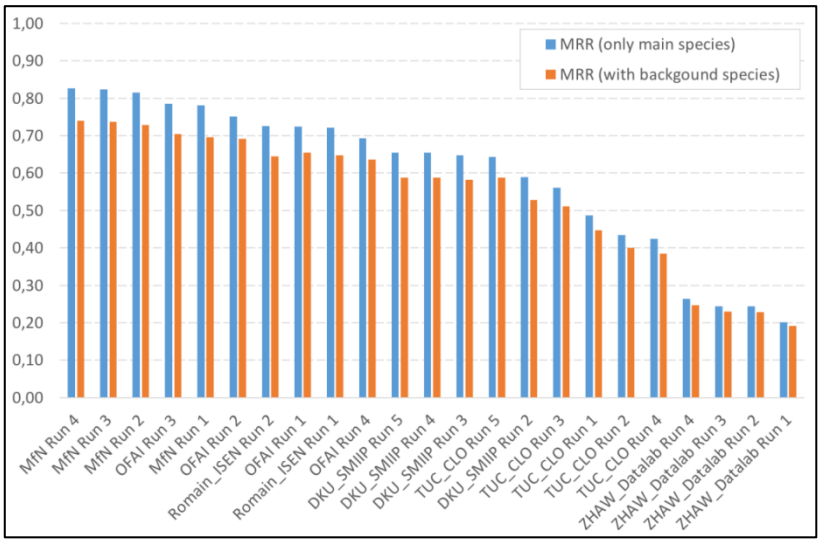

Fig. 1. Official scores of the BirdCLEF 2018 "**Subtask1: monophone recordings**" bird identification task. The above described methods and submitted runs belong to MfN.

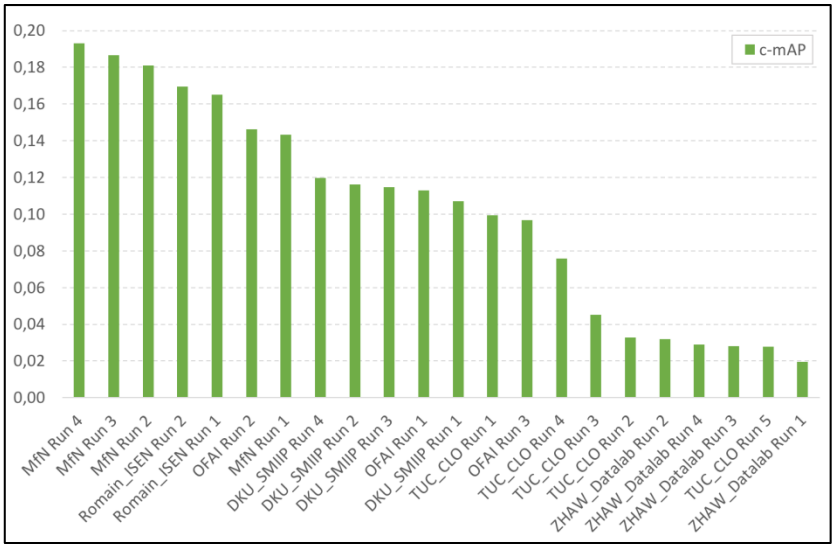

Fig. 2. Official scores of the BirdCLEF 2018 "**Subtask2: soundscape recordings**" bird identification task. The above described methods and submitted runs belong to MfN.

4 Discussion

In the following paragraph some insights are summarized about what worked and what did not work so well during the experiments and processing of the task. Although low frequencies are discarded, pre-process the audio files by applying a soft high pass filter led to slightly better identification results. Also mel spectrograms performed better compared to spectrograms with linear frequency scaling.
The best network for the task seemed to be the InceptionV3 architecture. Other networks were also tested for a few training epochs (ResNet152, DualPathNet92, InceptionV4, DensNet, InceptionResNetV2, Xception, NasNet). But even the more recent or larger architectures that are superior in other image classification tasks could not meet the performance of the InceptionV3 network with attention branch. Of course that doesn't mean that other networks are not potentially able to give better results with the proper tuning.
For model training extracting audio chunks at random position on the fly worked better than using pre-sliced, overlapping chunks. Squaring predictions of chunks before taking the mean was beneficial when summarizing the results per file or time interval.
It proved to be a good decision to create training and validation splits with respect to bird individuals recorded over several files. For Subtask1 the MRR scores were in most cases even better on the test set than on the validation set.
If species composition is known in advance based on knowledge about habitat or recording season and time, removing unlikely species leads to less confusion and significantly helps classification. It would be interesting to see to what extent identification performance can be even further improved if metadata was additionally used and incorporated into the training process.
Augmentation prevents overfitting and strongly improves accuracy of models or model ensembles. A few techniques were adapted from previous work and could be successfully complemented with some new approaches. In an extra experiment it was tried to show their individual influence by turning them off one by one. Another maybe even better approach would have been to start from the model without augmentation and turning them on individually. For example the impact of duration jitter cannot be really verified if piecewise time stretching is also applied.
Some augmentation methods did not work as well as expected like e.g. deliberately degrading the audio quality. Nevertheless it still might be a beneficial training strategy if the model is deployed for bird recognition on mobile or embedded devices where recordings need to be highly compressed because of energy and storage constrains.

Acknowledgments. I would like to thank Hervé Goëau, Alexis Joly, Hervé Glotin and Willem-Pier Vellinga for organizing this task. I especially want to thank Elias Sprengel for sharing his knowledge and the fruitful cooperation during the bird detection challenge [21]. I also want to thank the Museum fuer Naturkunde Berlin and the Bundesministerium fuer Wirtschaft und Energie for supporting my research.

References

1. Xeno-Canto Homepage, https://www.xeno-canto.org/, last accessed 2018/05/29
2. Goëau H, Glotin H, Planqué R, Vellinga WP, Stefan K, Joly A (2018) Overview of BirdCLEF 2018: monophone vs. soundscape bird identification. In: CLEF working notes 2018
3. Joly A, Goëau H, Botella C, Glotin H, Bonnet P, Planqué R, Vellinga WP, Müller H (2018) Overview of LifeCLEF 2018: a large-scale evaluation of species identification and recommendation algorithms in the era of AI, In: Proceedings of CLEF 2018
4. Lasseck M (2015) Towards Automatic Large-Scale Identification of Birds in Audio Recordings, In Lecture Notes in Computer Science Vol.9283: pp 364-375
5. Lasseck M (2015) Improved Automatic Bird Identification through Decision Tree based Feature Selection and Bagging, In: Working notes of CLEF 2015 conference
6. Sprengel E, Jaggi M, Kilcher Y, Hofmann T (2016) Audio based bird species identification using deep learning techniques. In: Working notes of CLEF 2016
7. Sevilla A, Glotin H (2017) Audio bird classification with inception-v4 extended with time and time-frequency attention mechanisms. In: Working Notes of CLEF 2017
8. Lasseck M (2013) Bird Song Classification in Field Recordings: Winning Solution for NIPS4B 2013 Competition. In Proceedings of 'Neural Information Processing Scaled for Bioacoustics: from Neurons to Big Data - NIP4B', joint to NIPS Conf., pages 176-181, http://sabiod.org/NIPS4B2013_book.pdf
9. Lasseck M (2016) Improving Bird Identification using Multiresolution Template Matching and Feature Selection during Training. In: Working Notes of CLEF 2016
10. Paszke A, Gross S, Chintala S et al. (2017) Automatic differentiation in PyTorch. In: NIPS-W
11. McFee B, McVicar M, Balke S et al. (2018) librosa/librosa: 0.6.0 (Version 0.6.0). Zenodo. http://doi.org/10.5281/zenodo.1174893
12. Kahl S, Wilhelm-Stein T, Klinck H et al. (2018) Recognizing Birds from Sound - The 2018 BirdCLEF Baseline System. In: arXiv preprint arXiv:1804.07177
13. Kahl S, Wilhelm-Stein T, Hussein H et al. (2017) Large-Scale Bird Sound Classification using Convolutional Neural Networks. In: Working Notes of CLEF 2017
14. Fazekas B, Schindler A, Lidy T (2017) A Multi-modal Deep Neural Network approach to Bird-song Identication. In: Working Notes of CLEF 2017
15. Szegedy C, Vanhoucke V, Ioffe S (2015) Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567
16. Fritzler A, Koitka S, Friedrich CM (2017) Recognizing Bird Species in Audio Files Using Transfer Learning. In: Working Notes of CLEF 2017
17. Hu J, Shen L, Sun G (2017) Squeeze-and-Excitation Networks. arXiv preprint arXiv:1709.01507
18. BirdCLEF 2018 Homepage, http://www.imageclef.org/node/230, last accessed 2018/05/29
19. crowdAI Subtask1 leaderboard page, https://www.crowdai.org/challenges/lifeclef-2018-bird-monophone/leaderboards, last accessed 2018/05/29
20. crowdAI Subtask2 leaderboard page, https://www.crowdai.org/challenges/lifeclef-2018-bird-soundscape/leaderboards, last accessed 2018/05/29
21. Stowell D, Wood M, Stylianou Y, Glotin H (2016) Bird detection in audio: a survey and a challenge. arXiv preprint arXiv:1608.03417