Bird Audio Detection Challenge Working Notes

Jens Johannsmeier
University of Potsdam
Potsdam, Germany
Email: johannsmeier@uni-potsdam.de

Sebastian Stober
University of Potsdam
Potsdam, Germany

***Abstract*—This is a description of our algorithm entry for the Bird Audio Detection Challenge. I will first give a brief summary followed by a more detailed description of the different parts.**

I. Overview

At first, we simply used a deep convolutional neural network with mel spectrograms as input. After the convolutional layer, there is a single fully-connected layer with a sigmoid activation function producing the output. As the sequences were variable in length (meaning that the spectrograms couldn't just be flattened before putting them through the fully-connected layer) we opted to take the maximum over the time axis after the convolutions, which removed any variability in terms of dimensionality between inputs.

In an attempt to improve performance, we used a scheme inspired by [1]. We split each sequence into snippets of equal length (zero-padding if necessary). Each snippet is assigned the same label as the corresponding sequence. This gives a correct labeling for sequences that do not contain a bird (since no snippet will contain one), but will likely mislabel many snippets from sequences containing birds (all snippets are assumed to contain a bird even though many will not). A model is trained on these labels until convergence on a development set. Then, all non-zero labels in the data are replaced by the model's output on the corresponding snippets. The idea is that the model will learn (to a degree) what a "non-bird" snippet will look like and "refuse" to assign the label 1 (or any value close to that) to some of the wrongly-labeled snippets from sequences containing birds. This process can be iterated (training on the new dataset containing "soft" labels) any number of times, e.g. until convergence. The output for a sequence can then be obtained by aggregating over the outputs for all corresponding snippets in some way (e.g. we found taking the maximum works well as a simple method). In the rest of this report, we will call this the "soft label" approach.

II. Details

Everything was implemented in Python, using the Blocks and Fuel libraries [2]. The code repository can be found here: https://bitbucket.org/xdurch0/bird_detection/

A. Dataset and Processing

We used the entirety of the Freefield and Warblr sets. 15% of the sequences were chosen at random and used as a development set. The audio processing was done in Python with the help of librosa [3]. Each sequence was transformed to a mel spectrogram using 128 frequency bins, a window size of 5000[1] and a window overlap of 75% (hop size of 1250). Sequences longer than 600,000 frames were skipped, but this only amounted to a handful in the Warblr set.

For the soft label approach, we divide the data based on the mel spectrograms. As a reference, a clip of exactly 10 second length (441,000 frames) would result in a mel spectogram with 353 time frames, given our window size and overlap. We divided these into snippets of 35 frames (so covering roughly one second of audio) with a hop length of 17 (meaning slightly above 50% overlap).

B. Model

Our model is a deep convolutional neural network. The data is first put through four convolutional layers, the parameters of which are given in Table I. Every layer uses the *PReLU* nonlinearity [4] as well as batch normalization [5] (before the nonlinearity). Pooling is done every second layer, *after* applying the nonlinearity. The order is relevant as *PReLU* can be non-monotonic. After the last convolutional layer, the maximum is taken over the time dimension and the remaining dimensions (frequencies and channels) flattened (resulting in a 675-dimensional vector) and put through a fully-connected layer with a sigmoid nonlinearity mapping to a single number, which is the output of the model.

Overall, the model has 38,077 parameters, most of which are in the convolutional layers (the last two layers alone have over 30,000 parameters).

C. Training

Weights were initialized using the scheme described in [4].

Batch size was set to 2048 for the soft label approach (64 when we did the "regular" end-to-end setting), with training examples being shuffled before every epoch. Note that for the soft label approach, the snippets coming from one sequence

[1]This probably unusually large number came from a miscalculation to be honest, but smaller, more fine-grained windows did not lead to improvements.

TABLE I
CONVOLUTIONAL LAYER PARAMETERS. ALL CONVOLUTIONS USED STRIDE 1 IN BOTH DIMENSIONS. MAX POOLING WAS DONE OVER AREAS OF SIZE 2×2 WITH IDENTICAL STRIDES.

Layer#	#Filters	Filter Size	Max Pooling
1	25	3×3	No
2	25	3×3	Yes
3	25	5×5	No
4	25	5×5	Yes

were treated as being completely independent, i.e. may or may not have appeared together in a batch. If needed, data within each batch was zero-padded to achieve equal length in the time axis.

Cross-entropy was used as the cost function. We trained using gradient descent and Adam [6] with the default parameters used in Blocks, which are the parameters proposed in the original paper. We clipped gradients to a norm of 1, but generally observed gradients much smaller than this.

We used an early stopping approach for terminating the training; cross-entropy was measured on the development set after every epoch and training was stopped if no improvement was seen over the course of three epochs.

For our final submissions, the soft label approach was iterated six times. We did not test whether the soft labels had converged at this point, but found very little performance difference between the later iterations.

D. Submission

For our last two submissions (which are our best ones), the test set was processed as described above and all of the snippets run through the model. The "score" (probability) for each sequence was then the maximum over the probabilities for all snippets (second to last submission) or the mean of the maximum three probabilities (last submission).

III. DISCUSSION

A. Things we tried but ended up not using

There are several techniques that certainly have potential to improve our model, but we were not able to use to any effect. Among these are tweaks such as different activation functions, batch sizes, gradient descent step rules or convolutional parameters, but here we would like to briefly discuss some that we believe are worth mentioning:

Dropout [7] is seen as a way to improve pretty much any deep model with regards to its generalization ability. However, we did not see any benefit from using it, likely due to lack of tuning.

The decision to take the maximum over the time axis after the last convolutional layer may seem a bit drastic. As an alternative, we tried randomly cropping or padding all sequences to the same length (exactly 441,000 frames). This allowed us to flatten the time axis as well instead of summarizing over it. While this significantly improved performance on the training set, it also negatively impacted generalization performance, and we could not get it to the same levels as with using the maximum.

Since the dataset is fairly small by deep learning standards, we investigated augmenting it by randomly picking two sequences, and adding them together (and then converting to a mel spectogram) to form a "new" sequence. This new sequence would count as containing a bird if either or both of the added sequences contain one. We sampled sequences such that the proportion of birds in the augmented data stayed the same. Since the Freefield and Warblr sets contain about 50% birds taken together, this meant sampling sequences containing birds with about 29% probability for the purpose of creating new sequences. While this augmentation procedure resulted in some gains for the original end-to-end approach, we did not see any improvements for the soft label approach – possibly because the dataset is much larger as every snippet is counted as one data point, resulting in around a twentyfold increase. We also thought about trying a "smart" augmentation approach where new examples are generated on the fly to replace those that the model already classifies correctly, but did not get around to implementing this.

Finally, we investigated chirplets [8] as an alternative to mel spectrograms. Here, we ran into some technical difficulties since the dataset would have taken up too much space to store on our server using the default parameters, and transforming sequences on-the-fly took prohibitively long. We investigated using chirplets where we maximized over either the filter or frequency axes, but got worse results than with mel spectrograms.

B. Avenues for improvement

One big problem for us was a lack of time and computational resources. Given more of these factors, we could have conducted a proper search for suitable hyperparameters. As is, the model is very much unoptimized and many of the parameter choices (also regarding data preprocessing) are arbitrary. In particular, our model only has around 38,000 parameters as stated above – to our knowledge, this is very small for a convolutional network by today's standards. It may well be that a much larger (i.e. more filters and/or more layers) model that has been properly regularized (e.g. by a well-tuned dropout) would have performed significantly better.

Regarding our soft label approach, the method used for aggregating the different snippets could have been different. We used the maximum because it was simple and worked reasonably well (slightly better than the original end-to-end approach). However, another method might be more suitable. We also investigated using the average of the three maximum snippets, hypothesizing that this would give more robust results. While this improved performance on the development set slightly, test set performance was slightly reduced.

Our motivation for the soft label approach was that it might allow the model to learn on a more fine-grained basis rather than on long sequences spanning several seconds. Many of our decisions here have been arbitrary, so further investigation could pay off. What is nice is that this method is orthogonal to

most considerations regarding the actual model used. In fact, we used exactly the same model for the soft labels as for the original end-to-end approach, simply swapping out the dataset.

IV. Conclusion

While we are likely far from taking a top spot in the challenge, we hope that some of the things we did (or did not do) can shed light on what does or does not work when trying to detect birds in audio recordings. We might try to gain further insights into where our model runs into trouble in the coming weeks and hope to improve it further.

References

[1] J. Schlüter, *Learning to Pinpoint Singing Voice from Weakly Labeled Examples*, Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) (pp. 44-50), 2016.

[2] B. Van Merrinboer et al., *Blocks and fuel: Frameworks for deep learning*, arXiv preprint arXiv:1506.00619, 2015.

[3] B. McFee et al., *librosa: Audio and music signal analysis in python*, Proceedings of the 14th Python in Science Conference, 2015.

[4] K. He et al., *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, Proceedings of the IEEE International Conference on Computer Vision (pp. 1026-1034), 2015.

[5] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, arXiv preprint arXiv:1502.03167, 2015.

[6] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, 2014.

[7] G.E. Hinton et al., *Improving neural networks by preventing co-adaptation of feature detectors*, arXiv preprint arXiv:1207.0580, 2012.

[8] D. Stowell and M.D. Plumbley, *Framewise heterodyne chirp analysis of birdsong*, Proceedings of the 20th European Signal Processing Conference (EUSIPCO) (pp. 2694-2698), IEEE, 2012.