Bird sound classification using a bidirectional LSTM

Lukas Müller and Mario Marti

ZHAW Zurich University of Applied Sciences, School of Engineering, Winterthur, Switzerland
lukas@roraro.ch, mario.marti@outlook.com

Abstract. While RNN's are widely applied to many similar tasks, such as automatic speech recognition and transcription, nobody succeeded yet using them for classification in the BirdCLEF challenge. Recent work at ZHAW on the topic of speaker clustering has yielded very promising results using a Bidirectional LSTM.
We have set out to apply the LSTM-Network mentioned above to the BirdCLEF challenge. By doing so, we hope to make a valuable contribution to the ongoing research in this field.
Unfortunately, our approach did not perform as well as we hoped for. In the following, we detail the approach taken and some issues a bachelor student with little experience in the field of machine learning will encounter when participating in such a challenging competition.

Keywords: LSTM · Bi-directional · RNN

1 Introduction

Recognising birds by their song in the wild is a challenging task. One reason for this is that according to recent research by [2] there are approximately 18'043 different bird species. Since birds do not just vocalise in bird songs but also various types of calls such as alarm calls or contact calls [1], this leads to a huge classification problem.

The Cross-Language Evaluation Forum (CLEF) hosts an annual challenge for bird classification. This BirdCLEF challenge [5] allows researchers from all over the world to test their approaches against state of the art for bird voice recognition. The organisers have built a large dataset based on contributions of Xeno Canto[1], a network for sharing bird songs from all over the world.

The Datalab at Zurich University of Applied Sciences had great success over the last years in applying recurrent neural networks, specifically LSTMs [4], to speaker recognition tasks, such as speaker clustering.

Since the BirdCLEF challenge provides an exciting opportunity to test those approaches on a big dataset, we have set out to show that an LSTM can perform as well on the task as a CNN based approach.

[1] https://www.xeno-canto.org

2 Methodology

We participated in the challenge in the context of our bachelor's thesis. As such we did not have much experience in machine learning, let alone signal processing. For that reason, we decided to try and bring established preprocessing methods, and the LSTM mentioned earlier together. This way we would not need to build up the full knowledge required when building a different approach from scratch.

2.1 Preprocessing

When the baseline [7] [6] was released, we had already started to implement a pipeline in Python 3.5, matching the software versions used in [10]. We integrated and subsequently used the preprocessing and augmentation steps found in the baseline [7] [6].

Differences to the baseline As a signal to noise threshold value, we used $1e-3$, as that value was mentioned in the readme in [6]. We did not experimentally validate that threshold.

The LSTM Model was originally designed to work on spectrograms of 0.5 second long segments and a resolution of 50 pixels wide by 128 pixels wide. With the intentions to keep the input similar, we changed the corresponding settings compared to the baseline.

This results in an identical nfft-window length of 1024 but increases the hop-length from 173 to 450. Effectively the baseline system used windows overlapped by 83.2%, while our variant shrinks this overlap to 56.0%.

2.2 Model

The model we have used for our participation in BirdCLEF was based on [10]. As mentioned above, we use inputs of 50 pixels width for segments that are 5 seconds long. Other than in the paper above, we use more hidden units in each bidirectional LSTM layer. This change was derived during one of our experiments.

2.3 Training

Datasets Before we started training, 5% of the provided training set were put aside in a validation set. We used stratified sampling to ensure that at least one recording for each class was present in the validation split.

We then went on to generate spectrograms out of these recordings, as described in 2.1.

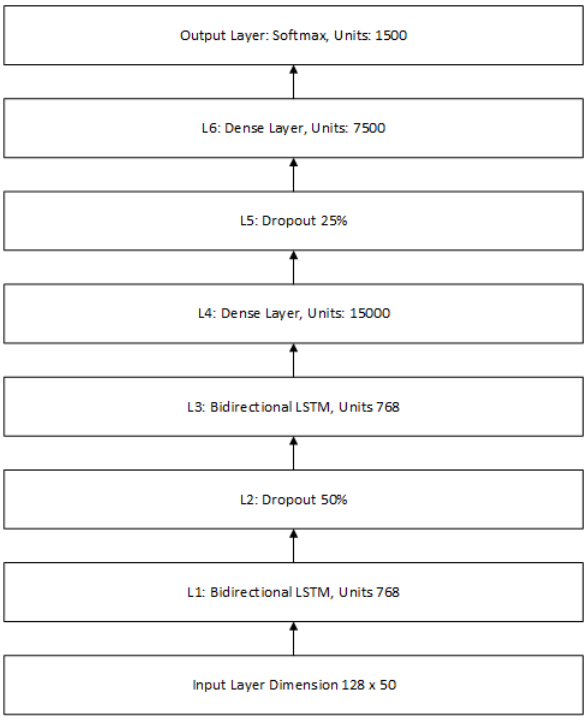


Fig. 1. Architecture used based on [10]

Caches Since the dataset was stored on network storage, we have encountered a bottleneck when loading the samples. To enable for more sequential reads, we first shuffled the generated dataset and then created caches containing 12800 samples and labels each. For this we have made use of pythons `pickle` module.

During training, a producer-consumer pattern was applied, with multiple threads loading samples from the caches and augmenting them for training. Every time a cache was loaded the contained sample, and label pairs were shuffled to safeguard against the constellation of samples in a batch repeating each epoch.

Augmentation For augmentation, we have settled on adding noise samples to the training samples. As in [7] those are byproducts of the preprocessing steps. We have also tested adding some gaussian noise and applying vertical roll. Those augmentation steps did not improve the generalisation of our model, so we have dismissed them. Based on the baseline we did augmentation on the fly with a probability of 0.5.

Training Duration and Batchsize While experimenting, we could not make out a difference in the performance of our model between training with batch sizes of 64, 128 or 256 samples per batch. The time it took to train however was

significantly lower using a higher batch size. For this reason, we have usually used a batch size of 256 samples.

Time permitting, we trained for at most 100 epochs without applying early stopping. Throughout this paper, we will refer to the time or number of batches it takes until the model was trained on each sample once as one epoch. After each epoch, we saved a snapshot of the trained model. Training one epoch took anywhere from 30 minutes to a couple of hours.

During training Adam [8] was used as optimiser with a learning rate of $1e-4$.

We did evaluate the snapshots for each epoch on the provided soundscape validation set. Time permitting, we also calculated an MRR score on our 5% validation split.

3 Experiments

To find the settings used for our runs, we have conducted multiple experiments. Out of these, we describe the two most relevant for our submission.

Hidden Units Since the model was originally used on TIMIT [3], a relatively small dataset of studio recordings, its capacity was lacking for the task at hand.

Our experiments showed that increasing the number of hidden units from 256 to 512 and 768 for each bidirectional LSTM layer lead to a significantly better MRR score. On the soundscape validation set, only a minor increase in performance was seen. While more hidden units lead to a better score, the performance gains showed a diminishing trend between higher numbers. At the same time, the training time increased. For this reason, we did not push past 768 hidden units.

Table 1. MRR scores for different numbers of hidden units on the local 5% validation split.

Nr. of hidden units	max MRR
256	0.384057
512	0.437551
768	**0.488797**

Augmentation As [7] shows, not every form of augmentation is necessarily good for a model. For this reason, we have conducted an experiment to show how augmentation through noise samples, adding of Gaussian noise and applying vertical roll influences the models performance. To do so, we have trained a model

using all of these methods and one using none of them as baselines. For each of the augmentation methods, we trained one model where that method was not used, while the other two were. Due to time limitations, we were not able to test all permutations of these augmentation methods.

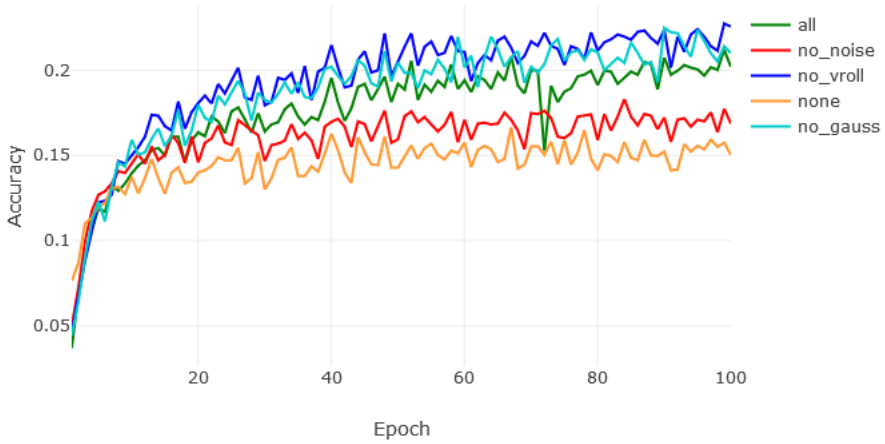


Fig. 2. Performance with different augmentation methods. The validation score is reporting lower than we would expect MRR performance to be, because we used top 1 accuracy on all 0.5-second segments for any recording. Including segments that only contain noise.

Since the models that were trained either without adding Gaussian noise or without applying vertical roll performed better than the baseline using all augmentation methods, we decided on not using those for our submission.

4 Submitted Runs

We have submitted four runs, one of which was an ensemble of two other runs. For both the soundscape task and the monophone task, we reported the 100 classes that were predicted with the highest confidence. For MRR the wrong predictions after the correct class do not matter for the score. For the soundscape task, we expected that there would be a sweet spot reporting fewer classes for each segment. However, experiments on the soundscape validation set showed that the score increased with the number of classes reported.

In the following, we present the settings used to train the models for our submissions. The number of hidden units refers to the hidden units in both

bidirectional LSTM layers, while dropout is the setting we used for the first dropout layer. The second dropout layer was set to use half as much dropout as the first. The augmentation probability is the probability determining if any augmentation would be done. The Gaussian noise parameter gives the maximum intensity of Gaussian noise that will be added to the sample, and lastly, we controlled how much downsampling was done, by setting an upper limit on the number of samples per class.

4.1 Run 1

The model used for Run 1 was trained for 94 epochs on an old dataset which initially had a faulty separation of the training and validation splits, letting it train on audio files that were also used in validation. For the submission, the model was trained for an additional 34 epochs on a new and corrected dataset which did not have that issue.

Table 2. Configuration for run 1

Hidden units	256
Dropout	0.5
Augmentation probability	0.5
Gaussian noise	0
Samples per class	1500

4.2 Run 2

Run 2 was trained for 34 epochs on a dataset, which was downsampled to 1000 samples per class instead of the 1500 used in the experiments and other runs. This run was primarily intended as a test run to evaluate if a new update to our pipeline would work. Due to its performance and the lack of better models, we submitted this run as our second run. Some settings for this run could be proven in our experiments to be beneficial (hidden units, learning rate), the influence of others was not experimentally determined (increased augmentation probability).

Table 3. Configuration for run 2

Hidden units	512
Dropout	0
Augmentation probability	0.6
Gaussian noise	0.05
Samples per class	1000

4.3 Run 3

This run was an ensemble of both run 1 and run 2. The ensemble was generated by using a python script and averages the values reported in run 1 and 2. We did not use a weighted average.

4.4 Run 4

For run 4 the newly generated dataset was used. The selection of the parameters was based on the results of the experiments described earlier in this thesis.

Table 4. Configuration for run 4

Hidden units	768
Dropout	0.5
Augmentation probability	0.5
Gaussian noise	0
Samples per class	1500

Since the results of the first three submissions were lower than expected, we reworked the process used for making predictions. As we used all spectrograms for a given monophone sample or soundscape segment, it is likely that some of them did not contain a usable signal. For these spectrograms, the results would be close to random. Once the predictions for the segment or monophone sample are pooled, these random predictions would inevitably pull down correct predictions.

As we have used a signal to noise threshold of 0.001 to decide if we train on any given spectrogram, we tested using the same threshold to filter out unsuitable spectrograms during predictions.

Briefly validating this hypothesis on the soundscape validation set, confirmed that the scores could be improved this way. Unfortunately, we only gained an additional score of 0.02 on the validation set on average.

5 Results

The results we achieved on both subtasks were underwhelming. Compared to other submissions, our approach consistently performed the worst. For the monophone task, we can see, that the increased capacity of the network and the changes made for run 4 did have a positive effect on the results. On the soundscapes that does not seem to be the case.

6 Challenges we faced

As mentioned before, we participated in BirdCLEF in the context of our bachelors' thesis. It was the first scientific challenge we took part in. Being bachelor

Table 5. Results achieved on the monophone task (MRR) and the soundscape task (c-MAP)

Run	MRR only main species	MRR with backgound species	c-mAP	c-mAP Peru	c-mAP Colombia
1	0.2014	0.1909	0.0195	0.0079	0.0244
2	0.2439	0.2285	**0.0319**	0.0123	**0.037**
3	0.2441	0.2295	0.0281	**0.0126**	0.0339
4	**0.2642**	**0.2466**	0.0291	0.009	**0.037**

students, we were fairly inexperienced in the field of artificial intelligence. In the previous semester, we have attended lectures on the subject, but we had little practical experience but were interested in gathering some experience by doing a bigger project. The BirdCLEF challenge was an ideal candidate for this, as recognising bird calls is an interesting topic with room for improvement.

While we were aware at the beginning that it is a challenging task, we overestimated our abilities and ultimately did not achieve the results we strived for. Handling such a large data set brought additional unexpected difficulties. In the following section, we would like to briefly describe the experiences of participating in the challenge and pick out some lessons we have learned.

6.1 Deadline, pressure and code quality

While we had experienced stressful times and worked towards deadlines both during our studies as well as our careers, we were surprised by the amount of pressure we felt close to the deadline of this challenge. This pressure was partly because we had high expectations for ourselves and partly because at times we hit so many problems that it looked like we would not even be able to submit any runs to the challenge.

We realised that the closer the deadline came, the more mistakes we began to make. Since the turnaround time for preprocessing a dataset was long, even small mistakes turned out to be costly. To get back on track and get our code quality back to a barely acceptable level, we started to either do code reviews or make use of pair programming. Even though it sometimes seemed to take a lot of time to do so, it was ultimately worth it, since even small mistakes did cost more time as mentioned above.

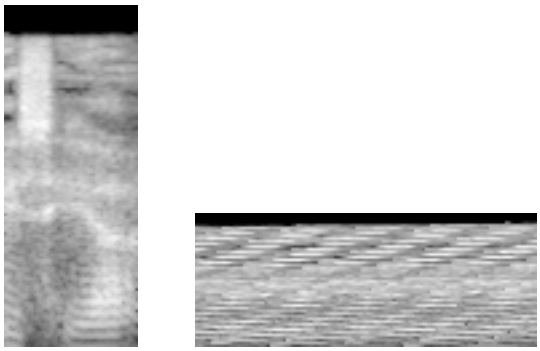
6.2 Time management

Initially, we planned to use most of our time to experiment with different approaches. In reality, an unexpectedly large part of our time went into the debugging of numerous steps from the generation of spectrograms to the output of training and validation values through the network.

We were a bit naive and assumed that we could debug the pipeline by training a model and analysing what did not work as expected. Unfortunately, we had

made some mistakes that led to a bad performance of the model, which we could not debug this way.

An example of such an error was what we dubbed the "reshape bug". Initially, we used a tried and true CNN to verify the pipeline was working. Once satisfied it worked correctly, we replaced the CNN with the LSTM mentioned before. Compared to the CNN, the LSTM expected the input axes to be swapped.



(a) The original input

(b) The input after the reshape operation

Fig. 3. Figure showing the "reshape bug", where mistakenly a wrong transformation was done, leading to garbeled inputs for the network.

Instead of the axes, we mistakenly used `numpy.reshape`, an operation that rearranges the input to fit the desired shape. The resulting spectrograms did not resemble the initial data at all. With the new input, the network still learned some features but was useless overall. In 3 you can see the original image and the output after the reshape operation.

Ultimately we used a small set to debug our whole pipeline to find this error. In retrospect, it would have been beneficial to carry out such a test immediately after the pipeline's completion and after introducing the input transformation for the LSTM. Because the training a model and preprocessing the dataset takes a very long time, we lost much time by trying to find the error by training the model on a complete dataset. Even worse, we conducted some experiments while the pipeline was still broken. These experiments were rendered completely unusable, which we only realised after they have been carried out. Repeating the experiments cost time again. Doing a thorough test of the pipeline is particularly important with such large data sets and turnaround times of several days for a complete training run.

6.3 Machine learning basics

Validation This is a rather basic lesson that we have learnt: it is important to have meaningful measurements for validation. Since a validation set for the

soundscape task was provided, we were a bit lazy in fixing the issues outlined below. In hindsight, we should have prioritised fixing those higher.

At first, our split between validation and test samples did divide all generated spectrograms without taking into account that spectrograms from the same recording would end up in both splits. Needless to say that our models performed very well on the validation samples.

We have later fixed that by first splitting the recordings and then generating the spectrograms. However, for the validation split, we did keep all spectrograms, regardless if they contained only noise or indeed a bird. Since no model can predict a bird only from background noise, unless there was a lot of overfitting on a very similar training sample, this resulted in a situation where we would get quite a low score (approx. 22% top 1 accuracy) on the validation samples. As we did not know how many of the generated spectrograms even contained a usable signal, this was not very meaningful.

The best way to avoid these problems is to use validation methods that closely mirror the measurements used for the submissions. Unfortunately, we did not have enough time to implement such a validation function in keras for use during training. Instead, we have loaded the snapshots of the model and used a script to make predictions on the validation split and subsequently calculate an MRR score.

To validate the models performance for the soundscape task, we have heavily relied on the provided soundscape validation set and the official script to calculate c-MAP scores. As it turned out, this validation set did was not representative of the soundscape test set. Which lead to a marked difference between the score on this set and the scores achieved in the competition.

All in all, we have learned to use measurements that are indicative of the performance in the real task and to ensure that the validation set is representative of the test set used in the challenge.

Experiment When conducting experiments, it is important that they are repeatable and well documented. Again this is nothing new, and we were told that this is important early on.

We have made sure that we can have a configuration or a script which sets up the experiment for each one. That ensures repeatability up to a point. Namely, we have realised over the course of this challenge, that we have made use of Pythons keyword arguments with default values in various places. This, in turn, led us to omit some *"default"* parameters in the scripts. Later on, we changed some default values in the method signatures. Hence, the previous configurations were rendered obsolete, and it became hard to be sure we have used the correct settings. For that reason, we had to repeat some experiments.

While using configuration files or scripts to get reproducible experiments is great when the above pitfall is avoided, we have learned that it is best also to document the reasoning behind the experiment somewhere. Additionally, a consistent naming scheme for artefacts such as snapshots of the trained model, tensorboards, graphs or other analysis on the data can help to keep some order in

the experiments. Effectively all time spent on documenting ongoing experiments will save time when writing up the results in a paper or thesis.

For long-running experiments, it is also advisable to form a robust hypothesis and do some research to validate it, before running an experiment. As such it will ensure, that the precious time it takes to run the experiments will be well spent.

6.4 Handling a big dataset

The training set for 2018 contained 113 gigabytes worth of audio recordings. The resulting training sets contained over 1200000 spectrograms, which were saved in the PNG format. Additionally, some spectrograms were saved to be used for augmentation. Converting all recordings into a ready-to-train dataset took upwards of 12 hours.

Since the turnaround time for generating a dataset is so big, it is well worth creating a smaller dataset to test the generation process. Since it is not necessary to do any training on that dataset, this can be done by merely choosing recordings from a subset of the 1500 species. Ideally, the quality of the recordings should range from noisy to very clean recordings.

Such a dataset can be used to quickly see the effect of different settings for the transformations, gain a better understanding of the influence of different signal detection methods or perform sanity checks as described below. Due care needs to be taken to ensure no hyper-parameters are tuned on such a set, as it is not representative of the whole dataset.

7 Future Work

7.1 Using longer segments for training:

One experiment showed that using spectrograms which represented only 0.2 seconds of audio performed significantly worse than using the 0.5 seconds used for our submissions. Additionally, participants in the past seem to be using longer segments (e.g. lengths of 3 seconds were used by [9]).

Based on this information, the hypothesis that longer samples improve the performance of the model can be formed and should be experimentally tested.

7.2 Preprocessing

For this thesis, the preprocessing provided by the baseline system [7] was used. These steps were originally all used in combination with a CNN and might not be suitable for the use with an RNN. It could be beneficial to evaluate if there are some RNN specific preprocessing steps to help the network better understand the data provided.

7.3 Use of variable input lengths for the BLSTM

At the moment, the model is limited to a fixed input length. In principle, an LSTM supports variable input lengths. Using variable input lengths could allow for more complex and entirely different preprocessing methods. On the other hand, the whole training sample could be fed to the network, hoping that it will learn to discern noise from signal by itself.

8 Conclusion

Due to multiple technical issues and the limited time frame, we were not able to show that an LSTM can perform as well as a state of the art CNN in the BirdCLEF challenge.

While the results show that the network did indeed learn something usable, there certainly is room for improvement. We assume that there is a significant issue either within preprocessing or the used hyper-parameters. Unfortunately, we were not able to determine what the issue is. This is partly due to not being able to run extensive and systematic enough experiments due to time lost to technical issues. For this reason, we cannot conclusively say whether an LSTM based approach is or is not unable to perform as well as a CNN on this task.

However, the application of an LSTM to the complex task available in the form of BirdCLEF warrants further research, as LSTMs are applied highly successful in similar domains.

While we did not achieve the results we had hoped for, participating in BirdCLEF was interesting and a tremendous opportunity to gain practical experience with artificial intelligence. We certainly learned a lot.

9 Acknowledgements

We would like to thank all the people that supported us through this journey. Namely, we want to thank Thilo Stadelmann, Martin Braschler and Mohammadreza Amirian from the ZHAW Datalab for their great support and supervision of our project. Additionally, we would like to thank Niclas Simmler and Amin Trabi for discussions and inputs. Also, we want to thank Stefan Kahl for answering questions regarding their 2017 submission. Finally, we want to thank the organisers for organising such a great challenge and providing us with all the necessary information for the task.

References

1. Types Of Bird Songs And Calls, https://www.britishbirdlovers.co.uk/identifying-birds/types-of-bird-songs-and-calls
2. Barrowclough, G.F., Cracraft, J., Klicka, J., Zink, R.M.: How Many Kinds of Birds Are There and Why Does It Matter? PLOS ONE **11**(11), e0166307 (Nov 2016). https://doi.org/10.1371/journal.pone.0166307, http://dx.plos.org/10.1371/journal.pone.0166307

3. Fisher, W., Doddington, G., Goudie-Marshall, K.: The darpa speech recognition research database: Specifications and status. In: in Proc. DARPA Workshop on Speech Recognition. pp. 93–99 (1986)
4. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, http://dx.doi.org/10.1162/neco.1997.9.8.1735
5. Joly, A., Goëau, H., Botella, C., Glotin, H., Bonnet, P., Planqué, R., Vellinga, W.P., Müller, H.: Overview of lifeclef 2018: a large-scale evaluation of species identification and recommendation algorithms in the era of ai. In: Proceedings of CLEF 2018 (2018)
6. Kahl, S., Wilhelm-Stein, T., Hussein, H., Klinck, H., Kowerko, D., Ritter, M., Eibl, M.: [GIT] Source code of the TUCMI submission to BirdCLEF2017 (Jan 2018), https://github.com/kahst/BirdCLEF2017
7. Kahl, S., Wilhelm-Stein, T., Klinck, H., Kowerko, D., Eibl, M.: Recognizing Birds from Sound - The 2018 BirdCLEF Baseline System. arXiv:1804.07177 [cs] (Apr 2018), http://arxiv.org/abs/1804.07177, arXiv: 1804.07177
8. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs] (Dec 2014), http://arxiv.org/abs/1412.6980, arXiv: 1412.6980
9. Sprengel, E., Jaggi, M., Kilcher, Y., Hofmann, T.: Audio Based Bird Species Identification using Deep Learning Techniques. In: CLEF (Working Notes). pp. 547–559 (2016)
10. Stadelmann, T., Glinski-Haefeli, S., Gerber, P., Drr, O.: Capturing Suprasegmental Features of a Voice with RNNs for Improved Speaker Clustering. In: Proceedings of the 8th IAPR TC 3 Workshop on Artificial Neural Networks for Pattern Recognition (ANNPR18)