Audio Based Bird Species Identification using Deep Learning Techniques

Elias Sprengel, Martin Jaggi, Yannic Kilcher, and Thomas Hofmann

Eidgenössische Technische Hochschule (ETH) Zürich,
Rämistrasse 101, 8092 Zürich, Switzerland
elias.sprengel@alumni.ethz.ch,
{jaggi,yannic.kilcher,thomas.hofmann}@inf.ethz.ch
http://www.ethz.ch

Abstract. In this paper we present a new audio classification method for bird species identification. Whereas most approaches apply nearest neighbour matching [6] or decision trees [8] using extracted templates for each bird species, ours draws upon techniques from speech recognition and recent advances in the domain of deep learning. With novel preprocessing and data augmentation methods, we train a convolutional neural network on the biggest publicly available dataset [5]. Our network architecture achieves a mean average precision score of 0.686 when predicting the main species of each sound file and scores 0.555 when background species are used as additional prediction targets. As this performance surpasses current state of the art results, our approach won this years international BirdCLEF 2016 Recognition Challenge [3,4,1].

Keywords: Bird Identification, Deep Learning, Convolution Neural Network, Audio Processing, Data Augmentation, Bird Species Recognition, Acoustic classification

1 Introduction

1.1 Motivation

Large scale, accurate bird recognition is essential for avian biodiversity conservation. It helps us quantify the impact of land use and land management on bird species and is fundamental for bird watchers, conservation organizations, park rangers, ecology consultants, and ornithologists all over the world. Many books have been published [10,2,11] to help humans determine the correct species and dedicated online forums exist where recordings can be shared and discussed [15]. Nevertheless, because recordings, spanning hundreds of hours, need to be carefully analysed and categorised, large scale bird identification remains almost an impossible task to be done manually. It, therefore, seems natural to look at ways to automate the process. Unfortunately a number of challenges have made this task extremely difficult to tackle. Most prominent are:

– Background noise

– Multiple birds singing at the same time (multi-label)
– Difference between mating calls and songs
– Inter-species variance [9]
– Variable length of sound recordings
– Large number of different species

Because of these, most systems are developed to deal with only a small number of species and require a lot of re-training and fine-tuning for each new species. In this paper, we describe a fully automatic, robust machine learning method that is able to overcome these issues. We evaluated our method on the biggest publicly available dataset which contains over 33'000 recordings of 999 different species. We achieved a mean average precision (MAP) score of 0.69 and an accuracy score of 0.58 which is currently the highest recorded score. Consequently our approach won the international BirdCLEF 2016 Recognition Challenge [3,4,1].

1.2 Approach

We use a convolutional neural network with five convolutional and one dense layer. Every convolutional layer uses a rectify activation function and is followed by a max-pooling layer. For preprocessing, we split the sound file into a signal part where bird songs/calls are audible and a noise part where no bird is singing/calling (background noise is still present in these parts). We compute the spectrograms (Short Time Fourier Transform) of both parts and split each spectrogram into equally sized chunks. Each chunk can be seen as the spectrogram of a short time interval (typically around 3 seconds). As such, we can use each chunk from the signal part as a unique training/testing sample for our neural network. A detailed description of every step will be provided in the next chapters.

Figure 1 and Figure 2 give an overview of our training / testing pipeline.

2 Feature Generation

The generation of good input features is vital to the success of the neural network. There are three main stages. First, we decide which parts of the sound file correspond to a bird singing/calling (signal parts) and which parts contain noise or silence (noise parts). Second, we compute the spectrogram for both signal and noise part. Third, we divide the spectrogram of each part into equally sized chunks. We can then use each chunk from the signal spectrogram as a unique sample for training/testing and augment it with a chunk from the noise spectrogram.

2.1 Signal/Noise Separation

To divide the sound file into a signal and a noise part, we first compute the spectrogram of the whole file. Note that all spectrograms in this paper are computed in the same way. First the signal is passed through a short-time Fourier

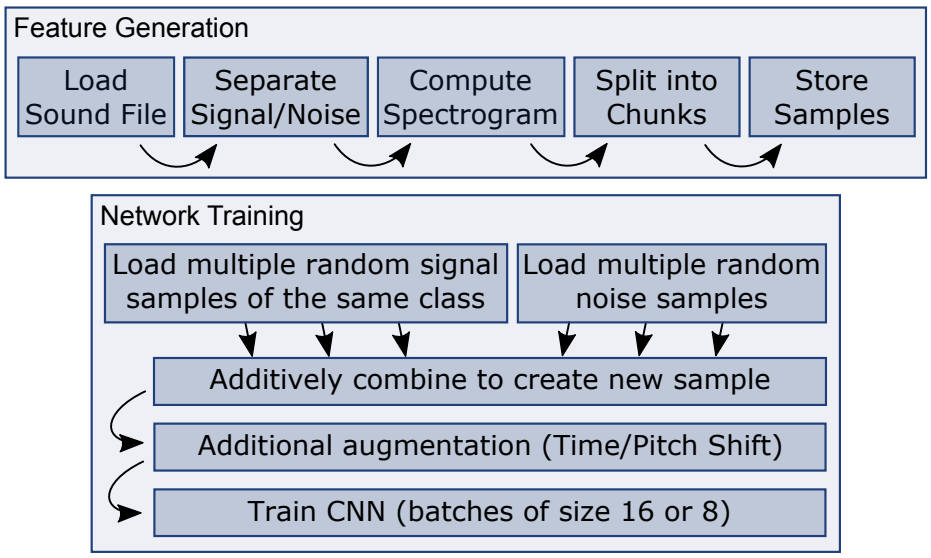

Fig. 1: Overview of the pipeline for training the neural network. CNN stands for convolutional neural network. During training, we use a batch size of 16 training examples per iteration. However, due to memory limitations of the GPU, we sometimes have to fall back to batches of size 8.

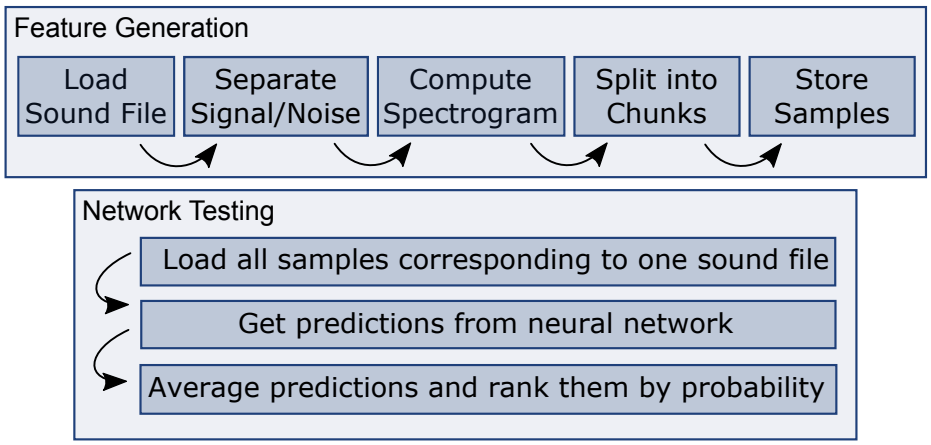

Fig. 2: Overview of the testing pipeline. Note that we get multiple predictions per sound file (one prediction per chunk/sample) which we can average to obtain a single prediction per file.

transform (STFT), this is done using a Hanning window function (size 512, 75% overlap). Then the logarithm of the amplitude of the STFT is taken. However, the signal/noise separation is the exception to this rule because here, we do not take the logarithm of the amplitude but instead divide every element by the maximum value, such that all values end up in the interval $[0, 1]$. With the spectrogram at hand, we are now able to look for the signal/noise intervals.

For the signal part we follow [7] quite closely. We first select all pixels in the spectrogram that are three times bigger than the row median and three times bigger than the column median. Intuitively, this gives us all the important parts of the spectrograms, because a high amplitude usually corresponds to a bird singing/calling. We set these pixels to 1 and everything else to 0. We apply a binary erosion and dilation filter to get rid of the noise and join segments. Experimentally we found that a 4 by 4 filter produced the best results. We create a new indicator vector which has as many elements as there are columns in the spectrogram. The i-th element in this vector is set to 1 if the i-th column contains at least one 1, otherwise it is set to 0. We smooth the indicator vector by applying two more binary dilation filters (filter size 4 by 1). Finally we scale our indicator vector to the length of the original sound file. We can now use it as a mask to extract the signal part. Figure 3 shows a visual representation of each step.

For the noise part we follow the same steps but instead of selecting the pixels which are three times bigger than row and column median, we select all pixels which are 2.5 times bigger than the row and column median. We then proceed as described above but invert the result at the very end. Note that, by construction of our algorithm, a single column should never belong to both signal and noise part. On the other hand, it can happen that a column is not part of either noise nor signal part because we use different thresholds (3 versus 2.5). This is intended as it provides a safety margin for our selection process. The reasoning is that everything that was not selected as either signal nor noise, provides almost no information to the neural network. The bird is either barely audible/distorted or the sound does not match our concept of background noise very well.

The signal and noise masks split the sound file into many short intervals. We simply join these intervals together to form one signal- and one noise-sound-file. Everything that is not selected is disregarded and not used in any future steps. The transition marks, that occur when two segments are joined together, are usually not audible because the cuts happen when no bird is calling/singing. Furthermore, the use of the dilation filters, as described earlier, ensures that we keep the number of generated intervals to a minimum when applying the masks. From the two resulting sound files we can now compute a spectrogram for both signal and noise part. Figure 4 shows an example.

2.2 Dividing the Spectrograms into Chunks

As described in the last section, we compute a spectrogram for both the signal and noise part of the sound file. Afterwards we split both spectrograms into chunks of equal size (we use a length of 512). The splitting is done for three

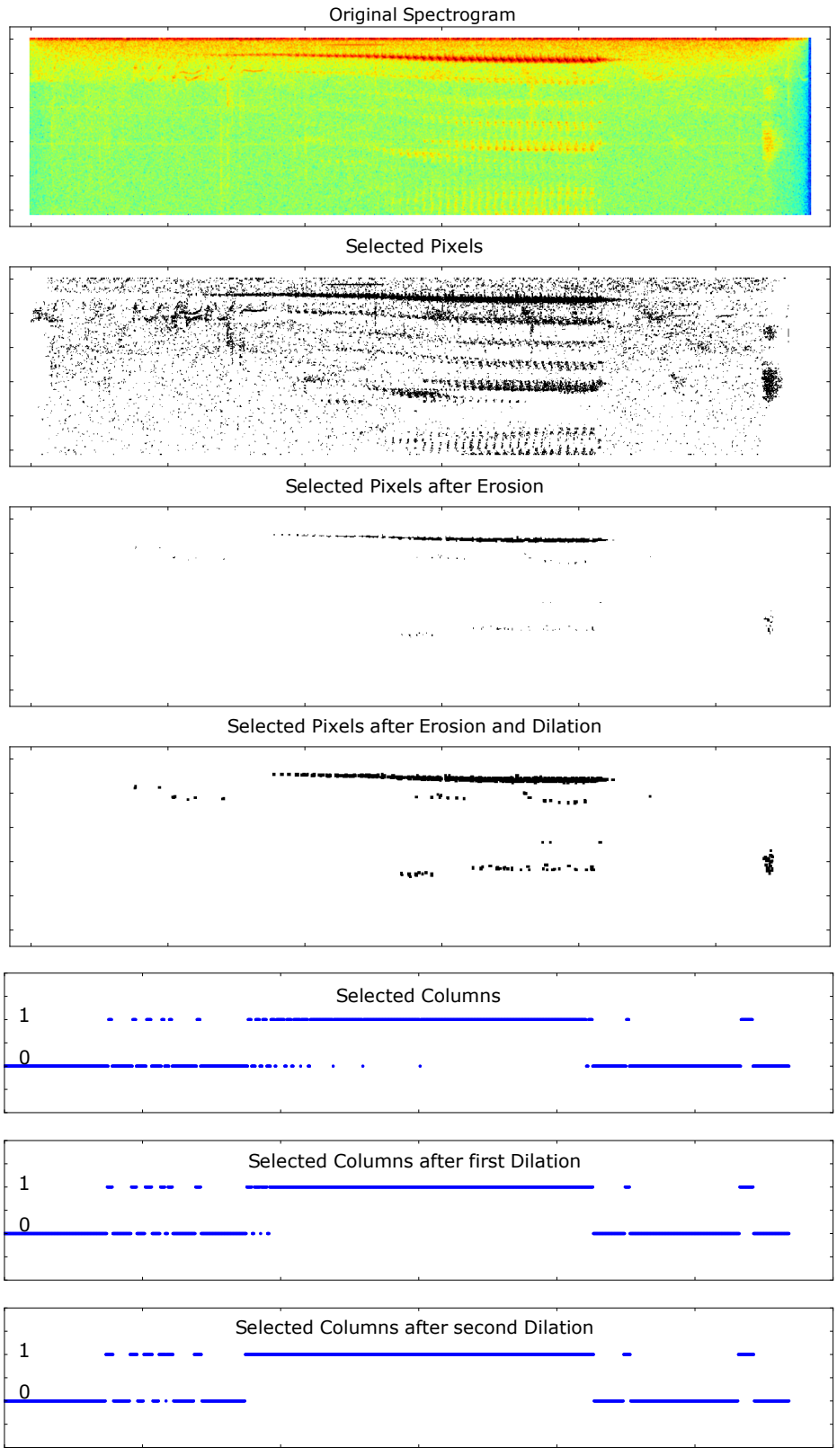

Fig. 3: Detection of signal parts for the file LIFECLEF2014_BIRDAMAZON_XC_WAV_RN3508. The two dilation steps at the end are important because they end up improving the smoothness of our mask/signal part.

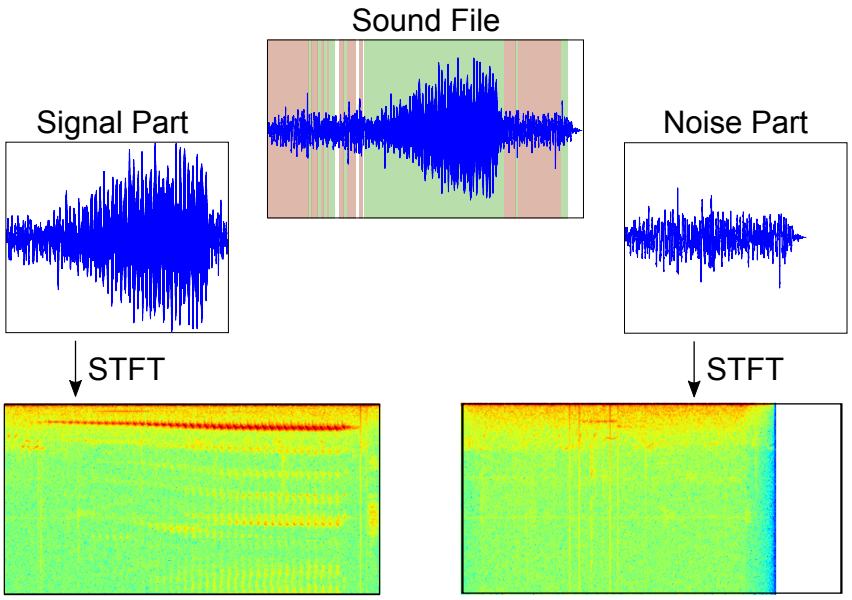

Fig. 4: Separation of signal and noise part for the sound file IFE-CLEF2014_BIRDAMAZON_XC_WAV_RN3508. The green color in the sound file image corresponds to the signal part, the red color to the noise part. Everything that has a white background was not considered as either signal nor noise and got discarded.

reasons. For one, we need a fixed sized input for our neural network architecture. We could pad the input but the large variance in the length of the recordings would mean that some samples would contain over 99% padding. We could also try to use varying step sizes of our pooling layers but this would stretch or compress the signal in the time dimension. In comparison, chunks allow us to pad only the last part and keep our step size constant. Second, thanks to our signal/noise separation method we do not have to deal with the issue of empty chunks (without a bird calling/singing) which means we can use each chunk as a unique sample for training/testing. Third, we can let the network make multiple predictions per sound file (one prediction per chunk) and average them to generate a final prediction. This makes our predictions more robust and reliable. As an extension, one could try to merge multiple predictions in a more sophisticated way but, so far, no extensive testing has been done.

3 Data Augmentation

Because the number of sound files is quite small, compared to the number of classes (the training set (of 24'607 files) contains an average of only 25 sound files per class), we need additional methods to avoid over fitting. Apart from drop-out, data augmentation was one of the most important ingredients to improve the generalization performance of the system. We apply four different data augmentation methods. For an overview of the the impact each data augmentation method has, consult Table 1.

Table 1: Mean Average Precision for different runs on a dataset with 50 random bird species. The baseline run uses all data augmentation methods (Background Noise, Same Class Combining, Time Shifts and Pitch Shifts), while all the other runs are missing one or two of the data augmentation methods. We use "w/o" as an abbreviation for "without". The first column corresponds to the mean average precision when only the foreground (FG) species are considered. The second column also considers the species in the background (BG) as prediction targets. Underlined are the best results in each category. We stopped all runs after 12 hours of training time.

	MAP (FG only)	**MAP (FG & BG)**
Baseline	<u>0.842</u>	0.728
w/o Noise	0.831	<u>0.731</u>
w/o Same Class	0.839	0.730
w/o Time Shift	0.801	0.701
w/o Pitch Shift	0.828	0.725
w/o Noise and Same Class	0.768	0.661

3.1 Time Shift

Every time we present the neural network with a training example, we shift it in time by a random amount. In terms of the spectrogram this means that we cut it into two parts and place the second part in front of the first (wrap around shifts). This creates a sharp corner where the end of the second part meets the beginning of the first part but all the information is preserved. With this augmentation we force the network to deal with irregularities in the spectrogram and also, more importantly, teach the network that bird songs/calls appear at any time, independent of the bird species.

3.2 Pitch Shift

In a review of different augmentation methods [12] showed that pitch shifts (vertical shifts) also helped reducing the classification error. We found that, while a small shift (about 5%) seemed to help, a larger shift was not beneficial. Again we used a wrap-around method to preserve the complete information.

3.3 Combining Same Class Audio Files

We follow [14] and add sound files that correspond to the same class. Adding is a simple process because each sound file can be represented by a single vector. If one of the sound files is shorter than the other we repeat the shorter one as many times as it is necessary. After adding two sound files, we re-normalize the result to preserve the original maximum amplitude of the sound files. The operation describes the effect of multiple birds (of the same species) singing at the same time. Adding files improves convergence because the neural network sees more important patterns at once, we also found a slight increase in the accuracy of the system (see Table 1).

3.4 Adding Noise

One of the most important augmentation steps is to add background noise. In Section 2.1 we described how we split each file into a signal and noise part. For every signal sample we can choose an arbitrary noise sample (since the background noise should be independent of the class label) and add it on top of the original training sample at hand. As for combining same class audio files, this operation should be done in the time domain by adding both sound files and repeating the smaller one as often as necessary. We can even add multiple noise samples. In our test we found that three noise samples added on top of the signal, each with a dampening factor of 0.4 produces the best results. This means that, given enough training time, for a single training sample we eventually add every possible background noise which decreases the generalization error.

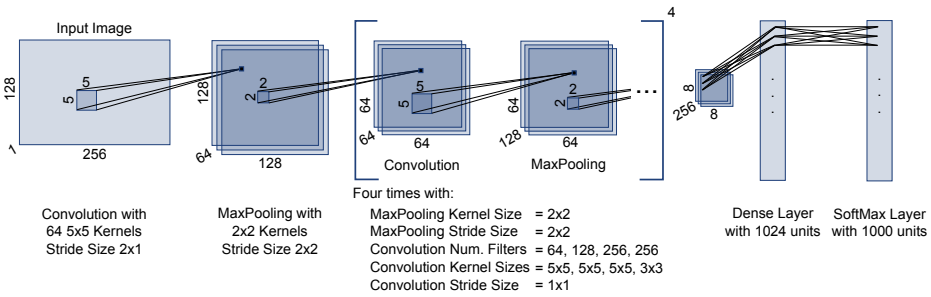

Fig. 5: Architecture used for the run "Cube Run 2" in the BirdCLEF 2016 Recognition Challenge. For "Cube Run 3" the same architecture was used but the input image had dimensions 256 by 512.

4 Network architecture

Figure 5 shows a visual representation of our neural network architecture. The network contains 5 convolutional layer, each followed by a max-pooling layer. We insert one dense layer before the final soft-max layer. The dense layer contains 1024 and the soft-max layer 1000 units, generating a probability for each class. We use batch normalization before every convolutional and before the dense layer. The convolutional layers use a rectify activation function. Drop-out is used on the input layer (probability 0.2), on the dense layer (probability 0.4) and on the soft-max layer (probability 0.4). As a cost function we use the single label categorical cross entropy function (in the log domain).

4.1 Batch Size

We use batches of 8 or 16 training examples. We found that using 16 training samples per batch produced slightly better results but, due to memory limitations of the GPU, some models were trained with only 8 samples per batch. If many samples, from the same sound file, are present in a single batch, the performance of the batch normalization function drops considerably. We, therefore, select the samples for each batch uniform at random without replacement. Normalizing the sound files beforehand might be an alternative solution.

4.2 Learning method

We use the Nesterov momentum method to compute the updates for our weights. The momentum is set to 0.9 and the initial learning rate is equal to 0.1. After 4 days of training (around 100 epochs) we reduce the learning rate to 0.01.

5 Results

We evaluate our results locally by splitting the original training set into a training and validation set. To preserve the original label distribution we group files

by their class id (species) and used 10% of each group for validation and the remaining 90% for training. Note that, even for our contest submissions, we never trained on the validation set. Our contest results would probably improve, if training would be performed on both training and validation set.

Training the neural network takes a lot of time. We, therefore, choose a subset of the training set, containing 50 different species, to fine tune parameters. This (20 times smaller) dataset enabled us to test over 500 different network configurations. Our final configuration was then trained on the complete training set (considering all 999 species) and reached an accuracy score of 0.59 and a mean average precision (MAP) score of 0.67 on the local validation set (999 species). On the remote test set our best run reached a MAP score of 0.69 when considering only the main (foreground) species, 0.55 when considering the background species as well and 0.08 when only background species were considered. This means our approach outperformed the next best contestant by 17% in the category where background species were ignored. Figure 6 shows a visual comparison of the scores for all participants. As seen in Figure 6 we submitted a total

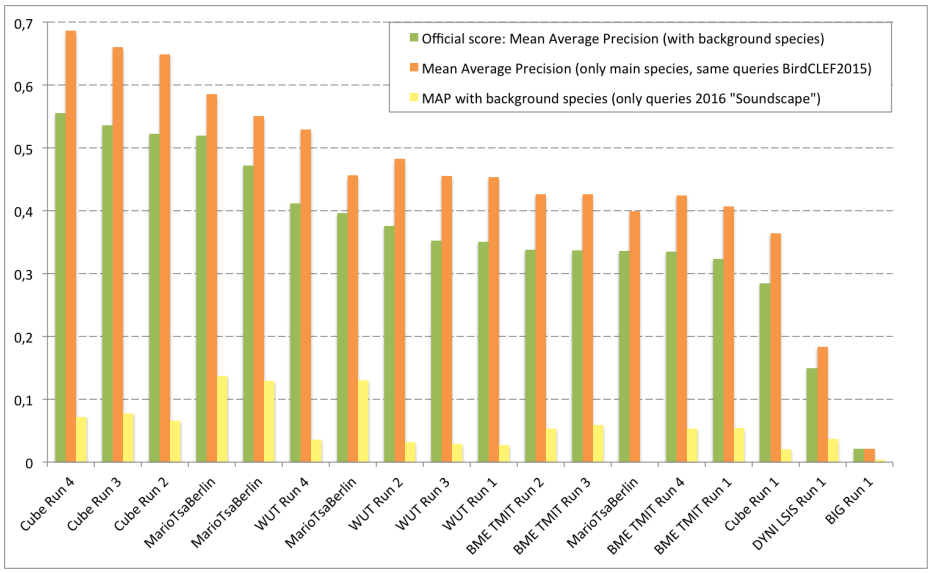

Fig. 6: Official scores from the BirdCLEF 2016 Recognition Challenge. Our team name was "Cube" and we submitted four runs.

of four runs. The first run "Cube Run 1" was an early submission where parameters had not yet been tuned and the model was only trained for a single day. The second and third run were almost identical but "Cube Run 2" was trained on spectrograms that were resized by 50% while "Cube Run 3" was trained on the original sized spectrograms. Both times the model was first trained for 4

days, using the Nesterov momentum method (momentum = 0.9, learning rate = 0.1) and then trained for one more day with a decreased learning rate of 0.01. Furthermore, "Cube Run 3" was trained with a batch size of 8 because of the limited GPU memory, while "Cube Run 2" was able to use batches of size 16 (scaled spectrograms). Finally, "Cube Run 4" was created by simply averaging the predictions from "Cube Run 2" and "Cube Run 3". We can see that "Cube Run 4" outperformed all other submission which means that an ensemble of neural networks could increase our score even further.

5.1 Discussion

Our approach surpassed state of the art performance when targeting the dominant foreground species. When background species were taken into account, other approaches performed almost as well as ours. When no foreground species was present one other approach was able to outperform us. This should not surprise us, considering our data augmentation and preprocessing method. First of all, we were cutting out the noise part, focusing only on the signal part. In theory this should help our network to focus on the important parts but in practice we might disregard less audible background species. Second, we are augmenting our data by adding background noise from other files on top of the signal part. As shown in Table 1, the score for identifying background species increases if we train without this data augmentation techniques. That means, even though, we do not use any data augmentation method when dealing with the test set, the network is still trained to ignore everything that happens in the background. One possible solution would be to alter the cost function and target background species as well. An other solution could be to employ a preprocessing step that tries to split the original files into differently sized parts, each part containing only one bird call/song. This is similar to [7] who compares single bird calls/songs instead of complete sound files.

5.2 Unsuccessful approaches we tested

We tested a lot of different ideas and not all of them worked, we will briefly list them in this chapter to give a complete picture.

Bi-directional LSTM Recurrent Neural Networks: We tried different cost functions and parameters but were not able to match the performance of our convolutional neural network.

Regularization: We tested L1 and L2 regularization of all weights but found that our generalization error did not decrease. Furthermore, adding these extra terms made training considerably slower.

Non-Square-Filters: For the convolutional layers we tried to use non square filters because we wanted to treat the time dimension differently than the frequency dimension. We found, however, that small variations did not change the performance while an attempt with a 1D (height of filter equals height of spectrogram) convolution produced worse results.

Deeper-Networks: We tried to add more layers to our neural network but the performance dropped after adding the 5th layer. This seems to be a common problem and many solutions have been proposed, for example, using highway networks [13]. We have not tested any of these proposed solutions but they might be an important ingredient in an attempt to increase the accuracy of the system even further.

6 Outlook

We already mentioned a few improvements that could be made. One idea is to use an ensemble of neural networks. An other idea is to modify our cost function to consider the background species or present single bird calls/songs instead of the currently fixed sized samples. One problem with the current approach is that longer files, as they generate more chunks, seem more important to the network. To combat this, we could show the same number of chunks for each class by repeating chunks from classes with a lower number of chunks / shorter files. Finally, the dataset provides us with a lot of meta-data: Date, time and location to name a few. We are currently only relying on the sound files but incorporating these values could greatly increase our score because we could narrow down the total number of species which we need to consider. While testing parameters, we found, for example, that with only 50 different species, we were able to reach a MAP score around 0.84 (compared to 0.67, our best score on the validation dataset). Training models for different regions / species and combining them using the meta-data, therefore, seems like a natural extension to the current approach.

Acknowledgements. We would like to thank Amazon AWS for providing us with the computational resources, Ivan Eggel, Hervé Glotin, Hervé Goëau, Alexis Joly, Henning Müller and Willem-Pier Vellinga for organizing this task, the Xeno-Canto foundation for nature sounds as well as the French projects Pl@ntNet (INRIA, CIRAD, Tela Botanica) and SABIOD Mastodons for their support. Last but not least E.S. would like to thank Samuel Bryner, Judith Dittmer, Nico Neureiter and Boris Schröder-Esselbach for their helpful remarks and guidance throughout this project.

References

1. Imageclef / lifeclef - multimedia retrieval in clef. `http://www.imageclef.org/node/199`, accessed: 2016-05-22

2. De Schauensee, R.M., Eisenmann, E.: The species of birds of South America and their distribution. Academy of Natural Sciences; dist. by Livingston, Narberth, Pa. (1966)
3. Goëau, H., Glotin, H., Planqué, R., Vellinga, W.P., Joly, A.: Lifeclef bird identification task 2016. In: CLEF working notes 2016 (2016)
4. Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W.P., Champ, J., Planqué, R., Palazzo, S., Müller, H.: Lifeclef 2016: multimedia life species identification challenges. In: Proceedings of CLEF 2016 (2016)
5. Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W.P., Planqué, R., Rauber, A., Palazzo, S., Fisher, B., et al.: Lifeclef 2015: multimedia life species identification challenges. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction, pp. 462–483. Springer (2015)
6. Joly, A., Leveau, V., Champ, J., Buisson, O.: Shared nearest neighbors match kernel for bird songs identification-lifeclef 2015 challenge. In: CLEF 2015. vol. 1391 (2015)
7. Lasseck, M.: Bird song classification in field recordings: winning solution for nips4b 2013 competition. In: Proc. of int. symp. Neural Information Scaled for Bioacoustics, sabiod. org/nips4b, joint to NIPS, Nevada. pp. 176–181 (2013)
8. Lasseck, M.: Improved automatic bird identification through decision tree based feature selection and bagging. In: Working notes of CLEF 2015 conference (2015)
9. Marler, P., Tamura, M.: Song "dialects" in three populations of white-crowned sparrows. The Condor 64(5), 368–377 (1962)
10. Restall, R.L., Rodner, C., Lentino, R., et al.: Birds of northern South America. Christopher Helm (2006)
11. Ridgely, R.S., Tudor, G.: Field guide to the songbirds of South America: the passerines. University of Texas Press (2009)
12. Schlüter, J., Grill, T.: Exploring data augmentation for improved singing voice detection with neural networks
13. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: Advances in Neural Information Processing Systems. pp. 2368–2376 (2015)
14. Takahashi, N., Gygli, M., Pfister, B., Van Gool, L.: Deep convolutional neural networks and data augmentation for acoustic event detection. arXiv preprint arXiv:1604.07160 (2016)
15. Xeno Canto Foundation: Sharing bird sounds from around the world (2012)