Recognizing Bird Species in Audio Files Using Transfer Learning

FHDO Biomedical Computer Science Group (BCSG)

Andreas Fritzler[1], Sven Koitka[1,2], and Christoph M. Friedrich[1]

[1] University of Applied Sciences and Arts Dortmund (FHDO)
Department of Computer Science
Emil-Figge-Strasse 42, 44227 Dortmund, Germany
andreas.fritzler@stud.fh-dortmund.de and sven.koitka@fh-dortmund.de and christoph.friedrich@fh-dortmund.de
http://www.inf.fh-dortmund.de
[2] TU Dortmund University
Department of Computer Science
Otto-Hahn-Str. 14, 44227 Dortmund, Germany

Abstract. In this paper, a method to identify bird species in audio recordings is presented. For this purpose, a pre-trained Inception-v3 convolutional neural network was used. The network was fine-tuned on 36,492 audio recordings representing 1,500 bird species in the context of the BirdCLEF 2017 task. Audio records were transformed into spectrograms and further processed by applying bandpass filtering, noise filtering, and silent region removal. For data augmentation purposes, time shifting, time stretching, pitch shifting, and pitch stretching were applied. This paper shows that fine-tuning a pre-trained convolutional neural network performs better than training a neural network from scratch. Domain adaptation from image to audio domain could be successfully applied. The networks' results were evaluated in the BirdCLEF 2017 task and achieved an official mean average precision (MAP) score of 0.567 for traditional records and a MAP score of 0.496 for records with background species on the test dataset.

Keywords: Bird Species Identification · BirdCLEF · Audio · Short Term Fourier Transform · Convolutional Neural Network · Transfer Learning

1 Introduction

Since 2014, a competition called BirdCLEF is hosted every year by the LifeCLEF lab [5]. The LifeCLEF lab is part of the "Conference and Labs of the Evaluation Forum" (CLEF). The goal of the competition is to identify bird species in audio recordings. The difficulty of the competition increases every year. This year, in the BirdCLEF 2017 task [2], 1,500 bird species had to be identified. The training dataset was built from the Xeno-canto collaborative database[3] and consists of 36,492 audio recordings. These records are highly diverse according to sample rate, length, and the quality of their content. The test dataset comprises 13,272 audio recordings.

In 2016, a deep learning approach was applied by [17] to the bird identification task and outperformed other competitors. In this research, a similar method, inspired by the last year's winner is used with an additional extension. Transfer learning [11] is applied by using a pre-trained Inception-v3 [19] convolutional neural network. Related works of identifying bird species in audio recordings in the BirdCLEF 2016 task [3] can be found in [8, 12, 14, 17, 20].

2 Methodology

To solve the BirdCLEF 2017 task, a convolutional neural network on audio spectrograms was used. The main methodology was oriented on the winner [17] of the BirdCLEF 2016 task. The concept of their preprocessing method was partially used. The following sections describe the workflow and parameters in an abstract way, details on the parameters for the runs are given in Section 3.

2.1 Overview

First, the whole BirdCLEF 2017 training dataset was split into two parts. One part consisted of 90% of the training files and was used to train a convolutional neural network and the other part consisted of the remaining 10% and was used to validate on an independent validation set for model selection. For the rest of this paper, the whole BirdCLEF 2017 training dataset shall be referred to as "full training set", the 90% subset shall be referred to as "reduced training set", and the 10% subset shall be referred to as "validation set". The whole pipeline that creates a model that is ready to solve the BirdCLEF 2017 task can be seen in Figure 1.

Next, the audio files were preprocessed. The preprocessing step transforms audio files (.wav, .mp3) to picture files (.png). One audio file typically produces several picture files depending on the length of the audio file and its content.

Then, the generated picture files that were transformed from the reduced training set were used to fine-tune a pre-trained Inception-v3 convolutional neural network. Pre-training was done on the ILSVRC-2012-CLS [15] image classification dataset by the contributors of Tensorflow Slim model repository, and a checkpoint file of the model was provided[4]. By using the provided checkpoint, the models' knowledge was transferred to the BirdCLEF 2017 task. For fine-tuning, Tensorflow Slim[5] version 1.0.1 was used. For each picture, an adapted

[3] http://www.xeno-canto.org/ (last access: 31.05.2017)

[4] http://download.tensorflow.org/models/inception_v3_2016_08_28.tar.gz (last access: 27.03.2017)

[5] https://github.com/tensorflow/models/tree/master/slim (last access: 23.05.2017)

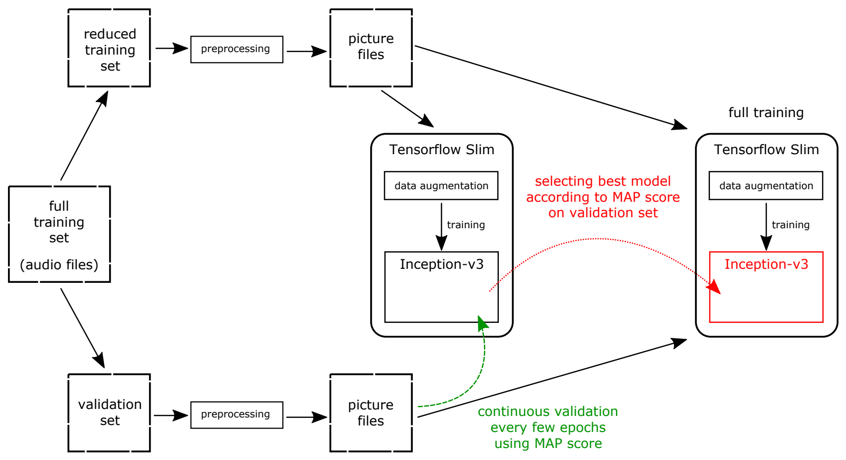

Fig. 1: Visualization of the model creation pipeline.

data augmentation was applied that includes time shifting, time stretching using factors in the range $[0.85, 1.15)$, pitch shifting, and pitch stretching using percentages in the set $\{0, \ldots, 8\}$.

The whole training was done in three phases. In the first phase, the top layers of the pre-trained model were deleted[6] and trained from scratch leaving the rest of the model fixed. The reason for this is to adjust the number of output classes from the pre-trained network with 1,000 classes to 1,500 species. Afterward, the second phase was started, and the whole model was fine-tuned including all trainable weights. Throughout the whole training during the second phase snapshots of the model were validated every few epochs with pictures that were transformed from the validation set. This way the models' progress according to the MAP score was monitored. It was done to recognize overfitting. After the second phase, a snapshot with the best-monitored MAP score was selected for a third training phase. In this phase, image files from the full training set were used to fine-tune the model further. When the third step was finished, the model was ready to classify test files.

Finally, the BirdCLEF 2017 test dataset was preprocessed in a similar but not an identical manner as the full training dataset. Details are described later in this Section. During preprocessing, every audio file was transformed into many picture files. In the prediction phase, a fixed region was cropped from the center of every picture file and was predicted by the fully trained model. The predictions were combined by averaging all image segments per audio file for final results. In addition, time-coded soundscapes were grouped in ranges of 5 seconds. The predictions were ordered in descending order per audio file. Furthermore, predictions in time-coded soundscapes were ordered per 5-second regions. In the end, a result file was generated.

[6] scopes InceptionV3/Logits and InceptionV3/AuxLogits

2.2 Preprocessing for Training

The progress of the following described preprocessing steps can be seen in Figure 2.

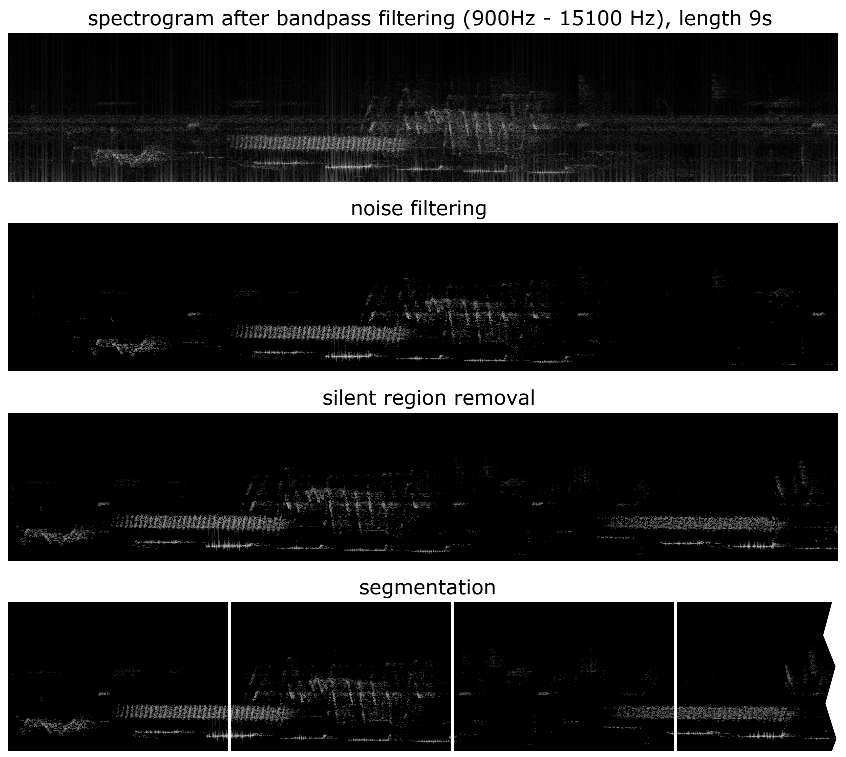

Fig. 2: Visualization of the preprocessing pipeline. The STFT spectrograms were logarithmized for better visualization.

Extracting Frequency Domain Representation A frequency domain representation was generated for all of the audio files using Short-Term Fourier Transform (STFT) [1]. For this purpose, a Java library "Open Intelligent Multimedia Analysis for Java" (OpenIMAJ)[7] [4] version 1.3.5 was used. It is available under the New BSD License, and it is able to process .wav and also .mp3 audio files. Unfortunately, OpenIMAJ does not support sample overlapping in an easy way by itself, so it had to be implemented. Furthermore, it seems OpenIMAJ is not capable of processing audio files with a bit depth of 24 bits. Two time-coded

[7] http://openimaj.org/ (last access: 20.05.2017)

soundscape audio files[8] in the test dataset were converted from a bit depth of 24 bits to 16 bits with the python library "librosa" version 0.5.0 [9], that is available[9] under the ISC License.

Audio files in BirdCLEF 2017 datasets have different sample rates thus the window size (amount of samples) that was used for the STFT depended on the file's sample rate. For a sample rate of 44.1 kHz, a length of 512 samples was used to create a slice of 256 frequency bands (later on the vertical axis of an image). One slice represents a time interval of approximately 11.6 ms. For a file with a different sample rate, the size of the window was adjusted to match the time interval of 11.6 ms. Audio files were padded with zeros if their last window had fewer samples than were needed for the transform.

The extracted frequency domain representation is a matrix. Its elements were normalized to the range $[0, 1]$. Every element of this matrix represents a pixel in the exported image. The logarithm of the elements was not taken, but instead, the values were processed in a linear manner. The matrix was further processed using different methods to remove unnecessary information to reduce its size.

Bandpass filtering A frequency histogram of the full training set is shown in Figure 3. Most of the frequencies below 500 Hz are dominated by noises, for example, wind or mechanical vibration. This circumstance explains the peak in the lower frequency range. It was determined by manually examining 20 files that were randomly selected from the full training set.

One previous work [10] removed frequencies under 1 kHz. Audio recordings were in 16 kHz PCM format. The authors in [20] participated in the BirdCLEF 2016 task and used a low-pass filter with a cutoff frequency of 6,250 Hz.

In this research, a lower frequency limit of 1,000 Hz and an upper frequency limit of 12,025 Hz was used for bandpass filtering. This reduced the 256 frequency bands by half to 128 bands.

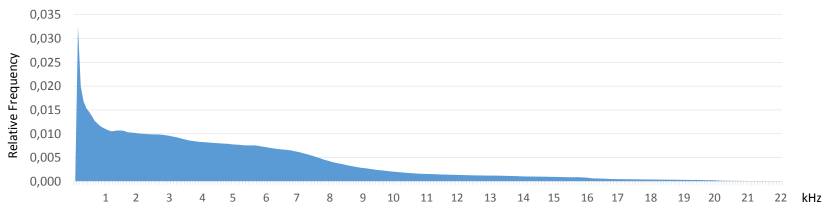

Fig. 3: Frequency histogram of the full BirdCLEF 2017 training dataset.

[8] LIFECLEF2017_BIRD_HD_SOUNDSCAPE_WAV_RN49908.wav and LIFECLEF2017_BIRD_HD_SOUNDSCAPE_WAV_RN49909.wav

[9] https://github.com/librosa/librosa (last access: 01.06.2017)

Noise Filtering Median Clipping was applied to reduce noise like wind blowing. This method was also used by the winner [17] of BirdCLEF 2016 task and formerly by [7]. It selects all of the elements in the matrix whose values are three times bigger than their corresponding row (frequency band) median and three times larger than their corresponding column (time frame) median. The other elements are set to zero. Afterward, tiny objects were removed. If all of the 8 neighbor elements of an element were zeros, then the element itself was also set to zero.

Silent Region Removal The authors in [17] used signal to noise separation to extract bird calls from audio files. In this research, regions with less information were deleted to retain bird calls in the following way. If the average of a fixed region did not reach a threshold, then the region was removed. Every column was examined on its own. In every column, the number of non-zero elements was counted and normalized by the total number of elements in each column. For this procedure, a threshold of 0.01 was used. After this step, the resulting matrix could have just a few or even zero columns.

In the end, if the resulting matrix had less than 32 columns, the audio file was completely discarded from training.

Exporting Image Files Images were exported using a fixed resolution. If after the previous processing steps a matrix had fewer columns than the defined target width of a picture then the matrix was padded to the desired amount of columns and its available content was looped into the padded area.

The completely processed frequency representation was segmented into equal-sized pieces of a fixed length and a predefined overlapping factor. The matrices' elements were in the range $[0, 1]$ and were scaled by a constant factor as well as clamped to the maximum value of 255. The elements were used for all of the three channels in the final picture. As a result, the three channels contained the same information.

2.3 Preprocessing for Prediction

During the preprocessing of the BirdCLEF 2017 test dataset, one exception was made to time-coded soundscapes. On these files, *silent region removal* was not applied to preserve their full length. Furthermore, no audio files were discarded if they had less than 32 columns in their matrix.

2.4 Data Augmentation

Due to the input dimension of Inception-v3 (299x299x3) the generated picture files were processed at this stage before they were forwarded to train the model. This was done by cropping a region from the original image. First, a target cropping location was computed with a jitter for the vertical axis (random y offset). Next, time shifting was applied by moving the starting x position randomly along the x-axis. Then, time stretching was used by moving the target width by a random factor in the range $[0.85, 1.15)$. After that, pitch shifting was combined with pitch stretching and was calculated by moving the starting y position randomly. The target height was reduced randomly the same way. The maximum amount of pitch stretch was 8% in total. The calculated region was cropped from the original picture and was scaled with bilinear interpolation to a size of 299x299 pixels on all of the 3 channels (red, green, blue) to match the input dimension of Inception-v3. Figure 4 shows this procedure visually.

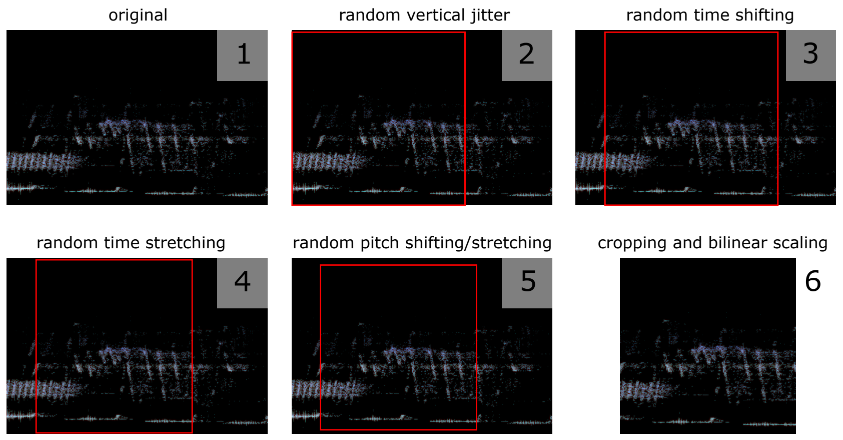

Fig. 4: Visualization of the real-time data augmentation pipeline during training.

3 Run Details and Results

Although more recent network architectures exist like Inception-v4 [18] and Inception-ResNet-v2 [18] which might improve the results in comparison to Inception-v3, the former ones were not used for this research because they are slower than the Inception-v3. The former ones are also available as pre-trained models[10] and are potential candidates for future work.

Four runs were submitted in total. Three runs used slightly different methods of preprocessing, and the fourth run combined the results of the former three runs by averaging them.

[10] http://download.tensorflow.org/models/inception_v4_2016_09_09.tar.gz (last access: 28.05.2017) and http://download.tensorflow.org/models/inception_resnet_v2_2016_08_30.tar.gz (last access: 28.05.2017)

First, *binary run (Run 2)* was created with the preprocessing pipeline (compare Section 2.2) and binary images. Next, *grayscale run (Run 4)* was created with a few changes to *binary run (Run 2)* to examine the differences in MAP scores in comparison to *binary run*. Lastly, *big run (Run 1)* was designed by improving some parts of the previous runs and correcting some mistakes. The runs were submitted in alphabetical order according to their description names thus the run's details in this Section does not follow the run's number but rather their temporal creation time.

Training was done on one NVIDIA Tesla K80 graphics card that contains 2 GPUs with 12 GB of RAM each. A mini-batch size of 32 was used per GPU, which results in an effective batch size of 64. Fine-tuning of a single model until the stage of prediction took several days. The machine was used non-exclusively. Predicting was done on one NVIDIA Titan X Pascal GPU.

Table 1 shows the runs' achieved results measured in MAP score on the reduced training set and the validation set using all predictions. To show the advantages of transfer learning, all of the runs were executed twice with identical parameters. On the one hand a pre-trained Inception-v3 was used, and on the other hand, the Inception-v3 was trained from scratch. Results in Table 1 show that fine-tuning a pre-trained convolutional neural network performs better than training a neural network from scratch, although pre-training was done on another domain. In addition, official results on the BirdCLEF 2017 test dataset of the submitted runs are stated as well.

Table 1: Achieved results measured in MAP

	BirdCLEF 2017 training dataset				**BirdCLEF 2017 test dataset official results**			
	Inception-v3 **from scratch**		**pre-trained** Inception-v3		**pre-trained** Inception-v3			
	Reduced training set (90% subset)	Validation set (10% subset)	Reduced training set (90% subset)	Validation set (10% subset)	Soundscapes with time-codes	Soundscapes without time-codes (same queries 2016)	Traditional Records (only main species)	Traditional Records (with background species)
Binary Run (Run 2)	0.627	0.415	0.815	0.487	0.069	**0.048**	0.491	0.431
Grayscale Run (Run 4)	0.490	0.303	0.928	0.541	0.083	0.023	0.504	0.438
Big Run (Run 1)	0.415	0.333	0.832	0.531	0.056	0.041	0.492	0.427
Combined Run (Run 3)	**0.672**	**0.455**	**0.932**	**0.598**	**0.097**	0.039	**0.567**	**0.496**

3.1 Binary Run: Run 2

The following Section describes only additions and differences compared to the description in Section 2.

Preprocessing STFT used 512 samples without sample overlapping. After the step *noise filtering*, all of the elements in the matrix greater than 0 were set to 1 to create a monochrome picture file. After *silent region removal*, 45 audio files were discarded from training.

Images were exported using a resolution of 256 pixels in width and 128 pixels in height. One image file represents a length of 2.97 s. For this purpose, the previously generated matrices were segmented into equal-sized fragments of 256 pixels in width with an overlapping factor of $\frac{7}{8}$. Before matrices were exported to pictures, their elements were multiplied by 255. The resulting values were used for all of the three channels in a picture. The reduced training set led to 1,365,849 picture files (2.5 GiB). From the validation set, 145,724 image files were generated (282.6 MiB). The test dataset produced 1,583,771 picture files (2.66 GiB).

Training and Data Augmentation Learning rates were fixed in this run. The top layers of Inception-v3 were trained for 1.48 epochs with a learning rate of 0.01. Training on the reduced training set was done for 15.8 epochs with a learning rate of 0.0002. A MAP score of 0.487 was achieved on the validation set. After that, the full training set was used for training for another 4.28 epochs with a learning rate of 0.0002.

During data augmentation, a region of 128 pixels in width ($\pm15\%$) and 128 pixels in height (-8%) should have been randomly cropped.

Predicting In the predicting phase, a region of 128x128 pixels was cropped from the center of every picture file. The cropped length of 128 pixels corresponds to a time interval of 1.49 s.

Mistakes In this run, data augmentation was implemented incorrectly. No randomness was used. When training was started then the parameters for time shifting, time stretching, and pitch shifting were generated in a random manner, but these values were always the same as long as training was not restarted.

The model reached a phase of overfitting. Because the best checkpoint according to MAP score was not saved, an overfitted version of the model was used to complete the BirdCLEF task. The best-monitored MAP score of the lost checkpoint was 0.511 after 8 epochs of training.

3.2 Grayscale Run: Run 4

This run was almost the same as *binary run (Run 2)*. Here only differences to *binary run (Run 2)* are described.

Preprocessing In the preprocessing step, there were only two differences compared to *binary run (Run 2)*. First, the frequency domain representation in the range $[0, 1]$ was used without being transformed into zeros and ones. Second, before image files were exported, the elements of the matrices were multiplied by 2,000 and cut off at value 255. This led to picture files that contained grayscale information. Everything else in the preprocessing pipeline was left unchanged. The number of files compared to *binary run (Run 2)* had not changed, but the file size had increased. The reduced training set had a size of 7.4 GiB, the validation set consisted of 812 MiB, and the test set counted 7.25 GiB.

Training and Data Augmentation The top layers of Inception-v3 were trained for 1.74 epochs with a fixed learning rate of 0.02. Afterward, all layers were trained using an exponential learning rate. The learning rate descended smoothly. A staircase function was not used. As training had started, the learning rate had a value of 0.005. After 5.4 epochs, the learning rate reached a value of 0.0003, and a MAP score of 0.541 was achieved on the validation set. Unfortunately, training was restarted every few epochs to slightly adjust the learning rate. Afterward, training was started on the full training set for another 2.6 epochs with an exponential learning rate, starting at 0.0002 and ending at 0.0001.

Mistakes The same mistakes as they were made in the *binary run (Run 2)* were also made in this run. Data augmentation was not working properly. This led to an overfitted model after 6 epochs of training. Training was restarted every few epochs to correct the learning rate. As a side effect, the model was trained on more different pictures than the model in the *binary run (Run 2)*.

3.3 Big Run: Run 1

The name big run is derived from the size of pictures that were generated in the preprocessing step. Pictures were created by processing each channel (red, green, blue) differently. After 7 epochs of fine-tuning, this model had a MAP score of 0.531. Due to the deadline of the BirdCLEF 2017 task, this model could not be trained completely as planned. One can assume that if this model was trained for more epochs, the MAP score should become a little bit better because data augmentation mistakes from the previously made models were corrected.

Preprocessing STFT used a window size of 942 samples. A slice of 471 frequency bands was generated this way. This slice represents a time interval of approximately 21.4 ms. Furthermore, sample overlapping of 75% was used.

Bandpass filtering used a lower frequency limit of 900 Hz and an upper frequency limit of 15,100 Hz. This reduced the 471 frequency bands to 303 bands.

Before the method described in *silent region removal* was applied, two other processing steps were executed. First, all of the elements in the first 50 columns (approximately 0.27 s) were examined. That means the arithmetic mean of that region was calculated. If the calculated value did not reach a threshold of 0.0001, then the whole region was discarded. Otherwise, the region to be examined was shifted with 75% overlapping. This was repeated throughout the whole matrix. Very silent regions of an audio signal were deleted this way. Second, every column was examined on its own. If the arithmetic mean of a column did not reach a threshold of 0.0001, then the column was removed using a special treatment. Up to three sequenced columns may have each an average value below the threshold. These columns were not deleted. Up to three following columns were set to zero if each of their averages was also below the threshold. All subsequent columns each with an average below the threshold were removed. This procedure separated parts with much audio information visually even more from each other while quiet frames were deleted. After these two steps, the process described in *silent region removal* was applied. In the end, 7 audio files were discarded from training.

Images were exported using a resolution of 450 pixels in width and 303 pixels in height. The width of 450 pixels represents a length of approximately 2.4 s.

The completely processed frequency representation was segmented into equal-sized pieces with a length of 450 columns and an overlapping factor of $\frac{2}{3}$. The matrices' were multiplied by 1,000 and then cut off at 255. The result was copied to three matrices. Each matrix represents a color channel of the final picture. One matrix (red channel) was blurred using Gaussian blur [16] with a radius of 4. Another matrix (blue channel) was sharpened using CLAHE algorithm [13]. A block radius of 10 and 32 bins were used. The third matrix (green channel) was left untouched. An example of the three differently processed channels is shown in Figure 5.

The reduced training set was transformed into 816,421 image files (23.3 GiB), the validation set has produced 87,448 image files (2.5 GiB), and the test set was converted to 932,573 images (24.4 GiB).

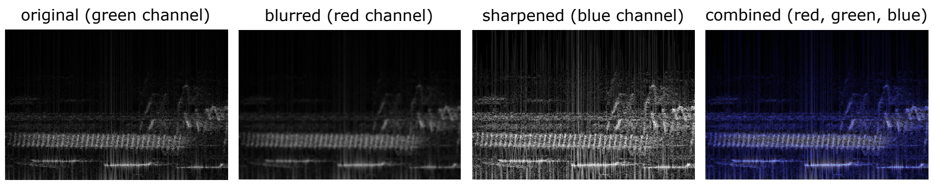

Fig. 5: Visualization of the generated channels as well as the final composed image. For better visualization the spectrogram was not preprocessed.

Data Augmentation A target cropping location was computed with a jitter of 4 pixels ($\varDelta_y \in \{0, \ldots, 4\}$). At this point, the target region had a shape of 299x299 pixels. Time stretching manipulated the target width. Pitch shifting and pitch stretching were applied by moving the starting y position randomly by 0, 3, 6, 9, or 12 pixels (that corresponds to percentages in the set $\{0, \ldots, 4\}$). Target height was manipulated the same way.

Training During the first phase of training, a learning rate of 0.02 was used for 1 epoch, and a rate of 0.01 was used for a second epoch. After that, the second phase was started with a learning rate of 0.0008. In the second phase, the learning rate was exponentially decreased by a staircase function. That means the rate was adjusted after every epoch was fully completed. A learning rate decay value of 0.7 for every completed epoch was used. After 7 epochs, the model reached a learning rate of 0.000066. A MAP score of 0.531 was achieved on the validation set. The third phase was started using a fixed learning rate of 0.0002 for another 1.98 epochs.

Predicting In the prediction phase, a region of 299x299 pixels was cropped from the center of every picture file and was predicted by the fully trained model. 299 pixels represent a length of 1.6 s.

3.4 Combined Run: Run 3

Two different methods of combining predictions [6] were tried in every run when predictions of picture files were combined to create a prediction of an audio file. Calculating the arithmetic mean was one method. The other method was majority voting. This can be explained in the following way: a prediction of a picture is an expert. One asks all of the experts of an audio file to vote for a single target class. The class with the maximum number of votes is the predicted class. Calculating the arithmetic mean always performed better. Its MAP score had a relative difference of 1%–10% compared to the MAP score of majority voting.

Run 3 had not a separate model that was used to predict test audio files but rather the predictions of the test dataset of the other three runs were combined. This was done by averaging the predictions of every single picture file that belongs to one audio file. The combination of results of every model after the second training phase led to a MAP score of 0.598.

4 Conclusion and Future Work

An approach to identify bird species in audio recordings was shown. For this purpose, a preprocessing pipeline was created and a pre-trained Inception-v3 convolutional neural network was fine-tuned. It could be shown that fine-tuning a pre-trained convolutional neural network leads to better results than training a neural network from scratch. It is remarkable, that this type of transfer learning is even working from the image to the audio domain.

Unfortunately, the error-free model was not trained long enough to show its full potential. The models presented in this paper reached fair results in the context of the competition and leave room for improvement. A possible enhancement concerns the preprocessing pipeline and data augmentation. Future works should consider transferring the preprocessed frequency domain representation to a convolutional neural network avoiding the use of picture files.

Furthermore, this research has not focused on identifying bird species in soundscapes. The winner team of the BirdCLEF 2016 task has extracted noisy parts from audio files and mixed them into other audio files. Additionally, a sound effects library with many different ambient noises recorded in nature could be used. This could increase the diversity of the training files during the phase of data augmentation further. This approach was not implemented in this research due to time limitations.

Acknowledgement

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU which supported this research.

References

1. Allen, J.B.: Short term spectral analysis, synthesis, and modification by discrete fourier transform. IEEE Transactions on Acoustics, Speech, Signal Processing, vol. ASSP-25 pp. 235–238 (1977)
2. Goëau, H., Glotin, H., Planqué, R., Vellinga, W.P., Joly, A.: LifeCLEF bird identification task 2017. In: Working Notes of CLEF 2017 - Conference and Labs of the Evaluation forum, Dublin, Ireland, 11-14 September, 2017. (2017)
3. Goëau, H., Glotin, H., Vellinga, W.P., Planqué, R., Joly, A.: LifeCLEF bird identification task 2016: The arrival of deep learning. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. CEUR-WS Proceedings Notes, vol. 1609, pp. 440–449 (2016)
4. Hare, J.S., Samangooei, S., Dupplaw, D.P.: OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images. In: Proceedings of the 19th ACM international conference on Multimedia (MM 2011). pp. 691–694 (2011)
5. Joly, Alexis and Goëau, Hervé and Glotin, Hervé and Spampinato, Concetto and Bonnet, Pierre and Vellinga, Willem-Pier and Lombardo, Jean-Christophe and Planqué, Robert and Palazzo, Simone and Müller, Henning: LifeCLEF 2017 lab overview: multimedia species identification challenges. In: Proceedings of CLEF 2017 (2017)
6. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms, 2nd Edition. Wiley (2014)
7. Lasseck, M.: Bird song classification in field recordings: Winning solution for NIPS4B 2013 competition. Proc. of int. symp. Neural Information Scaled for Bioacoustics, sabiod.org/nips4b, joint to NIPS pp. 176–181 (2013)
8. Lasseck, M.: Improving bird identification using multiresolution template matching and feature selection during training. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. CEUR-WS Proceedings Notes, vol. 1609, pp. 490–501 (2016)

9. McFee, B., McVicar, M., Nieto, O., Balke, S., Thome, C., Liang, D., Battenberg, E., Moore, J., Bittner, R., Yamamoto, R., Ellis, D., Stoter, F.R., Repetto, D., Waloschek, S., Carr, C., Kranzler, S., Choi, K., Viktorin, P., Santos, J.F., Holovaty, A., Pimenta, W., Lee, H.: librosa 0.5.0 (feb 2017), https://doi.org/10.5281/zenodo.293021
10. Neal, L., Briggs, F., Raich, R., Fern, X.Z.: Time-frequency segmentation of bird song in noisy acoustic environments. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011). pp. 2012–2015 (2011)
11. Oquab, M., Bottou, L., Laptev, Ivan, S., Josef: Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014). pp. 1717–1724 (2014)
12. Piczak, K.J.: Recognizing bird species in audio recordings using deep convolutional neural networks. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. CEUR-WS Proceedings Notes, vol. 1609, pp. 534–543 (2016)
13. Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., Haar Romeny, B.t., Zimmerman, J.B., Zuiderveld, K.: Adaptive histogram equalization and its variations. Computer Vision, Graphics and Image Processing, vol. 39 pp. 355–368 (1987)
14. Ricard, J., Glotin, H.: Bag of MFCC-based words for bird identification. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. CEUR-WS Proceedings Notes, vol. 1609, pp. 544–546 (2016)
15. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. International Journal of Computer Vision 115(3), 211–252 (2015)
16. Shapiro, L.G., Stockman, G.C.: Computer Vision. Prentice Hall (2001)
17. Sprengel, E., Jaggi, M., Kilcher, Y., Hofmann, T.: Audio based bird species identification using deep learning techniques. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. CEUR-WS Proceedings Notes, vol. 1609, pp. 547–559 (2016)
18. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: Proceedings of the International Conference on Learning Representations Workshop (ICLR 2016) (2016)
19. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016). pp. 2818–2826 (2016), https://arxiv.org/abs/1512.00567v3
20. Tóth, B.P., Czeba, B.: Convolutional neural networks for large-scale bird song classification in noisy environment. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. CEUR-WS Proceedings Notes, vol. 1609, pp. 560–568 (2016)