Detection and Multi-Label Classification of Bats

Lucile Dierckx[1,2][0000−0003−2855−1042], Mélanie Beauvois, and Siegfried Nijssen[1,2][0000−0003−2678−1266]

[1] TRAIL Institute, Belgium
[2] ICTEAM/INGI, UCLouvain, Belgium
{lucile.dierckx, siegfried.nijssen}@uclouvain.be

Abstract. As bats are an important indicator for the health of their habitat, projects in multiple countries monitor bat populations by collecting audio recordings of bat calls. Analysing these recordings is however a tedious task and there is a need for systems that accurately and efficiently detect and classify bat calls. While earlier studies focused on detection and classification separately, in this paper we propose a first approach that combines these two tasks. Moreover, we aim to build a multi-label classifier that is able to detect if multiple bat species are present in the same audio recording. One of the challenges we face is that the available data focuses either on detection or single-label classification, but not on the combined task of detection and multi-label classification. We propose to address this by a data augmentation approach, and demonstrate that the resulting approach achieves the objectives of being accurate and efficient.

Keywords: Acoustic event detection and classification · Multi-label classification · Acoustic bat monitoring

1 Introduction

Bats are mammals that are very sensitive to their environmental changes and are, therefore, often used as a bioindicator of the health of their habitat [8]. Nowadays, one out of three species in Europe suffers from a sharp population decline, which is, among others, due to the destruction of the bats' roosting and hunting sites, the poisoning of insects with chemicals and the collisions of bats with built infrastructures such as wind farms [15]. That is why various organisations monitor the number of bats and the different species present on given sites. Bats can be monitored using portable recorders that start to record when hearing sounds above a fixed frequency threshold. The obtained recordings then have to be analysed to identify the calls and the species that emitted them. This tedious task is very time-consuming and requires a good knowledge of bats.

As the number of available recordings constantly increases and not all study groups have a chiropterologist available, there is a clear need for an automated analysis of these audio recordings. Some commercial products exist, such as

SonoChiro, SonoBat and Kaleidoscope, but these are expensive and, as shown in [11,14], not highly accurate.

The motivation for our work is that there is a need for open software that meets the following challenges:

C1 It should be able to *detect* at which moments in audio signals a bat is present;
C2 It should be able to *classify* which types of bats are present at those moments;
C3 It should be able to perform *multi-label* classification, i.e., it should identify multiple bats in the scenarios in which more than one bat species emit calls simultaneously, which happens regularly in nature;
C4 It should perform its task *accurately*;
C5 It should perform its task with *low computational resources,* as the final goal is to analyse the recordings on the devices that collect the data.

Some research projects have already studied the analysis of bat recordings. In particular, Bat detective [9] studied the detection of bats; BatNet [5] studied the classification of bat calls. However, none of these projects meets all challenges C1–C5. The goal of this paper is to propose a first approach to meet all these challenges.

Meeting these challenges is not straightforward. A major concern is the lack of available data. We only have access to two different types of data:

Detection data: these are audio recordings in which labels are present to indicate when a bat call is present, but the bat species are not known;
Classification data: these are audio recordings for calls of individual bat species; however, there is no label to indicate the timing of the bat calls.

To complicate the situation further, these data sets are from different locations, where the distribution of bat species may be different.

In this paper, we propose to address these challenges by using a data augmentation approach, in which the existing data sources are used to create new multi-label training data. This data is subsequently used to train a machine learning model. This machine learning model is a combination of a Convolution Neural Network (CNN) and an XGBoost, for which we will show that, on the combination of detection and multi-label classification tasks, the overall performance on the different challenges is better than that of architectures used in earlier work.

The rest of the paper is structured as follows: Section 2 gives an overview of the related work on this topic. Section 3 presents the input data available, the data augmentation and the architecture of our proposed model. The datasets used, the evaluation methods, the models used for comparison and the different results obtained are presented in Section 4. Finally, Section 5 concludes this paper.

2 Related Work

The first methods developed to automatically classify bat calls are based on the features that are used when the classification is performed manually, as the

peak frequency, the duration and frequency band of each call [1,2,10]. The Bat detective project [9] then showed that it was possible to apply computer vision models like CNNs to obtain good performance in the audio monitoring of bats. More importantly, it showed that it was possible to use raw audio as input instead of the precomputed features. Bat detective's program detects the positions of bat calls in audio files and achieves an average precision of approximately 0.88 and a recall at 0.95 precision of about 0.75. To do so, it uses a simple CNN with three convolutional and two dense layers, which receives spectrograms as input and outputs, for each window, the probability that it contains a call.

Some other projects then focused on the classification of bat calls to their corresponding species using CNNs. A first example is the work of Schwab et al. [12] that classifies the calls of bats present in Germany. They first implemented a small algorithm that identifies the position of calls in recordings, but they do not evaluate this detection in terms of performance. They compared four algorithms on the classification task and obtained the best mean accuracy (0.96) when using a modified version of ResNet50 [7] that takes spectrograms as input. Instead of using classical full-spectrum recordings, Tabak et al. [14] use zero-crossing acoustic data to perform the classification of bat calls. The calls were detected by finding sequences of consecutive decreasing frequency, and the images of the generated spectrograms were used as input features. The model used is ResNet18 and has an average accuracy of 0.92 and an average F1-score of 0.91. Some other works tried to avoid the use of too deep and heavy networks like ResNets. Zualkernan et al. [17], for instance, developed a CNN that has only 220K parameters and has an average accuracy of 0.9751 and average F1-score of 0.9578. The network receives Mel-scaled filter banks as input features. Finally, Chen et al. [5] implemented a network called BatNet to recognise 36 tropical bat species. Their particularity is that they have a "weak" label for all calls, no matter the species, that are not strong enough and could be misidentified. BatNet is composed of twenty-two convolutional layers and eight shortcut connections. The input of BatNet consists of the spectral image of the audio files. BatNet was compared to ResNet_v2, VggNet [13] as well as to a CNN inspired by Bat detective, and BatNet has the best AUC and overall accuracy among all.

All these papers mainly focus on the classification without performing or evaluating the detection step. Furthermore, they all have to discard from their training set the recordings where more than one species is present. Only Schwab et al. [12] provide circumstantial evidence that shows that for one of their recordings with two species present in it, their model gives a high probability for both species, but no specific evaluation is made on that topic. It is, however, important to be able to recognise calls in a multi-label context as it is common that, on the same observation site, more than one species is present and therefore, the different calls overlap in the recordings. That is why the model we propose in this paper focuses on making correct multi-label predictions in addition to the detection of bat calls.

To train models for detection and classification tasks, labelled training data is needed. Two types of labelled data can be distinguished in the aforementioned

studies. The first type is audio recordings in which the time intervals during which a bat call is present are indicated; the second type is audio recordings for which it is indicated which bat type is present in that recording, without indicating the time interval of the calls. Hence, a challenge is that there is no single labelled data set that can be used to train a model for both detection and multi-label classification at the same time.

3 Approach

This section introduces the details of our contribution.

3.1 Input Data

All datasets consist of audio recordings; to turn these data into training data, a window is slid over these recordings and, for each window, a spectrogram is calculated following the approach of Bat detective [9].

The available data differs in the label information that is available for each audio recording. In detection datasets, a label is known for each window that indicates whether or not a bat call is present in that window. In classification datasets, there is no label for the time windows, but only one label per time series is given, indicating the bat type present in that time series.

3.2 Data Augmentation

To train a model that is capable of both detection and multi-label classification, we propose to use an approach of *data augmentation*, where we create new data in which each window is labelled with the bat species present in it; if no bat species is indicated, no bat is assumed to be present in that window.

We propose to create such data in the following steps. First, we train a Bat detective model on a detection dataset. Subsequently, we use this model to indicate windows at which a call is present in a classification dataset. Then, we label the windows with the species of bat of the audio recording; as a result, we have a dataset with time series in which windows are labelled with the presence of individual bat species. Finally, a new dataset is created by artificially superimposing time series for different bat species on top of each other; this yields windows in which multiple bat species are present, and the identity of these species are known. On this new dataset, we propose to train a multi-label classifier, which for a given window predicts all species present in that window.

Note that the evaluation of the resulting model is complex, as we have no ground truth for the multi-label classifier. As we will discuss in more detail later, we propose to resolve this by evaluating our algorithm in 3 different ways: (1) by evaluating its performance on the augmented data; (2) by evaluating its performance as a detector on the original detection data; (3) by evaluating its performance as a classifier on the original classification data. The challenge is to find a model that performs well across these different metrics.

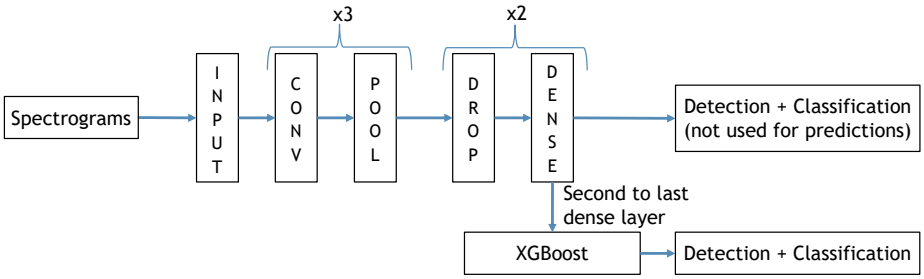

Fig. 1. The architecture of our proposed model, the CNN XGBoost, for the detection and multi-label classification of bat calls.

3.3 Proposed Architecture

One of the most common architectures used in detection and classification tasks is the simple CNN, as it is for instance the case for the Bat detective tool. However, this simple network limits the level of performance that can be reached in complex tasks due to the small number of layers that are used. A common method to overcome this limitation is to use very deep convolutional networks instead. Indeed, it is more and more common to use networks such as variations of ResNet to perform detection or classification tasks. Unfortunately, these models require a much higher amount of nodes and have a bigger computational complexity than the simple CNNs. In this paper, we propose to improve the performance of the simple CNN by separating the calculation of features from the classification algorithm and by using other efficient, but computationally lighter, nonlinear multi-label classifiers than deep neural networks.

The model we propose to fulfil these requirements is made of a simple CNN that computes new features, which are then given as input to XGBoost models that perform the classification for each of the classes in the multi-label context.

We train this model as follows. First, a CNN is trained in which the last layer consists of artificial neurons, one for each class. This model is trained as if it has to perform detection and classification. Once it is fully trained, it is applied to the augmented training data; the output of the second to last dense layer on this training data will be used as an input dataset to train XGBoost models, which will then detect and classify. The resulting architecture is presented in a schematic way in Figure 1.

To tackle the multi-label classification problem, the binary relevance [16] method was chosen to train the CNN, including the last fully connected layer.

Subsequently, one binary XGBoost model is trained for each separate class. These models predict a probability for each class.

Note however that these probabilities need to be thresholded in order to predict whether or not a specific type of bat is present in a window. We exploit this ability to pick thresholds to take into account the various requirements on our model. After all, our aim is not only to obtain good performance on the augmented data, but also on the original detection dataset. We take this require-

ment into account by choosing the thresholds such that the sum is optimised of the average multi-label classification F1-score on the augmented dataset and the detection F1-score on the detection dataset.

Our algorithm for optimising the thresholds works as follows, as inspired by [6]. We iterate over all the classes, and for each class separately vary the thresholds from 0 to 1 with steps of 0.01; if our optimisation criterion improves, we accept the new threshold; we continue this process of iterating over the classes and the thresholds until no further improvement can be found.

4 Results

This section presents the detection dataset and the classification dataset used in our experiments, as well as how they were combined to be used in a detection and multi-label classification context. It also presents the evaluation methods and the models to which our architecture will be compared on the challenges C1–C5. It then analyses to which degree our proposed model answers to those challenges and verifies whether the latter indeed performs better than the architectures used in earlier works.

4.1 Datasets

The two datasets that we have at our disposal are a detection dataset, made available by Bat detective on their website[3], and a classification dataset, provided by Natagora, a Belgian association aiming at protecting nature and collecting various data samples in Wallonia and Brussels.

Bat detective's dataset is made of bat calls recorded from four different regions in Europe between 2005 and 2011, namely in Bulgaria, Romania, UK and Norfolk. The dataset is labelled with the starting position of each call but contains no information about the species that emitted the call. Therefore, these recordings are used to train the networks that are only used for detection.

Natagora's dataset is composed of bat calls recordings taken between 2012 and 2019 at various observation sites in Belgium. The dataset contains seven different groups of bat species that cover the twenty-three species of bat that can be found in Belgium. The seven groups of bat species are Barabar, Envsp, Myosp, Pip35, Pip50, Plesp and Rhisp. The labels provided indicate for each file which group of species can be heard in it. So each file is associated with a tag, but the exact timings of the calls are not known.

To create the augmented multi-label dataset, we first needed a detection model. The detection model we used is the one proposed by Bat detective [9] and it was trained with their dataset. This detection model could then be used to find calls positions in the Natagora classification dataset. From that newly labelled dataset, calls were superimposed to generate multi-label samples.

To have a dataset representative of a real-life case, the same amount of files having superimposed calls and of files with no calls and a single group in it were

[3] `http://visual.cs.ucl.ac.uk/pubs/batDetective/`

taken for the multi-label dataset. In that way, the models learn to differentiate a call from background noise and to determine whether there is a single call or more than one at a given time of the recording. The total number of bat calls available is 94766. 8801 of them form the test set, used to compute the multi-label performance. The validation set is made of 8772 calls and the 77193 remaining ones form the training set. The validation set is used for thresholds optimisation while the training set is used for training and tuning the models.

4.2 Evaluation

As we do not have ground truths available for the multi-label classifiers, we evaluate our algorithms on three different aspects.

The first aspect is the evaluation of the model on the augmented data. This is done by computing the precision, recall and F1-score for the detection as well as for the classification of each class on the multi-label test set. A macro average of the classification metrics is also performed to have a global score of the classification performance while giving the same importance to each class.

To compute the different metrics in a multi-label fashion, 2x2 confusion matrices are used, one for each of the classes. To fill in the confusion matrices, the choice was made to consider each class independently. In other words, to update the confusion matrix of a given class, we ignore all the calls and all the predictions that are not from that class and evaluate the remaining labels as one would do in a binary detection problem. The only difference is that a detected call is considered as correctly predicted if it overlaps with an actual call position, i.e. if they are less than ten milliseconds away from each other, and not only if the prediction and ground truth are in the same window.

The second aspect that needs to be evaluated is the performance of the model as a detector. This is done on the original detection dataset, as it is our only dataset in which the call positions are ground truths. The metrics used are the same as for the multi-label evaluation, but only the detection performance is considered as we cannot say if the classes predicted are correct or not.

Finally, the models are also evaluated as classifiers on the original classification data. The ground truths from that dataset are, for each recording, the type of bat present in it but not the exact timing of the calls. Therefore, we have to look at the class that is the most predicted for each file and we verify if this matches with the ground truth class for that file. Our metric here is thus the percentage of files that are labelled with the correct majority class.

4.3 Architectures for Comparison

To make sure that our new architecture is indeed better in terms of performance than a simple CNN and smaller in terms of size than a ResNet, we compare our approach with a number of other architectures. Our proposed model and all the following architectures can be found in the GitHub of our project on `https://github.com/luciledierckx/batML`.

For calculating features, we consider ResNet50 as an alternative, as used in earlier work and adapted to receive a spectrogram as input [12], in addition to a simple CNN.

As our use XGBoost models aim to improve the performance over a simple fully connected layer, we also compare with a basic neural network in which a fully connected layer is used for both detection and classification. In our experiments, we use data with seven bat species groups; these networks have eight output classes with one for the class "not a bat" and the seven others for the seven bat species groups. We will refer to these networks as CNN8 and ResNet8, based on the convolutional neural network used.

In addition to these architectures, two baseline alternatives are also evaluated. In these, the detection and classification tasks are separated into two different networks. The recordings are thus first fed to the detection network, and the windows that are predicted as containing a bat call are then given to the second network. The latter predicts which of the seven groups of species are present in the considered window.

The decision thresholds needed for the binary relevance method are tuned for all architecture independently, as these have different strengths and weaknesses in the detection and classification.

Finally, all the networks' hyperparameters and architecture are tuned using the Hyperopt [3] library. The models were trained, tuned and tested on an i7-9800X CPU @ 3.80GHz, 32 GB RAM and an RTX 2080 SUPER GPU.

4.4 Performance on the Different Challenges

To evaluate challenge C1, corresponding to the detection task, the tuned models were evaluated on the Norfolk test set. In the first experiment, the thresholds are only optimised on the augmented dataset. The results of this evaluation are presented in Table 1a, which also gives the results obtained by the Bat detective architecture when using the same evaluation method.

From Table 1a, we see that our CNN XGBoost model has the highest precision and F1-score, and is the most balanced for the detection, but remains nine percent lower than Bat detective for all metrics. This can be explained by the fact that, even though the species present in Belgium and Norfolk are the same [4], their occurrence frequency at the two places is different, and therefore, the thresholds for each class have to be adapted. This is why we decided to use half of the Norfolk dataset as a training set in order to optimise the thresholds not only on Natagora's validation set but also on the Norfolk training set, as described earlier. This is presented in Table 1b.

Table 1b shows that the performance improves with the new thresholds, although it does not reach the performance of Bat detective. While our network is not the best in terms of F1-score, overall its performance as a detector is reasonable; the important next question is whether this performance as a detector provides benefits as a classifier.

Concerning the classification challenge C2, the evaluation of the classification is done using the recordings from Natagora's dataset that contain a single species.

Table 1. Performance of the compared architectures on Norfolk detection dataset. The best value for each threshold type and metric is highlighted.

Bat detective			**ResNet8**			**Double ResNet**			**ResNet XGBoost**			**CNN8**			**Double CNN**			**CNN XGBoost**		
Pre	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**
a) Thresholds on Natagora																				
82	86	84	55	76	64	63	87	73	67	81	73	60	85	71	64	88	74	73	77	75
b) Thresholds on both Natagora and Norfolk																				
82	86	84	72	78	75	68	88	77	74	80	77	75	83	79	79	85	82	77	82	79

Table 2. Percentage of files for which the model predict the correct species. The model is considered as predicting the correct class for a file when it is the class predicted the most for that file. The best value for each threshold type is highlighted.

	ResNet8	**Double ResNet**	**ResNet XGBoost**	**CNN8**	**Double CNN**	**CNN XGBoost**
Thresholds on Natagora	78	76	84	84	79	85
Thresholds on both Natagora and Norfolk	79	36	85	74	77	86

The percentage of files labelled with the correct majority class is reported in Table 2 for both threshold types.

From Table 2 we observe that, no matter the type of threshold used, our CNN XGBoost model has the best classification performance. A noticeable aspect is that, for our proposed architecture, the number of correctly predicted files do not decrease with the new thresholds and is even a bit higher here. This shows that the model has a strong classification and is not too quickly disturbed by a change of distribution in the number of class samples.

Now that the basic cases of detection and classification have been assessed, the multi-label classification and detection, i.e. challenges C3 and C4, must be evaluated on the artificially created multi-label dataset. The performance of each architecture for the detection task is reported in Table 3 for the two different thresholds. The classification performance of the different models for each of the bat species groups and the macro average over the seven classes is presented in Table 4 for the new thresholds. The performance with the thresholds tuned on the augmented data only are not displayed but are higher for all models.

Looking at the detection performances from Table 3a, we see that the network the most fitted to detect the bat calls in the recordings is the CNN8 as it has the best F1-score and best recall. Our CNN XGBoost model has an F1-score of two percent less so its detection performance is not too low compared to the best architecture. It can also be noticed that, for the detection, the models using ResNet all have a lower F1-score than those using CNNs. And in a general manner, we see that all the models perform better in terms of precision than in terms of recall. That indicates that the models tend to miss some low quality,

Table 3. Performance of the compared architectures for the detection task on the augmented data set. The best value for each threshold type and metric is highlighted.

ResNet8			**Double ResNet**			**ResNet XGBoost**			**CNN8**			**Double CNN**			**CNN XGBoost**		
Pre	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**
a) Thresholds on Natagora																	
77	73	75	76	73	74	76	74	75	81	76	78	86	69	77	78	74	76
b) Thresholds on both Natagora and Norfolk																	
81	67	73	78	74	76	82	67	74	85	65	74	90	60	72	83	68	75

Table 4. Performance of the compared architectures for the classification of each species group as well as the macro average over all the classes with the thresholds optimised on both Natagora and Norfolk. The best F1-score for each class and for the macro-average is highlighted.

	ResNet8			**Double ResNet**			**ResNet XGBoost**			**CNN8**			**Double CNN**			**CNN XGBoost**		
	Pre	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**	**Pre**	**Re**	**F1**
Barbar	71	88	78	97	56	71	87	61	72	90	73	81	90	80	85	91	52	66
Envsp	89	87	88	88	51	64	88	93	90	92	81	86	94	82	87	91	92	91
Myosp	94	83	88	92	53	67	95	90	92	98	80	88	94	84	89	97	87	92
Pip35	73	71	72	60	53	57	71	78	74	73	72	73	74	72	73	82	76	79
Pip50	51	89	65	48	23	31	62	88	72	47	92	63	52	90	66	56	93	70
Plesp	87	70	78	85	43	57	81	70	75	95	67	79	93	63	75	82	68	75
Rhisp	87	83	85	10	100	19	92	81	86	86	86	86	93	73	82	95	84	89
Avg	79	81	79	69	54	52	82	80	80	83	79	79	84	78	80	85	79	80

less obvious calls, and recognise more easily when a sound is not a bat call. When using the new thresholds, we can see in Table 3b that for most of the models, the F1-score only slightly decrease, but the gap between detection and recall gets higher. In this case, it is the double ResNet model that performs best but our CNN XGBoost remains the second-best performing model.

In terms of average multi-label classification performance, it can be observed in Table 4 that with the Norfolk-optimized thresholds our CNN XGBoost model is one of the models having the best average F1-score. This model also has the best F1-score for most bat species groups. For the classes where it is not the best, it is always quite close to it, except for the Barbar class. In general, we observe that the networks using CNNs tend to perform better than the ones composed of ResNets.

The last challenge to evaluate is the challenge C5, concerning the computational resources used by the architectures to perform a prediction. The resources used by the models are presented in Table 5. For the neural networks, the number of parameters is used as a metric. For models based on XGBoost, in addition, the number of nodes visited to produce a prediction is estimated by summing up the depths of all the trees in the model.

Table 5. Computational resources of the different networks in terms of number of parameters for deep networks and longest paths in the decision trees for XGBoost.

	ResNet8	**Double ResNet**	**ResNet XGBoost**	**CNN8**	**Double CNN**	**CNN XGBoost**
Size	23 571 272	47 128 201	23 794 880	569 048	1 185 521	806 480

From Table 5 we can conclude that the models using ResNet require much larger computational resources than the ones using simpler CNNs. Our CNN XGBoost model is heavier in terms of computational resources than the basic architecture made of a single CNN, but the difference is much smaller than with ResNet. We can therefore say that our model is light compared to others state of the art architectures.

From the evaluation of challenges C1–C5, we see that our proposed model is the best performing model for many of the evaluated challenges, even though not for all of them. However, in those cases, our CNN XGBoost architecture is always reasonably close to the best model. Therefore, we can say that among all the proposed models, it is the one that is the most well-balanced and is hence best fitted for the problem of multi-label detection and classification using detection and classification data solely.

5 Conclusion

We have addressed the need for an automated tool performing both detection and multi-label classification for bat calls in recordings as well as the challenge of the limited datasets available. We presented a data augmentation method and implemented a new architecture responding to that need. We showed that it has good detection and classification performance despite the fact of being trained on an augmented dataset. We then verified that our architecture delivered better performance than the state of the art models for multi-label acoustic bat monitoring tasks. Finally, we verified that the computational resources required by the model are reasonable. From these various evaluations, we concluded that our CNN XGBoost model was the one responding the best to the multiple challenges of the multi-label detection and classification. Future work includes implementing our model on a portable recording device, and studying its use on a more fine-grained bat classification tasks.

Acknowledgements We would like to express our gratitude to Natagora and the Plecotus team for the large amount of labelled bat call recordings they shared with us. We would also like to thank Bat detective for the data they have made available and their detection tool, used as starting point in this work. We thank Olivier Bonaventure for joining discussions on this project.

References

1. Armitage, D.W., Ober, H.K.: A comparison of supervised learning techniques in the classification of bat echolocation calls. Ecological Informatics **5**, 465–473 (2010)
2. Barataud, M.: Acoustic ecology of European bats. Species Identification and Studies of Their Habitats and Foraging Behaviour. Biotope Editions (2015)
3. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on Machine Learning. pp. 115–123 (2013)
4. Border, J.A., Newson, S.E., White, D.C., Gillings, S.: Predicting the likely impact of urbanisation on bat populations using citizen science data, a case study for norfolk, uk. Landscape and Urban Planning **162**, 44–55 (2017). https://doi.org/https://doi.org/10.1016/j.landurbplan.2017.02.005
5. Chen, X., Zhao, J., Chen, Y.H., Zhou, W., Hughes, A.C.: Automatic standardized processing and identification of tropical bat calls using deep learning approaches. Biological Conservation **241** (2020). https://doi.org/10.1016/j.biocon.2019.108269
6. Fan, R.E., Lin, C.J.: A study on threshold selection for multi-label classification (2007)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
8. Jones, G., Jacobs, D., Kunz, T., Racey, P.: Carpe noctem: The importance of bats as bioindicators. Endangered Species Research **8**, 93–115 (2009). https://doi.org/10.3354/esr00182
9. Mac Aodha, O., et al.: Bat detective - deep learning tools for bat acoustic signal detection. In: PLOS Computational Biology (2018). https://doi.org/10.1371/journal.pcbi.1005995
10. Runkel, V., Gerding, G., Marckmann, U.: The Handbook of Acoustic Bat Detection. Pelagic Publishing (2021). https://doi.org/10.53061/XDDW7329
11. Rydell, J., Nyman, S., Eklöf, J., Jones, G., Russo, D.: Testing the performances of automated identification of bat echolocation calls: A request for prudence. Ecological Indicators **78**, 416–420 (2017). https://doi.org/10.1016/j.ecolind.2017.03.023
12. Schwab, E., Pogrebnoj, S., Freund, M., Flossmann, F., Vogl, S., Frommolt, K.H.: Automated bat call classification using deep convolutional neural networks (2021)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
14. Tabak, M., Murray, K., Lombardi, J., Bay, K.: Automated classification of bat echolocation call recordings with artificial intelligence (2021). https://doi.org/10.1101/2021.06.23.449619
15. Voigt, C.C., Kingston, T. (eds.): Bats in the Anthropocene: Conservation of Bats in a Changing World. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-25220-9
16. Zhang, M.L., Li, Y.K., Liu, X.Y., Geng, X.: Binary relevance for multi-label learning: an overview. Frontiers of Computer Science **12** (2017). https://doi.org/10.1007/s11704-017-7031-7
17. Zualkernan, I., Judas, J., Mahbub, T., Bhagwagar, A., Chand, P.: A tiny cnn architecture for identifying bat species from echolocation calls. In: 2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G). pp. 81–86 (2020). https://doi.org/10.1109/AI4G50087.2020.9311084