CoordinateNet for BirdCLEF2023 problem

Stanislav Novikov
stanislav.novikov@gmail.com
Project on GitHub:

Abstract

This work brings to your attention an original solution of the BirdCLEF2023 problem which exceeds the results of [1] and BirdNET [2][1] with architecture simpler than EffitientNet b0. The achieved validation accuracy is 0.89 after 20 epoches (vs 0.75 of [1]) for 264 birds.
In this article I demonstrate a new approach for spectrogram classification and offer my readers to go far this way for better results. As you will see later there is a lot of room to further improve speed/accuracy ratio.

Introduction

This document describes the solution of the BirdCLEF2023 problem based on 1D CNN with a wide kernel. It contains 3 chapters. Chapter 1 describes data preparation for training and testing. Chapter 2 explains the ideas and architecture of the CNN. Chapter 3 describes work done for optimizing the neural network.

Chapter 1. Audio files preprocessing

File markup

Since BirdCLEF2023 rules assume estimation of 5s audio intervals 5s samples seem the best choice for training. Quick approach based on Efficient B0 revealed that one of reasons for decreasing prediction accuracy is empty intervals, i.e intervals estimated as labels but with no birdsong/call.
The second inference is concerned with noise structure. As a rule for BirdCLEF2023 recordings with rating >= 3 noise has a low frequency component and one or two middle/high. Low frequency band of noise lies in interval 0 - 1500 Hz, other band(s) takes place on 4000 Hz up to 9000 Hz.
Low frequency noise sources are industrial noise, microphone, wind, crackling branches and so on. Other bands are formed by mass birds singing and sometimes insect sounds.
Since the noise is concentrated in bands, there are frequencies of birdsong lying out of these bands, i.e. with low noise.
Looking ahead this observation led me to the idea of splitting the signal into bands and analyzing each band separately. In this approach noise influences only part of frequency bands whereas analyzing a whole mel spectrogram at once will mix noise and signal and may require more efforts for training.
Returning to empty intervals, to avoid them the following algorithm is realized:

[1] Comparison with BirdNet is a complex task. Nevertheless we can orient on the archived accuracy for primari labels - 0.78 and complexity - 157 of layers.

1. For initial audio recording with no secondary label mel spectrograms (n_mels = 128, sr = 32000, window = 1024, hop = 400) are built. So for a mel spectrogram we have numpy array of shape: (128, n), where $n = \frac{length\ of\ audio - window}{hop}$

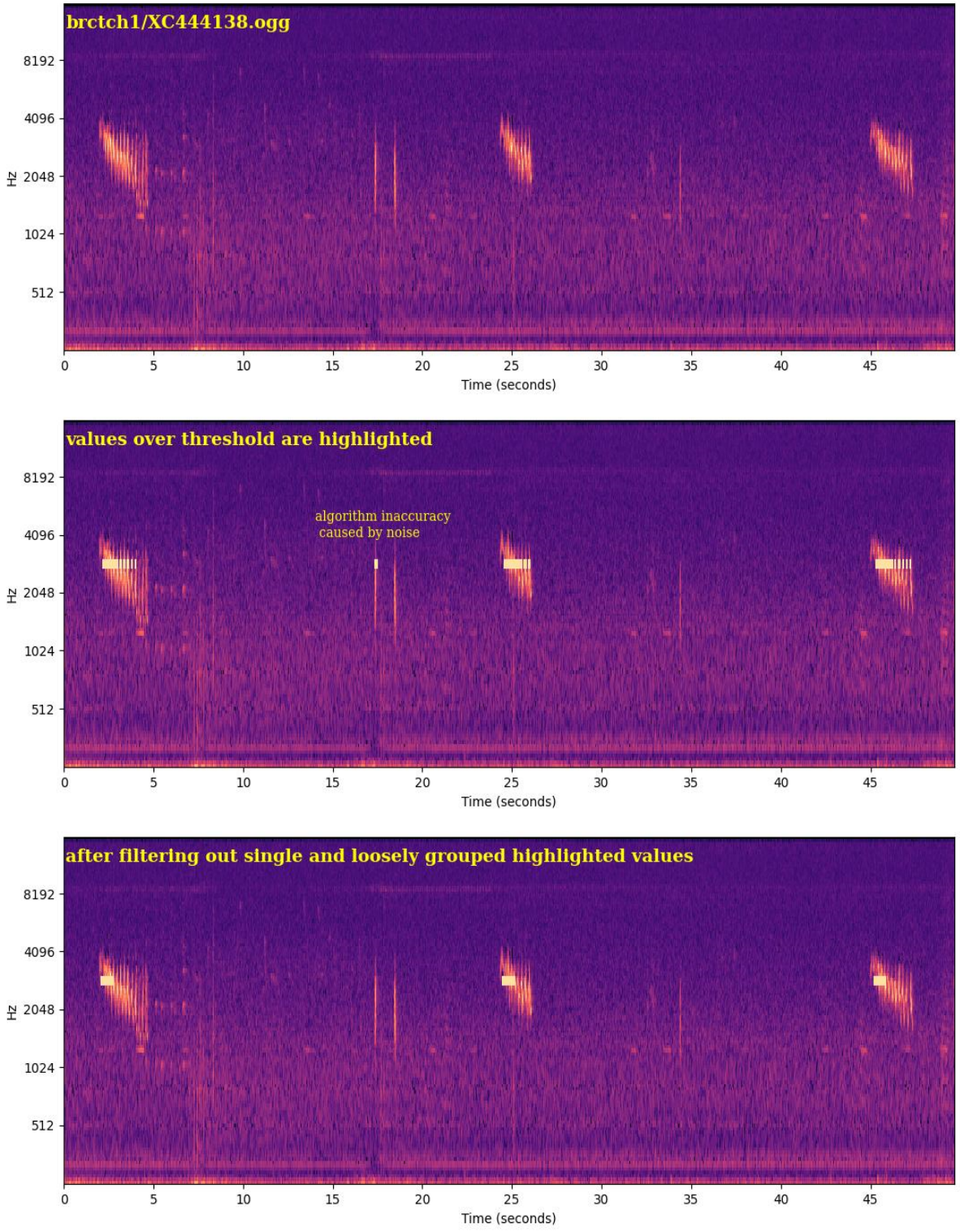

Fig.1 Example of audio file marking up

The first picture is the mel spectrogram of brctch1/XC444138.ogg record. Values over threshold = 0.7 are highlighted on the second picture. At moment t ~ 17s we can see two narrow noise stripes. Left

one is upper the threshold.The third picture shows final marks/nicks after filtering out single and loosely grouped highlighted nicks.

2. Each spectrogram is divided by 32 stripes per 4 rows in each stripe. Thus we consider spectrogram as 3D array: 4×32×n
3. Each stripe sums, i.e. we reduce the 4×32×n array to 32×n summing up values of 1st dimension4. Mels range in which bird's singing is most pronounced is selected for each bird For example for afecuc1 this range seems to be 2000 - 4000 Hz or between 15th and 21st stripes on mels dimension.5. Maximum values are selected in the stripes of the range. Namely, we find top3 maximums in each stripe of the range, sum them and find the stripe with the maximum sum.
6. In the selected stripe we find indexes of values which more then threshold.
7. If the indexes form tight groups the algorithm takes groups and ignores separate, single indexes. That allows further reduction of noise outliers. For details see script "mark_up_mels_v1.0".
8. The algorithm considers the selected indexes as traces (nicks) of desired signal and takes them into account dividing audio by 5s intervals.
It is worth mentioning here that more strict thresholds lead to reducing the number of selected empty 5s intervals and growth of losses of correct 5s intervals (samples).
Examples of the algorithm results are depicted on Fig.1
The above algorithm of marking up does not fit for recordings with secondary labels since it is not able to distinguish different birds. But probably high accuracy of the proposed neural network allows us to mark up secondary labels correctly. This idea will be described later in the next post.

Sampling

Single condition to select audio files for training/testing is rating >=3. Marked up audio recordings with no secondary label are divided into 5s intervals. If an interval contains at least one nick, we take it as a sample. Audio files with secondary labels are split into 5s intervals as is and we treat all intervals as samples of a corresponding label.

Three variants of training/validation landscapes

Besides the main results presented in Abstract, to show sensitivity of our approach to timeshift augmentation and give an estimation of its accuracy from above I consider three variants of training/validation pairs of datasets (landscapes). These variants differ in ways of augmentation and versions of validation datasets.
In the beginning audio recordings are marked up (see previous section) and cut into initial samples. Number of initial samples is 92000.
The first version of the validation dataset is formed by splitting the initial samples for training/validation. Samples selected for validation are cut from audio records before timeshift augmentation and have no chance to leak into the training process. I.e. cutting validation samples from audio recordings we create new recordings where validation intervals are omitted.
The second version of the validation dataset is built by the split after augmentation. It is methodically incorrect but we may consider validation accuracy in this case as an estimation

from above. Perhaps we could get such accuracy if we had a large amount of audio recordings for each bird. Imagine a situation when we have several audios for every kind of bird call/song/etc. If so, even correctly selected validation samples will differ from training ones by time offset and noise only. We have a similar situation when split input data after massive time offset/noise augmentation, i.e. validation dataset differs from training by time offset and noise.
Now let's describe the issue of augmentation. The total length of audio recordings is unevenly distributed among the 264 birds (see Fig...). To align this distribution and avoid tail effect I augment poorly represented birds up to 400 samples and restrict birds with lots of audio recordings by 2500 samples.
With this in mind we perform augmentation in two ways. Augmentation of the first way increases the number of samples higher than 400 for poorly represented birds only. In the second way we 3 times (2 more samples for 1 initial) increase the number of samples for a bird if the number of initial samples lies in the range of 400 - 2500. And the upper limit of 2500 samples is applied after the augmentation. Table 1 summarizes landscapes and results obtained

Landscape	Description	Number of samples	Validation accuracy, after 20+ epoches
#1	validation dataset of the first version, augmentation of the 2nd way	237 000 samples for training, 9200 for validation	0.8900
#2	validation dataset of the first version, augmentation of the 1st way	165000 samples for training, 9200 for validation	0.8003
#3	validation dataset of the 2nd version, augmentation of the 2nd way	220 000 samples for training, 24 000 for validation	0.9512

Table 1 Landscapes for training/validation

Result of the 1st landscape is presented as the main. Results on the 1st and 2nd landscapes show dependence of our approach to timeshift augmentation. That is expected (see explanations in the Conclusions section). Results of the 3rd landscape we consider as estimation[2] from above.
Since in the second landscape when we take validation and training samples from an audio recording of one bird, samples are already "time shifts" to each other. However, even one bird repeats no identical calls, and with a small amount of data, the neural network cannot build a generalized distribution.

[2] Increasing the variety of songs/calls, as well as increasing the number of birds, may require scaling the model. Scaling is considered here in detail.

Augmentation

Minimum and maximum numbers of samples per bird are chosen based on the time limit of epoch on the one hand and the need to avoid the tail effect on the other. As mentioned above, the selected minimum and maximum numbers are 400 and 2500, respectively.
If, after dividing recordings into samples, the number of initial samples is less than the minimum allowed, they are augmented by 3 times iteratively, until their number exceeds the minimum. Augmentation algorithms are detailed in Appendix B.
Two band noise is added to all augmented samples. Noise params are chosen randomly (see Noise generation in Chapter 1)
For some birds with extra small total length of recordings it is necessary to augment samples depending on the situation. Appendix A lists appropriate scripts for several birds.
Approximate ratio between samples with/w.o. secondary labels is about 14%

Samples after augmentation for brctch1/XC444138.ogg (see Fig.1) are shown on Fig.2

Noise generation

Noise spectrum S_n is described by the formula:

$$S_{low} = \begin{matrix} 1 - tanh(0.75\frac{f}{f_b}),\ f < f_b \\ 1 - tanh(0.75)\frac{f_b}{f}),\ f > f_b \end{matrix} \qquad S_{up} = e^{-\frac{1}{2}\left(\frac{f-f_b}{\sigma}\right)^2} \tag{2}$$

$$S_n = \frac{S_{low}}{\sqrt{S_{low}^2}} + \gamma\frac{S_{up}}{\sqrt{S_{up}^2}}$$

where all parameters are chosen randomly and uniformly in the following ranges:

$f_b \in [250,\ 750], Hz$

$f_u \in [4000,\ 9000],\ Hz$

$\gamma \in [0.1,\ 0.3]$

$\sigma \in [0.04,\ 0.08]$

These ranges were chosen based on visual similarity with “average” mel spectrogram and should be tuned for real conditions of the NN application
Noise variance is calculated as fraction of signal variance in range $[0.1,\ 0.3]$

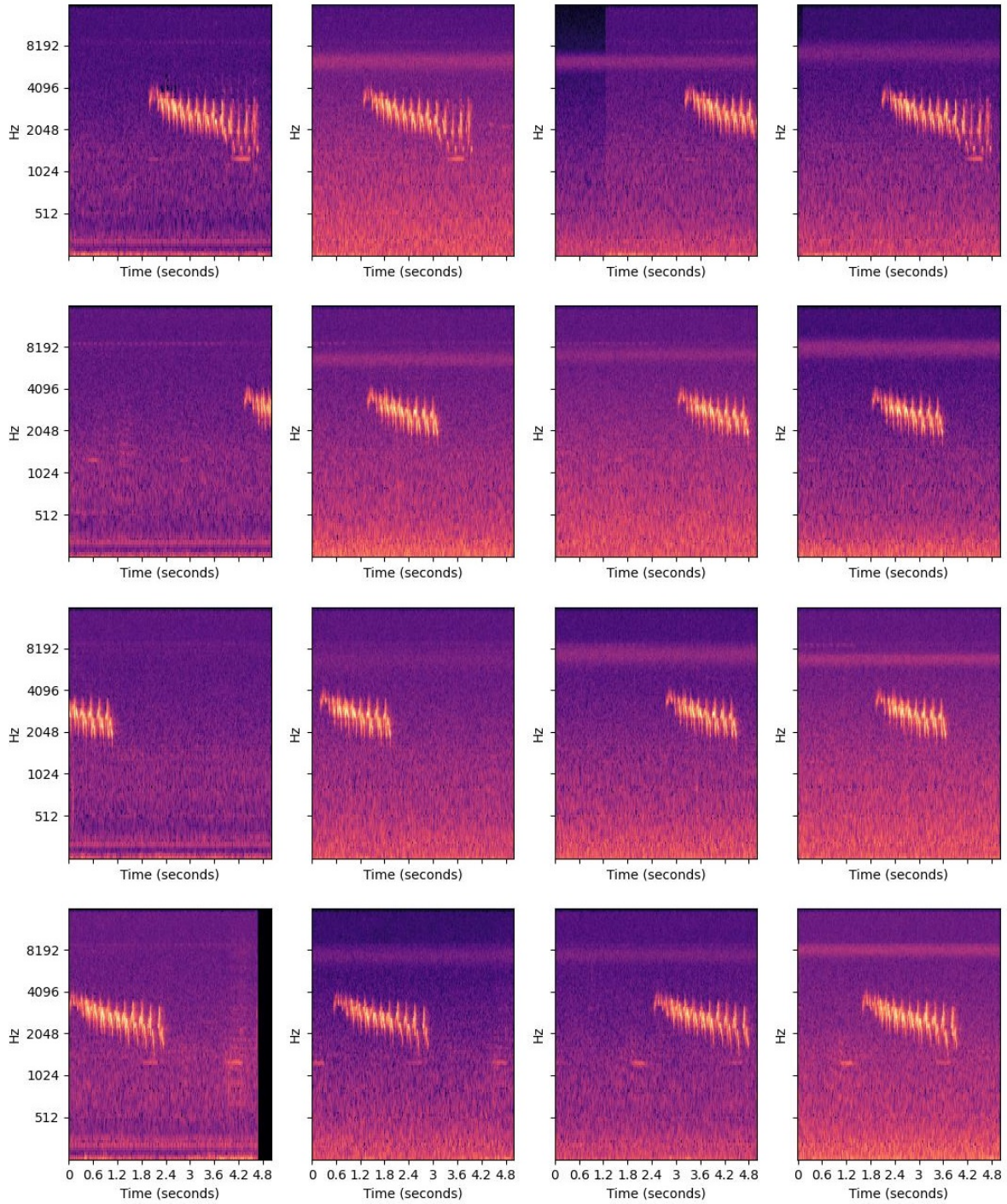

Fig.2 Samples from brctch1/XC444138.ogg (see Fig.1)

Each row consists of 5s samples based on one group of nicks.The first row corresponds to the first group of nicks depicted on Fig 1 about t ~ 2s3. The second and third rows are formed from the neighborhood of the second group of nicks located at t ~ 25s. Since one part of this group belongs to interval 20 - 25s and other lies in interval 25 - 30s the algorithm produces samples based on both 20-25s and 25-30s intervals.The fourth row corresponds to the third group of nicks at t ~ 46s.
The algorithm takes into account only whole 5s intervals and discards a remainder less than 5 seconds. This feature of the algorithm would leave the third group of nicks without consideration.
So for a sake of presentation, audio of the example was augmented to 50 seconds
Noise is added to all samples except for ones of the first column.

Chapter 2

CoordinateNet

Idea of "coordinate" net comes from the assumption that net architecture for spectrograms classificator has to differ from one for image recognition.
It is based on a couple of thoughts. The first one: location of a signal relative to the frequency axis on a spectrogram is very important. CNNs are good in object recognition but may lose accuracy of the signal positioning. To mitigate this risk I divide a mel spectrogram on stripes and analyze each stripe separately.
The second thought concerns the mathematical definition of convolution. Given frequency appears in a song of each bird according to a certain pattern. If our net can catch these patterns for a range of frequencies we would solve the problem.
Assume birds produce only one frequency and the mel spectrogram reduces to 1d array. Let the pattern be described by function $F\ (t, b)$, where t - time, b - bird. If NN is able to mimic $F\ (t, b)$, where b is a given bird, then convolve $F\ (t, b)$ with input $I(t)$ we get maximum response for $I(t) = F(t - t_0, b)$ where t_0 is an arbitrary offset. We consider here mathematical definition of convolution:

$$H = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(t)F(t - t_0, b)dt\ dt_0 \quad (3)$$

CNN built on Conv1D layers with a wide kernel is the best candidate to carry out such convolution. But the size of the kernel must be big enough to catch the most meaningful part of $F\ (t, b)$.
Let's pretend we have several sequential Conv1D layers and Maxpool1Ds between. The first Conv1D has the kernel wide enough to cover F(t, b). Its output feeds to Maxpool1D with strides = n and then to second Conv1D with kernel n times smaller than first and so on up to single value. In this case we probably get output that has maximum for a bird we would train this NN for.

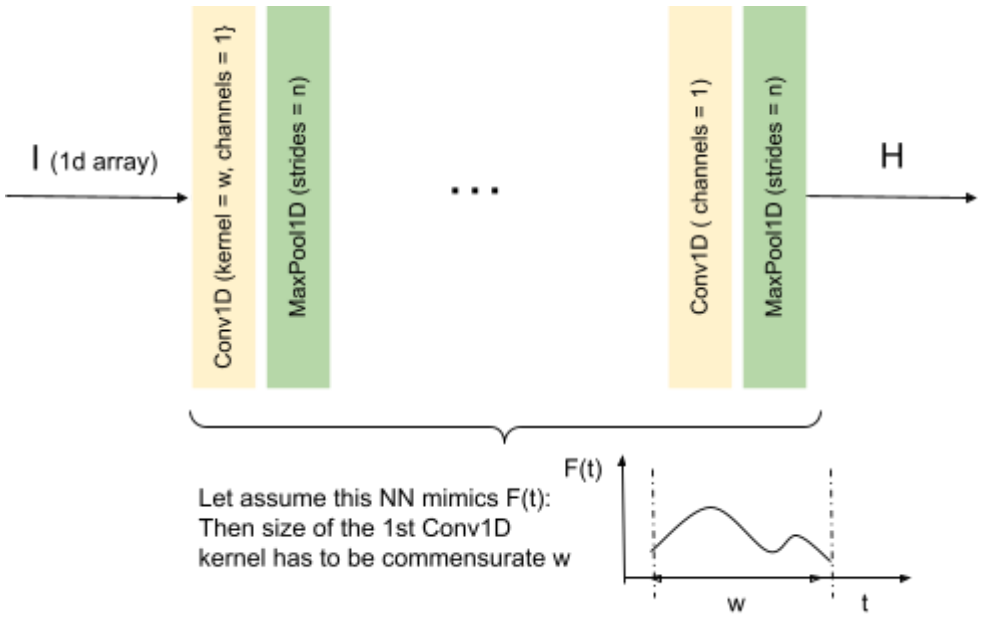

Fig.3 Hypothetic NN for a single frequency birdsongs

Since the input mel spectrogram has a size of 128×400 we can generalize the approach up to 128 mels. For each mel I introduce the structure described above (see Fig.3). The simplest way to generalize our approach to several mels is to introduce channels.
Also the functionality of conv1d in tensorflow allows extending batch shape. The resulting architecture is quite compact because it deals with the mels dimension as a part of the batch shape. But we keep the mels dimension in the batch shape for conv1d layers only and consider it as part of the input shape for the Flatten layer again.
To allow the net to distinguish different birds we just need more channels. At first glance, it may seem that the number of channels should be too large - 128 × 264, but as shown in this work, channels are required by 3 orders of magnitude less. This difference can be explained by two reasons. We leave a vector instead of one value (H) for each channel and feed this vector to FC. And it seems our intuition is not complete. The network can learn to evaluate several different characteristics of a spectrogram and then determine its class from the resulting set of characteristics.
The same idea for the time axis will give us a chain: conv1d -> maxpool1d -> conv1d ... for up to 400 stripes parallel to the mel axis. It does not have the same obvious intuition, but at least it can give to our neural network accurate information about the mel/frequency distribution of birdsong over the time axis.
Wide kernel here seems able to catch mel distribution faster and more precisely than a small one. "Faster" means less number of sequential conv1d layers and possibly less effort to train. In contrast to conv2d layers which operate with a whole mel spectrogram and may lose accuracy of signal positioning and form, conv1d and 1d stripes provide precise positioning and form of signal along the axis divided by the stripes.
It is important notice:
1. Width of the stripe does not have to be equal to 1. We can take wider stripe and reduce to 1d calculating average by width
2. We keep the number of stripes unchanged up to a Flatten layer (see Fig.4).

Flattened output vectors obtained from stripes along mel and time axes are concatenated and fed to FC.

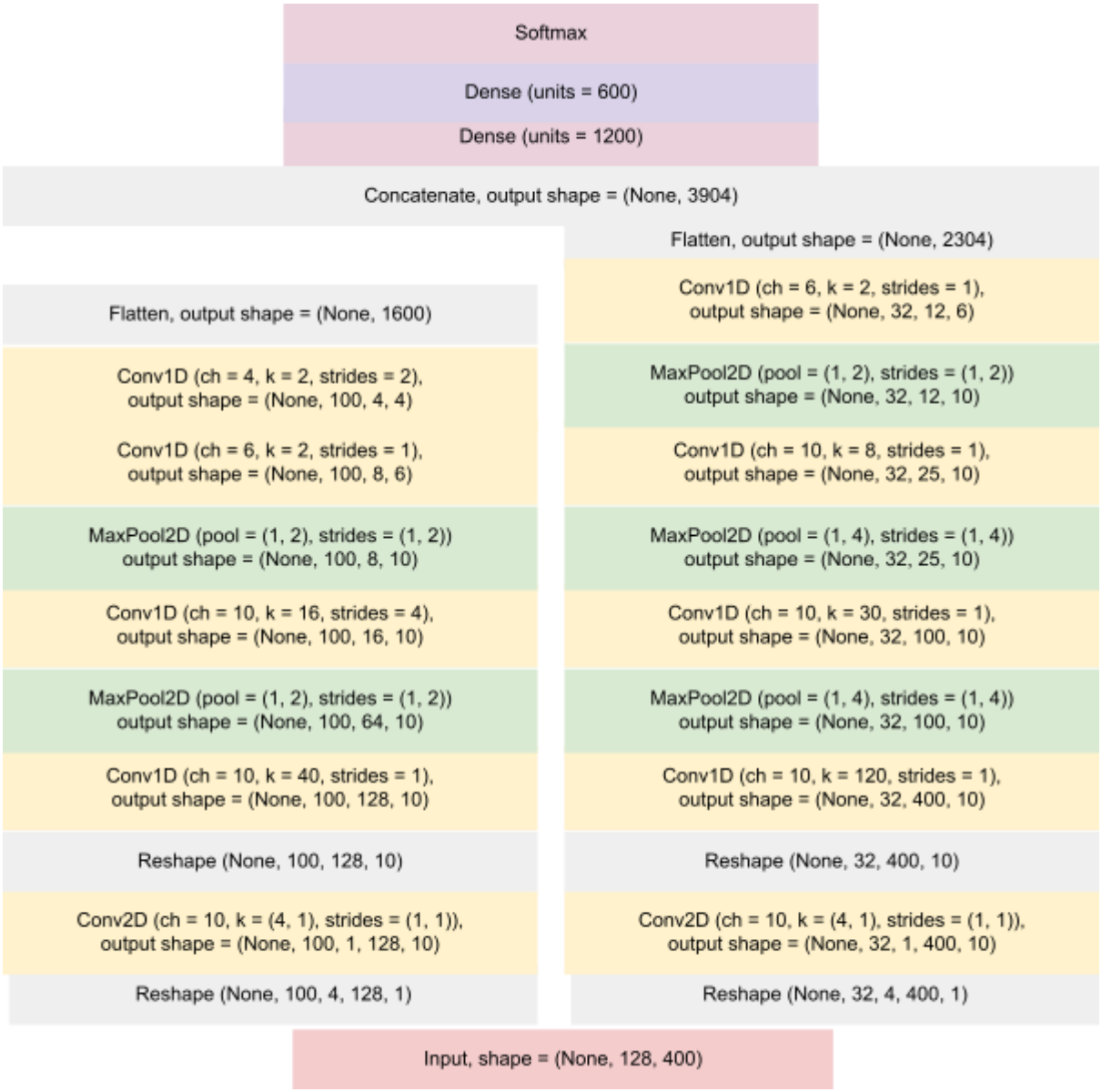

Fig.4 Architecture of the backbone "coordinate" net

The first Reshape layers form stripes as a new dimension. Conv2Ds reduce the dimension and introduce channels. Kernels of first Conv1Ds are 120 for the time dimension and 40 for the mel dimension respectively. For the time dimension of size 400 (5s) such kernel covers a signal of 1.5s length.
In Tensorflow Conv2D and Conv1D support extended batch shape. For both branches the batch shape consists of 2 dimensions: (batches, stripes). Stripes of batch shape remain unchanged up to Flatten. This allows CN to distinguish information between stripes.

BatchNormalization and Dropout layers are omitted (see SoundClassifier v6.4.4.ipynb on GitHub for the full model). Total number of trainable parameters is ~5.5 mln

Architecture of the original/backbone "coordinate" net (CN) is depicted in Fig.4
First hyperparameters of "coordinate" net architecture such as width of stripes, kernel sizes, number of conv1d layers, number of channels - including before Flatten layers -, number and

size of FC layers are chosen in performance tests. Each test was performed for 600 first batches (batchsize = 64) of epoch #1 on about 420 000 samples of 264 birds. I compared accuracy obtained after 600 batches and time estimation needed for a whole epoch. The working time of 1 epoch was restricted by 2 hours 30 minutes[3].

Chapter 3

Note: All comparisons below are done for the third landscape as the most observable.

The above version of CN did not show excellent accuracy for 264 birds but achieved impressive validation accuracy = 0.96% for 10 birds. Such results forced me to think about optimization and scaling up.

Optimization

First of all I looked at possible alternative ways of feature extraction in parallel to the CN and compared the result with the backbone version. Number of birds is 10. The main idea is to improve accuracy of CN by expanding its features and as possible not lose performance. Comparison was carried out for validation accuracy after epoch #1.

The following in parallel add-ons were tested:
1. 2D FFT of melspectrogram
2. MFCC, 24 coefficients
3. MHA
4. channel attention as described in [3]
5. large kernel

2D FFT

2D FFT transformation of input melspectrogram 128×400 is reduced to size 32×100 by MaxPooling2D. The result is cutted ([2:, 4:]) to (30, 96) for noise removing and feeds to:
1. Conv2D, kernel = 30×12, 12 channels, strides = (30, 12), padding = same. Output dimension: (1, 8, 12)
2. Conv1D kernel = 2, 12 channels, strides = 1, padding = same
3. Conv1D kernel = 2, 12 channels, strides = 1, padding = same
4. Flatten
5. 96 features are concatenated with the backbone features before first FC.

This add-on achieved the worst validation accuracy of epoch #1: < 0.7 vs 0.8520 of the backbone CN.
There is no evidence that 2D FFT is able to add new information to the CN. 2D FFT net with almost 32×100 features that are feeded up to Dense of 1600 units and so on (7 mln params in total) after 3 epoches achieves vac = 0.03.

MFCC (with and w/o attention)

[3] This project has been done (including training) on my notebook: CPU - AMD Ryzen 5 4600U, 16.0 GB RAM. So time estimations here are quite large.

MFCC transformation of size 24×400 (24 coefficients) feeds to:
1. Conv2D with 24 channels, k = (24, 120), strides = (24, 4), padding = same. Output dimension: (1, 100, 24)
2. Conv1D: 24 channels, k = 30, strides = 1, padding = same.
Output dimension: (1, 100, 24)
3. MaxPool2D: pool_size = (1, 4), strides = (1, 4), padding = same
Output dimension: (1, 25, 24)

Attention [2]:

- reducing input (1, 25, 24) to 2×(1, 1, 24) as max and average
- Flatten: 48 features
- Dense: units = 24, output shape = (1, 1, 24)
- Output of MaxPool2D from point 3 is multiplied by A

4. Conv1D: 24 channels, k = 8, strides = 1, padding = same
Output dimension: (1, 25, 24)
5. MaxPool2D: pool_size = (1, 5), strides = (1, 5), padding = same
Output dimension: (1, 5, 24)
6. Flatten
7. 120 features are concatenated with the backbone features before first FC

MFCC w/o attention vac = 0.7599, with attention vac = 0.8207. MFCC with attention performs better but results of both options are worse than CN (vac = 0.8520).

MHA

MHA is implemented along the mel axis. Input mel spectrogram of size 128×400 is considered as a sentence of 128 "words". Dimension of "word" is reduced from 400 to 40 by Reshape (128, 40, 10) and tf.math.reduce_mean (axis = -1). Params of MHA: heads = 4, key_dim = 10.
MHA output (none, 128, 40) is reduced to (none, 128) by choosing maximum value. 128 features are concatenated with the backbone features before first FC. MHA option achieved vac = 0.8450.
Perhaps this direction is worth additional research.

Channel attention

Two attention mechanisms described in [3] are built in CN. The first one is located in processing of stripes along the mel axis after MaxPool2D pool_size = (1,2), strides = (1, 2), padding = 'same'. Output dim (100, 64, 10). And another one is embedded in processing of stripes along the time axis after MaxPool2D pool_size = (1,4), strides = (1, 4), padding = 'same'.

The first attention mechanism includes:
- reduction of input tensor (100, 64, 10) to 2×(1, 1, 10) as max and average
- Flatten: 20 features
- Dense: units = 10, A = (1, 1, 10)

- input tensor of shape (100, 64 10) is multiplied by A

The second one is made by analogy.
Validation accuracy after epoch #1: 0.7751.
Probably, the efficiency of the mechanism will grow with the number of channels. As you will see in Chapter “Scaling up” , the rise of birds’ number without accuracy losses requires enlargement of channels’ quantity. Thus repeating this experiment on CN for 264 birds looks not bad.

Large kernel

This option’s goal is to try an alternative version of CN with an extra large kernel and no stripes.

Mel spectrogram of size (128,400) feeds to
1. Conv2D: 39 channels, k = (128, 120), strides = (128, 2), padding = same. Output dimension: (None, 1, 200, 24)
2. Reshape (None, 200, 39). Output dimension: (None, 200, 39)
3. MaxPool1D (pool = 2, strides = 2, padding = 'same'). . Output dimension: (None, 100, 39)
4. Conv1D: 39 channels, k = 40, strides = 1, padding = same. Output dimension: (None, 100, 39)
5. Conv1D: 39 channels, k = 40, strides = 1, padding = same. Output dimension: (None, 100, 39)
6. Conv1D: 39 channels, k = 40, strides = 1, padding = same. Output dimension: (None, 100, 39)
7. Flatten. Output dimension: (None, 3900)
8. Dense: units = 1200
9. Dense: units = 600
10. 600 features are concatenated with the backbone features before softmax

As all above this version showed accuracy worse than the backbone (vac = 0.8429 vs 0.8520). Although the difference is not crucial, this option uses significantly more channels (39 vs 10). At first it decreases performance and at second the number of channels is a very valuable resource for scaling up as we will see later.
So the idea of dividing a mel spectrogram into frequency bands to win from uneven distribution of noise, set out in Chapter 1, may be correct.

Scaling up

The backbone CN showed high validation accuracy of 96% for 10 birds and significantly worse vac ~ 87% for 60 birds. It is reasonable to assume this is caused by insufficient capability of the CN to distinguish between mel spectrograms of an enlarged number of birds. I doubled the number of channels to 20 and validation accuracy of CN for 60 birds raised up to 95%. But one more doubling number of channels for both branches to scale up for 264 birds brings down performance.
According to [4] scaling up has to involve, besides number of channels (width), also resolution and depth. What is resolution in this case is a good question. Strictly speaking resolution of mel spectrograms is fixed and the most evident candidate for this role is width

of stripes. Since straight following to [4] is not possible here, I carried out some experiments to scale up the CN for 264 birds taking into account the main idea of [4]. Table 2 figures out accuracy achieved after 600 batches (batchsize = 64) of first epoch for different options and time estimation needed for a whole epoch. The total number of mel spectrograms for training is about 220,000.
Description of option is made in format: {number of stripes on time axis/along mel axis} - {number of channels processing stripes on time axis/along mel axis}, {number of stripes on mel axis/along time axis} - {number of channels processing stripes on mel axis/along time axis}. For example, this description of the backbone CN (see Fig.3) is 100-10, 32-10, CN for 60 birds: 100-20, 32-20.
Further doubling channels 100-40, 32-40 ruins performance, and Table 2 shows attempts to find options between 100-20, 32-20 and 100-40, 32-40 with high accuracy for 264 birds. In the attempts I balance the number of channels and stripes to find optimal relationships. If roughly assuming that doubling of stripes is equal dobling of channels, I increase channels up to 40 regarding 100-20, 32-20 on one branch only.

Option	Accuracy	Time for epoch
100-20 and 64-20	After 302 batches: 0.1879 After 600 batches: 0.2855	11-12 hours
100-20 and 32-40	After 302 batches: 0.1639 After 600 batches: 0.2618	12+ hours
200-10 and 128-10 1st conv2d of the branch processing striped along time axis is dropped	After 640 batches: 0.2481	less 4 hours
200-10 and 64-20	After 302 batches: 0.1807 After 600 batches: 0.2803	10-11 hours

Table 2 Different options to scale up CN for 264 birds

This table is not complete. At least we should try option: 200-10 and 128-10
and replace the dropped conv2d by conv1d with 10 channels.

The variant 200-10 and 64-20 was chosen to check this approach on long training distance. This option is not optimal. However it is good enough to achieve validation accuracy of 95% for 264 birds. The current architecture of the "coordinate" net is depicted in Fig.5.

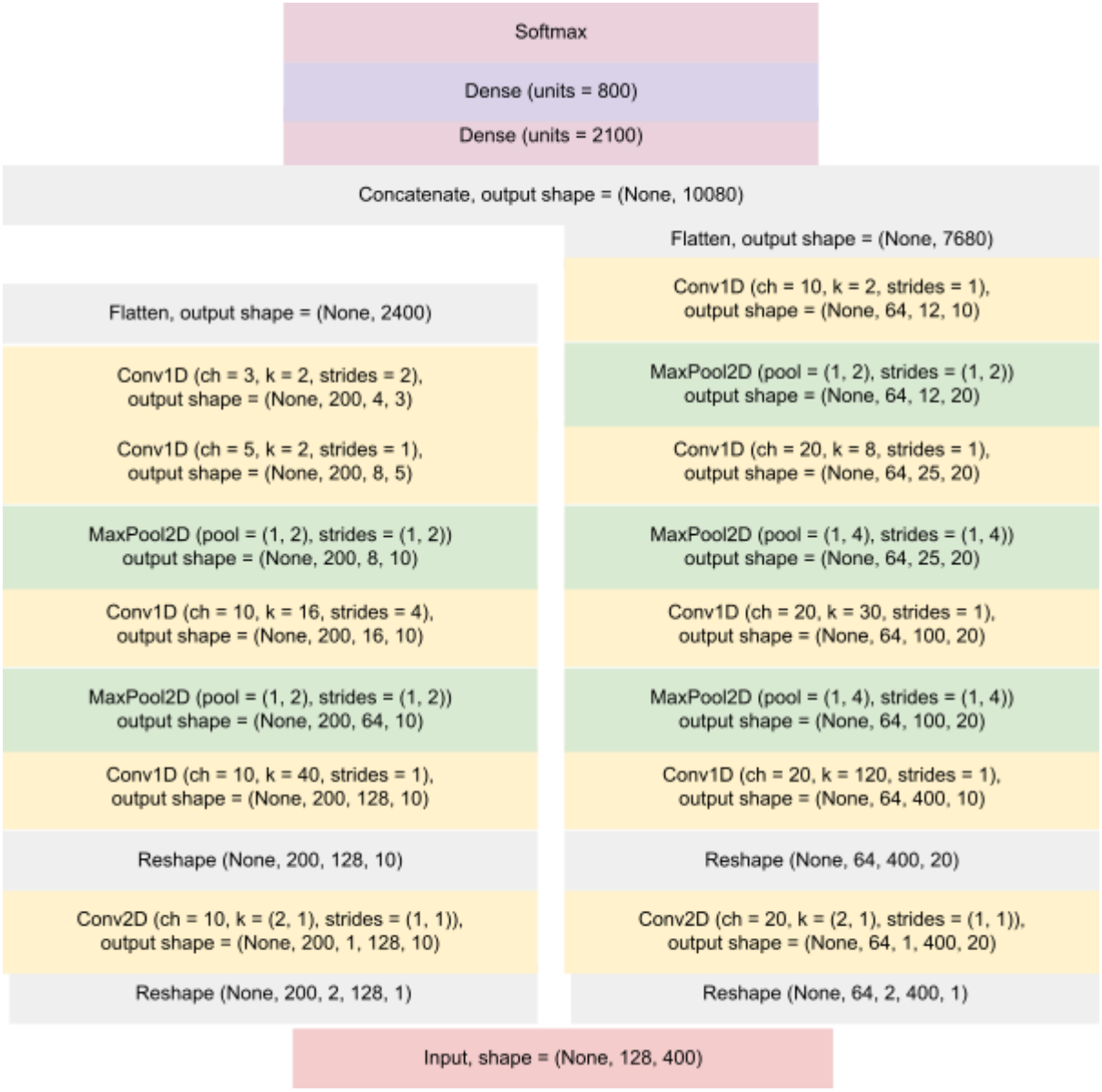

Fig.5 Architecture of the "coordinate" net for 264 birds

Total number of trainable parameters is ~23 mln. BatchNormalization and Dropout layers are omitted (see SoundClassifier v6.4.4 264b.ipynb on GitHub for the full model)

CN presented on Fig.5 achieved validation accuracy of 95% after 18 epoches. The training curve is depicted in Fig.6.

Conclusions

The approach presented here tries to realize the mathematical definition of convolution. So we have the right to expect the insensitivity of CoordinateNet to timeshift. More generally, since we are dealing with a trainable model, we can expect to improve the accuracy of the model noticeably using time shift based augmentation.
Table 1 shows significant accuracy growth with increasing the augmentation

Even the presented nonoptimal solution has two evident advantages over [1] and [2]. The first one is greater accuracy and the second is the understandable scaling algorithm. In my

opinion if we look at convergence rate (see Fig.6) CN for 264 birds does not require significantly more channels/complexity to show 0.89 accuracy on 900+ birds.

For goals of species diversity monitoring it seems a good idea to augment the amount of training samples combining marked up records of different birds. What birds to combine depends on local conditions.

Author hopes this approach can be applicable to wide spectrum of audio classification/recognition problems

This project has been done (including training) on a notebook: CPU - AMD Ryzen 5 4600U, 16.0 GB RAM. So time estimations here are quite large. Version of TensorFlow is 2.9.1

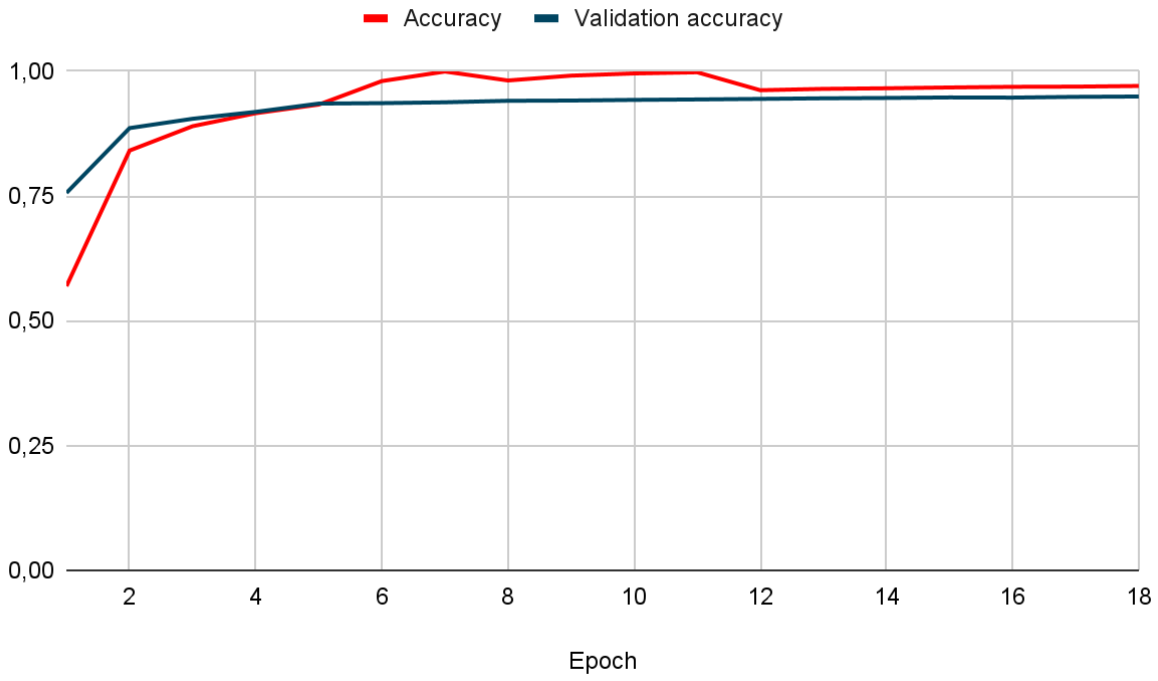

Fig.6 Training curve of CN for 264 birds, landscape #3

References

[1] An Efficient Model for a Vast Number of Bird Species Identification Based on Acoustic Features: Hanlin Wang, Yingfan Xu, Yan Yu, Yucheng Lin, Jianghong Ran. Animals 2022, 12, https://doi.org/10.3390/ani12182434
[2] Stefan Kahl, Connor M. Wood, Maximilian Eibl, Holger Klinck: BirdNET: A deep learning solution for avian diversity monitoring. Ecological Informatics 61 (2021). https://doi.org/10.1016/j.ecoinf.2021.101236
[3] Changlu Guo, Márton Szemenyei, Yangtao Hu, Wenle Wang, Wei Zhou, Yugen Yi: CHANNEL ATTENTION RESIDUAL U-NET FOR RETINAL VESSEL SEGMENTATION. 2019, https://github.com/clguo/CAR-UNet
[4] Mingxing Tan, Quoc V. Le: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Appendix A

Scripts of this work are presented in Tables A1. Notation <AAA> means a parameter from _config.html or local one declared in a script.

Script	**Description**
cutting long files v1.0.ipynb	Initial audios longer than 1 minute is cutted into 1 minute segments. Original long files are saved into <DIR_LONG> directory. By cutting long files into segments, we get more markup accuracy because we calculate the maximum values for more local intervals.
preparing_data_v1.1.py	This frame script calls scripts of step-by-step data preparation (see descriptions below). Parameters are passed through the _config.html file which is created here as well. This code does the following: 1. builds _config.html 2. calls bands_v1.0.py 3. calls files_v1.0.py 4. creates <FILES_2MRK>.html and <FILES_2LBL>.html <FILES_2MRK>.html is a list of recordings to mark up <FILES_2LBL>.html is a list recordings to cut without marking up 5. calls mark_up_mels_v1.0.py 6. calls sampling_v1.0.py 7. calls sampling_2lbl_v1.0.py 8. concatenates results of sampling_v1.0.py and sampling_2lbl_v1.0.py and splits the union into training and validation sets 9. calculates how many times to augment amount of initial samples of a bird to achieve the minimum required 10. calls ● cutting_files_v1.1.py/ ● cutting_files_v1.0.py/ ● cutting_files_v2.0.py to create samples of landscapes #1-3 respectively 11. calls mels_v2.0.py to create mel spectrograms
bands_v1.0.py	Builds reference table <BIRDS2BANDS>.html. This table

	determines for each bird the range of mels in which the bird's song is most pronounced.
files_v1.0.py	Merges list of files of <DIR_DATA> directory with train_metadata.csv and concatenates with files of <DIR_RARE> folder. Saves the result to <FILES>.html. Particularly noisy records are filtered out at this stage. <DIR_DATA> is directory where BirdCLEF2023 data is unpacked and Initial audios longer than 1 minute is cutted into 1 minute segments <DIR_RARE> stores new/modified additional recordings for rare birds.
mark_up_mels_v1.0.py	Marks up recordings from <FILES_2MRK>.html. Results saves to <NICKS>.html
sampling_v1.0.py	Registers initial (before augmentation) 5s samples in <SAMS1>.html according to <NICKS>.html
sampling_2lbl_v1.0.py	Registers initial (before augmentation) 5s samples from recording with secondary label.
cutting_files_v1.1.py/ cutting_files_v1.0.py/ cutting_files_v2.0.py	Builds samples of landscapes #1-3 respectively.
mels_v2.0.py	Creates mel spectrograms
_afpfly1.ipynb	Corrects XC156644.ogg. Deletes human speech starting from the 47th second
_afpkin1.ipynb, _brcwea1.ipynb, _brtcha1.ipynb, _crefra2.ipynb, _dotbar1.ipynb, _golher1.ipynb, _lotcor1.ipynb, _lotlap1.ipynb, _rehblu1.ipynb, _whctur2.ipynb, _whhsaw1.ipynb, _yebsto1.ipynb	These scripts make a desperate effort to increase the number of samples for eponymous birds, with not enough records to split into training and validation sets. New/modified audio recordings are laid into folder <DIR_RARE> where handled by files_v1.0.py _crefra2.ipynb, _lotcor1.ipynb, _yebsto1.ipynb handle new audio recordings from folder <DIR_NEW>. Archive new_data.zip contains this folder

Table A1. Scripts of this work

Appendix B, Augmentation algorithms

An audio file with secondary labels is splitted into an initial set of 5s intervals (samples) as is. Two more sets of 5s samples are produced from the audio with arbitrary time shifts of 0s - 2.5s.

Formation of samples from marked recordings depends on a landscape (See Table 1). For the landscape №1 the algorithm starts with calculation of the necessary amount of samples. This amount is defined by the formula:

$$I = ((M - 3 * S_{2lbl} - S_{1lbl})\%S_{1lbl} + 1)\%3 + 1 \quad (1)$$

, where

M - minimum number of bird samples,

S_{1lbl} - amount of samples from marked up recordings with primary label only

S_{2lbl} - augmented amount of samples from recordings with secondary labels

For each sample selected from marked up audio files the algorithm determines the most left/right micks: x_1 and x_2 respectively, calculate the middle point $(x_1 + x_2)/2$ and form iteratively 5s samples where the calculated point $(x_1 + x_2)/2$ is located in an arbitrary diapason of 1/2 ± 3/8 of the new sample. Equation (1) determines the number of iterations. 3 samples are produced on each iteration. Even if the number of initial samples for a bird is more than minimum required the algorithm produces 2 additional samples for an initial one. The upper limit of 2500 samples is applied after the augmentation.

For the landscape №2 we calculate the necessary amount of additional samples as well. This amount is defined by the formula:

$$I = (M - 3 * S_{2lbl} - S_{1lbl})\%S_{1lbl} + 1 \quad (2)$$

, where

M - minimum number of bird samples,

S_{1lbl} - amount of samples from marked up recordings with primary label only

S_{2lbl} - augmented amount of samples from recordings with secondary labels

Further algorithm is identical to the previous one except one just 1 sample is produced on each iteration from the Equation (2).

In the third landscape the logic is similar to the first one but the algorithm of sample selection differs. We determine the most left/right micks and do 3 additional 5s samples with time offset:

1. 5s sample where the most left nick of the sample interval is located in an arbitrary point between 1/4 ± 1/8 of the new sample.
2. 5s sample where the most right nick of the initial sample is located in an arbitrary point between 3/4 ± 1/8 of the new sample.
3. 5s sample where the middle point of the most right/left nicks is located in an arbitrary point between 1/2 ± 1/8 of the new sample.

This algorithm of sample selection was realized first and was simplified for the landscape №1.