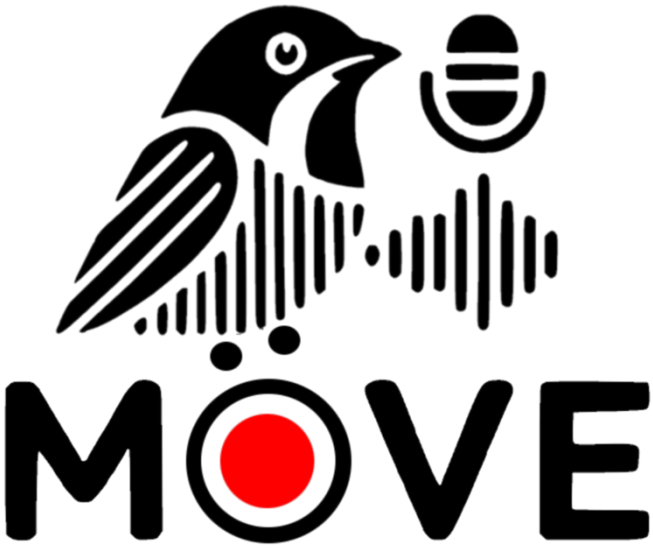


Moove (Marking Online using Only the Onsets of Vocal Elements) is a novel tool for real-time syllable segmentation and classification of birdsong, designed to enable closed-loop experiments in vocal learning research. Designed to study the learned vocalisations of Bengalese finches, Moove identifies target syllables in a bird's song and provides feedback in real time. Moove provides an out-of-the-box, neural network-based approach to reliably target vocal syllables before their end, enabling a reinforcement protocol where a specific syllable can be targeted with aversive white noise or an alternative feedback stimulus if adjusted.

Moove uses a two-stage architecture: a convolutional-based encoder that segments syllables in the audio signal and a CNN classifier that assigns each detected syllable segment a label, identifying its type based on the initial part of its structure. This design allows Moove to operate at a lower audio chunk duration than other tools, enabling faster and more accurate syllable recognition with minimal latency. Moove includes a GUI for creating training datasets using unsupervised methods and training the networks, as well as a recording script for real-time syllable targeting.

Authors: Franziska Heubach, Jacqueline Göbl

Published:

The following pages will provide you with guided explanations for using MooveTaf and MooveGUI. Following the instructions step-by-step should give you a nice introduction into all possible ways to use Moove.

If you find yourself having any questions unanswered or you want to share suggestions, feel free to check out our Github: https://github.com/veitlab/moove

or contact us directly.

Table of Contents

1. Installation

1.1. Windows

This chapter describes the installation process we recommend in detail. Following these instructions should make you ready and set to jump right into recording your data.
Moove requires:

- `Python versions 3.9 to 3.12,`
- `numpy version < 2.0 and`
- `torch version < 2.6.`

Before installation you should check your running python version in the Windows PowerShell with:

```
python --version
```

Moove can be easily installed using the Windows PowerShell and pip by entering the following line in a newly opened Windows PowerShell window.

```
pip install moove
```

or for a specific version:

```
pip install moove==1.0.0
```

Moove will be installed on the same disk and folder path your python is installed on.
If you don't know where your python is installed, follow these instructions.
Usually, your python is installed in **AppData > Roaming** or **AppData > Local**. You can access your AppData folder by typing *%appdata%* into Windows search or by pressing *Windows+R* and typing *appdata* in the opened window (Fig 1).

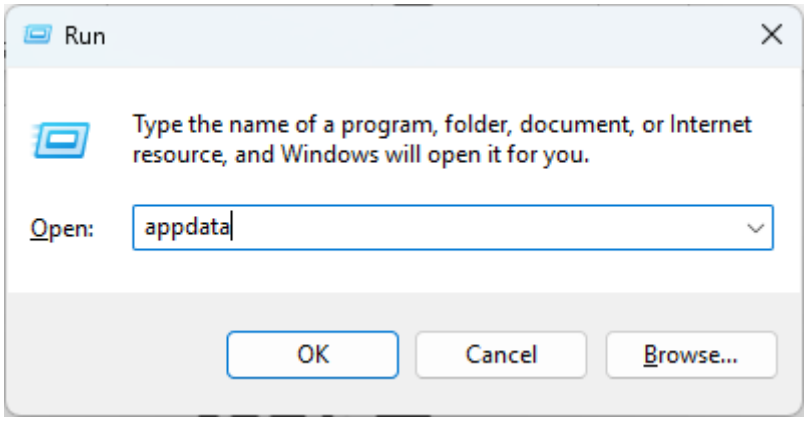


Figure 1: Finding appdata folder

Within your AppData folder, you can now search for your Python installation in the folders **Local** or **Roaming** (Fig. 2).

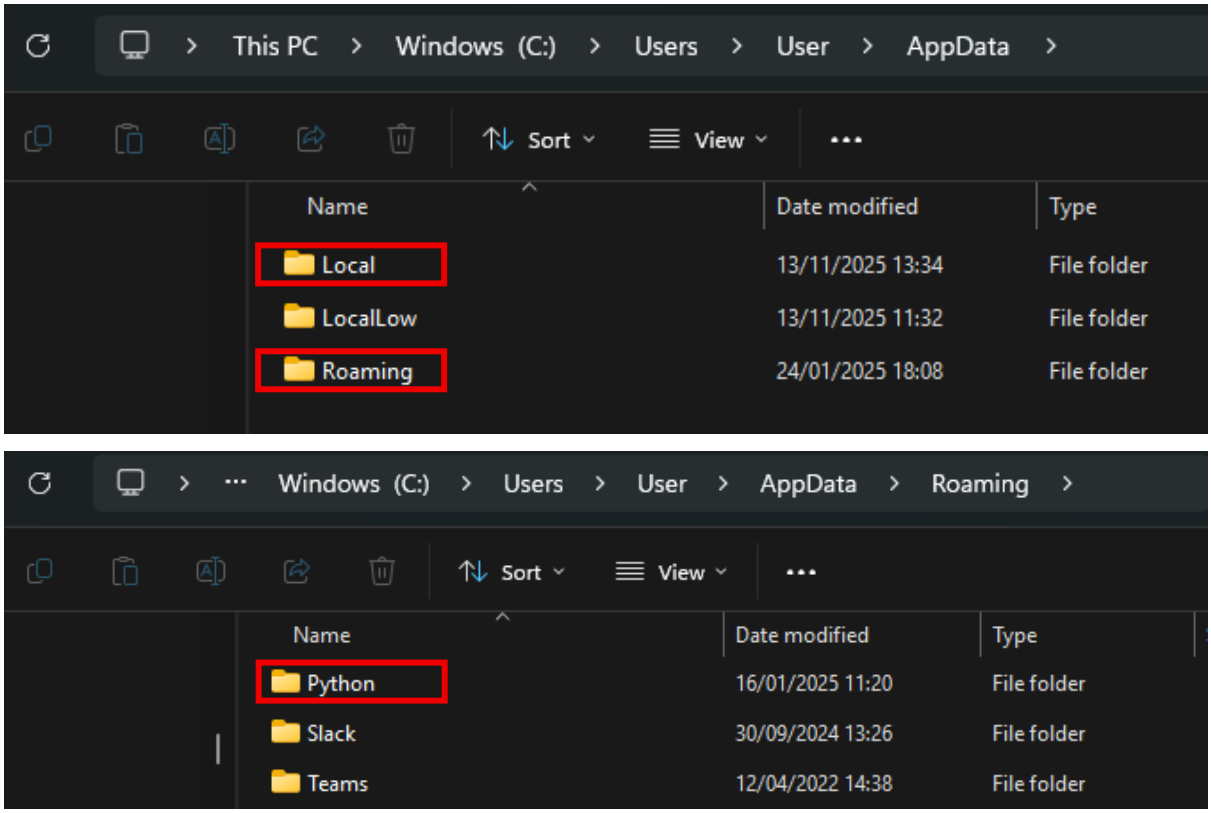


Figure 2: Find your python folder

Once you found your python folder, go into **Python > Python312** (or whatever version you have installed) **> site_packages**. This is where your **moove** folder is located, containing the codes for the **moovegui**, **moovetaf** and all additional utility codes. Furthermore, the folder **example_data** includes an example file from a previously recorded bird, that will be the first file opening in the MooveGUI (Fig. 3). You usually don't need to access these folders at all, unless you want to confirm that everything is installed correctly or perhaps modify the code.

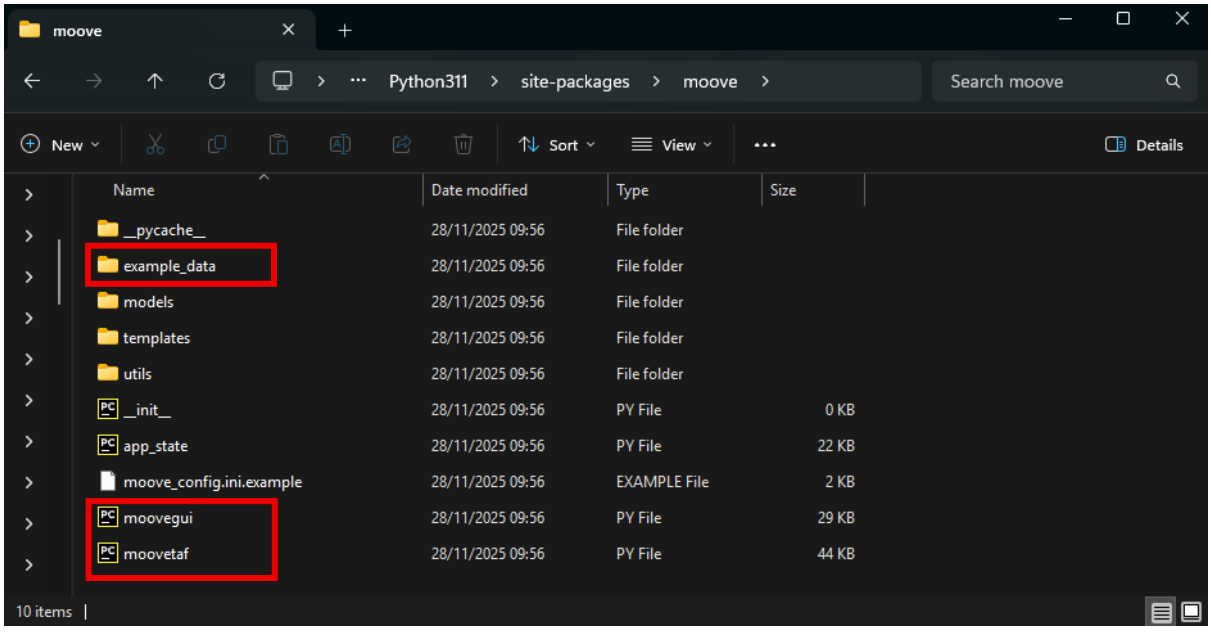


Figure 3: Moove folder containing python code files

In **Python > Python312 > Scripts** you can find the **moovegui.exe** and **moovetaf.exe** (Fig. 4). This is the folder from where your program will start. You can also start the application from this folder directly, but using Windows PowerShell is easier.

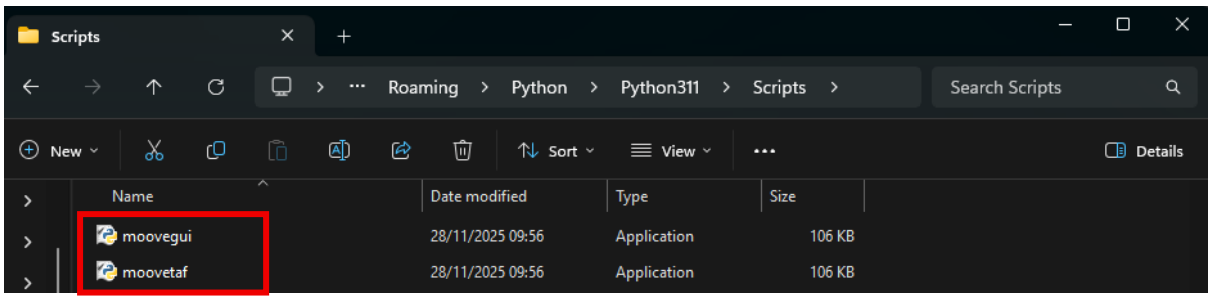


Figure 4: Moove applications

1.1.1. Portaudio

Moove uses the **sounddevice** (Geier et al., 2020) library, which depends on **PortAudio** (Bencina & Burk, 2001). On most systems, PortAudio is already available or bundled. If needed, e.g. if starting the program gives you an error saying sounddevice or portaudio missing, install **sounddevice using pip** in the Windows PowerShell.

```
pip install sounddevice
```

1.1.2. Enabling ASIO support

As Moove's real-time targeting capabilities highly depend on low latency, we recommend using the **ASIO audio driver**. In Windows, ASIO support must be enabled separately in sounddevice. To do so, follow the instructions below.

1.1.2.1 Portaudio includes ASIO file

The newest version of the sounddevice package already contains a file enabling ASIO support on windows. However, this needs to be made available for Moove first.
To do so, locate the folder where your sounddevice was installed. For this, again move into your **AppData** folder and into python **site-packages** (see above). Common paths for the sounddevice installation are

C:\Users\<YourUsername>\AppData\Roaming\Python\<YourPythonVersion>\site-packages_**sounddevice_data\portaudio-binaries**\

or

C:\Users\<YourUsername>\AppData\Local\Programs\Python\<YourPythonVersion>\Lib\site-packages_**sounddevice_data\portaudio-binaries**\

In this folder, you should find two .dll files, one named *libportaudio64bit.dll* and one named *libportaudio64bit-ASIO.dll*. If your folder does **not** contain the ASIO file, follow the instructions in 1.1.2.2.

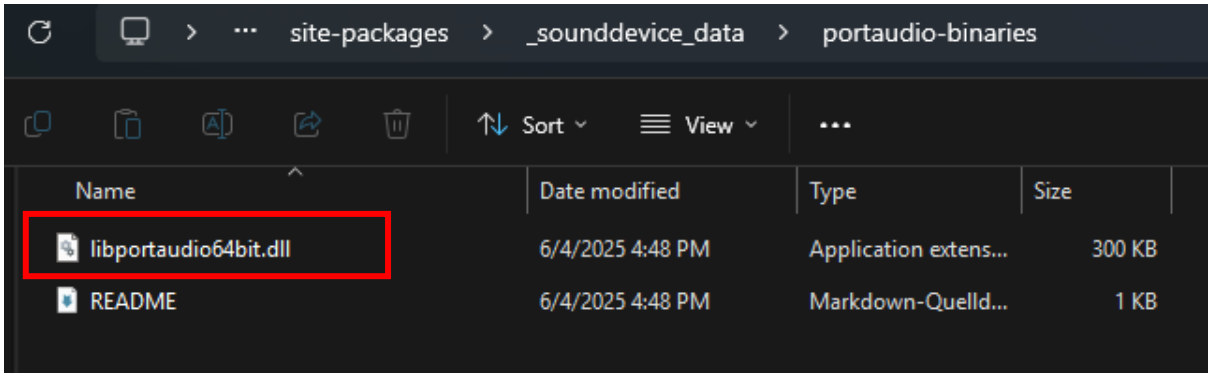


Figure 5: Replacing the portaudio file

If you really want to be sure you're in the correct folder/ file, start moovetaf.exe and try to delete the .dll file: it will tell you you can't because it is open in python – that's when you know you're correct.

Now, delete the *libportaudio64bit.dll* and **delete the -asio** ending from the second file in the folder. Once the **driver is installed, the PC restarted and the file replaced**, you should now be able to find the correct inputs/ outputs including **ASIO** in their name in the device list opened when starting MooveTaf. In our recording setup, this is the Yamaha Steinberg inputs/ outputs (see *Recommended Hardware*).

```
Yamaha Steinberg USB ASIO
```

1.1.2.2 Portaudio doesn't include ASIO file

Download a new portaudio file with ASIO drivers enabled for Windows, e.g. from Github (Geier et al.).

https://github.com/spatialaudio/portaudio-binaries

Depending on your Windows, download the 32- or 64-bit version. If you are unsure, check in your system, but usually the highlighted file is the correct version (Fig. 6). Once installed, again follow the steps in 1.1.2.1 to head to your sounddevice folder, and replace the existing *libportaudio64bit.dll* file with the new downloaded one, renaming it from *libportaudio64bit-ASIO.dll* to *libportaudio64bit.dll*.

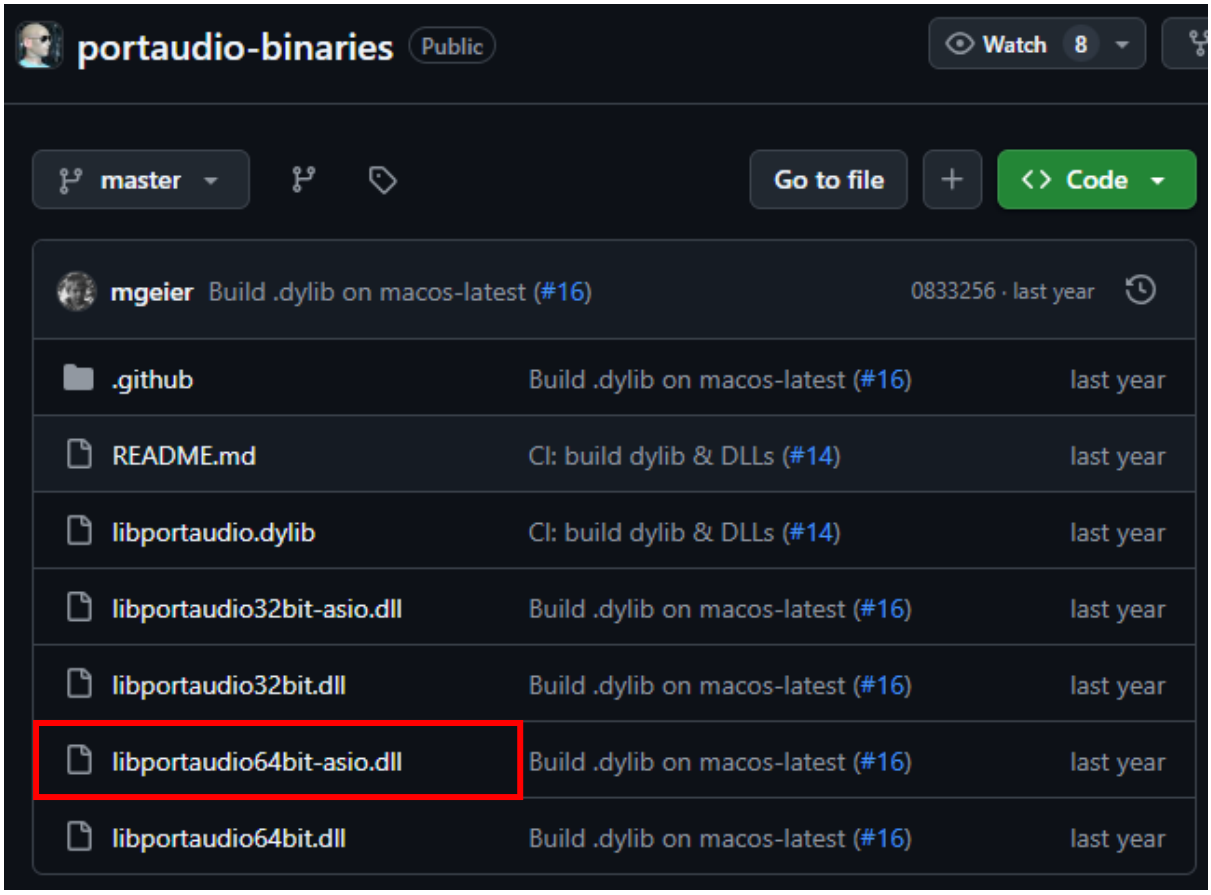


Figure 6: Downloading the ASIO enabled portaudio file

1.2. MacOS

Use homebrew to install portaudio before installing Moove.

```
brew install portaudio
```

Note: I think your python has to be installed via homebrew as well

1.3. Linux

For Linux/ Ubuntu systems we recommend installing Moove in an environment.

In the terminal, first create a new python environment:

```
python -m venv venv
source venv/bin/activate
```

Then install all required packages:

```
sudo apt install python-dev-is-python3 gcc
sudo apt update && sudo apt install portaudio19-dev
sudo apt-get install python3-tk
```

Then install moove in the environment:

```
pip install moove
```

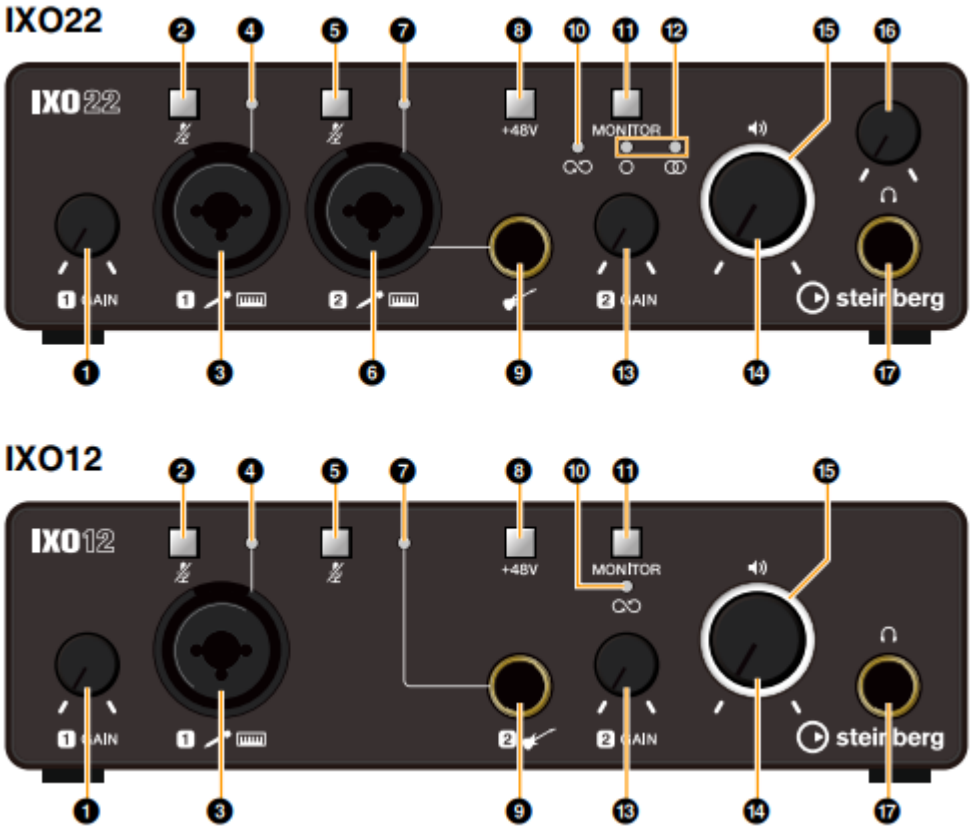
1.4. (Optional) Recommended hardware

In our lab, we use the **Yamaha Steinberg IXO12/IXO22 audio interface** for recordings. If you are using a different setup, you can **ignore** the following pages explaining the setup with this specific interface.
This manual is mostly taken and modified from the Steinberg Website, so if you need more information check that out.
https://www.steinberg.net/audio-interfaces/ixo12/

1.4.1. Yamaha Steinberg IXO12 / 22 manual

IXO 12: single microphone input

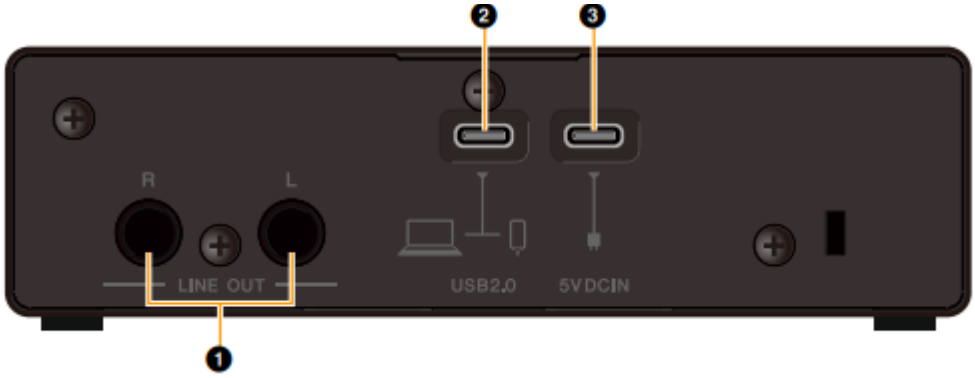
IXO 22: double microphone input



1. Input 1 gain adjustment
2. Mute input 1 (red light means muted)
3. Input 1 for microphone
4. Input 1 signal/ peak display
   Input gain should be adjusted such that with normal sound level it lights up green, and with peak sound level it should be blinking red shortly
   i. Red: -3 dBFS (decibel relative to full scale; 0 is max, higher leads to überwachungclipping) or higher
   ii. Green: -20 dBFP to less than -3 dBFS
   iii. No light: less than -20 dBFP
5. Mute input 2 (second mic for IXO22; instrument input at IXO12)
6. Input 2 for microphone (only IXO22)
7. Input 2 signal/ peak display

8. +48V switch (phantom power):
    a. Turn on if you use a condenser microphone (without external power; e.g. RODE)
    b. DON'T turn on if you use a dynamic microphone that already has a power supply (e.g. small microphones with power supply box)
9. Additional adapter for jack plug (e.g. digital instrument)
    a. Info: if you connect a digital instrument to IXO22 the second microphone input is deleted
11. Monitor switch, switches between loopback and direct monitoring
    a. Loopback function display (mixes input signals with software generated audio signals from the computer, sends them back to computer) (10)
    b. Direct monitoring display (12)
        i. Mono motoring : input 1 and 2 are emitted at LINE OUT or PHONES port
        ii. Stereo monitoring : displayed if input 1 is L and input 2 is R (both inputs for one signal; to treat them as single inputs use mono)

    The loopback and monitoring functions should always be turned off, as it could lead to either sound playback (e.g. the bird song getting played back instantly) or the output channel being displayed on the input channel.
13. Input 2 gain adjustment (mic 2 or instrument for IXO22, instrument for IXO12)
14. Output level adjustment for LINE OUT L/R (for IXO12 also adjusts PHONES)
15. Power display (blinks permanent if power supply is insufficient)
16. PHONES adjustment for headphones (only IXO22)
17. PHONES plug for stereo headphones



1. LINE OUT L/R connection for external speakers (jack plug)
    a. For level adjustment use OUTPUT at the front
2. USB 2.0 port to connect to computer
3. 5V DC IN port to connect to an external power source

    Only necessary if you connect to a device that cannot supply sufficient power (e.g. iPad)

    Needs 5V DC with >=500mA

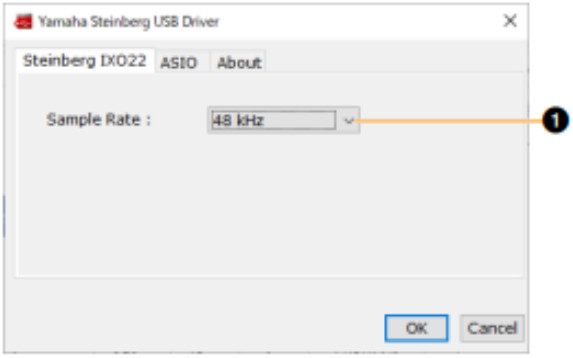
1.4.2. Yamaha Steinberg USB Driver

To use the audio device with MooveTaf, download the respective driver from the Steinberg website. This is only available for MacOS and Windows.

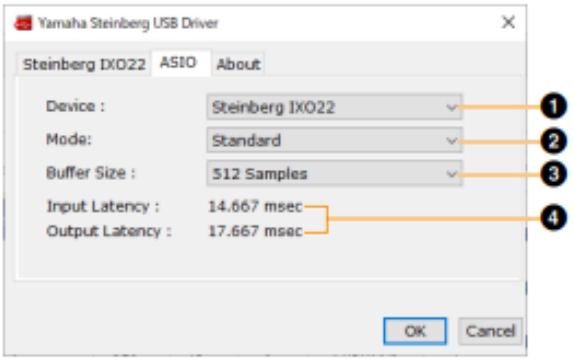
https://o.steinberg.net/de/support/downloads_hardware/yamaha_steinberg_usb_driver.html

Then restart your computer to enable the driver.

1.4.3. Setting options in the driver



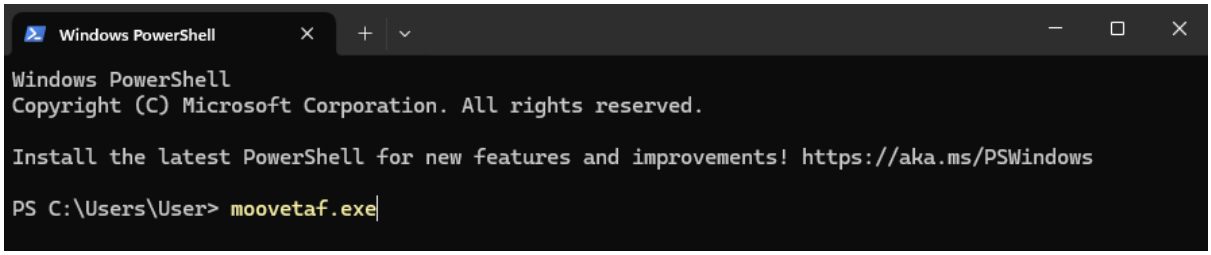
1. Can set the sample rate for your recordings (44.1 – 192 kHz)



1. Choose your device (only necessary if more than one connected)
2. Choose latency mode
    a. Low latency (needs high computing power on your device)
    b. Standard latency
    c. Stable latency (high latency, focused on a stable signal rather than a quick one; for devices with lower computing power)
3. Buffer size
    a. varies depending on your sampling frequency
    b. latency depends on buffer size
    lower buffer size = lower audio latency
4. Input/ Output latency of the signal
    a. latency depends on buffer size
    lower buffer size = lower audio latency

2. MooveTaf

You can start MooveTaf directly from the terminal/ Windows PowerShell by typing **`moovetaf.exe`**.



While recording, we strongly recommend assigning **high priority** to the running python program executing MooveTaf, as this can improve recording quality as well as classification accuracy and speed. To do so on Windows, open the Task Manager, in the tab *Details* search for python, right-click on your running task and *Set Priority* to *Real-Time*.

Once started, the program will show you a list of all your connected **input and output devices** in a numerical order (Fig. 8). Behind each device, it is listed how many inputs (in) and outputs (out) are available. The program will ask you to choose your desired input device (microphone) and output device (speaker). They can be set by typing in the respective **device number** and pressing **Enter**. If one device offers input and output, the same number can be set for both.
If you can't find your desired devices in the list, check out chapter **1. Installation** for a possible solution.

```
PS C:\Users\User> moovetaf
   0 Microsoft Sound Mapper - Output, MME (0 in, 2 out)
<  1 PHL 241B8Q -2 (HD Audio Driver , MME (0 in, 2 out)
   2 PHL 241B8Q (3- HD Audio Driver , MME (0 in, 2 out)
   3 Speakers (Realtek(R) Audio), MME (0 in, 2 out)
   4 Primary Sound Driver, Windows DirectSound (0 in, 2 out)
   5 PHL 241B8Q -2 (HD Audio Driver for Display Audio), Windows DirectSound (0 in, 2 out)
   6 PHL 241B8Q (3- HD Audio Driver for Display Audio), Windows DirectSound (0 in, 2 out)
   7 Speakers (Realtek(R) Audio), Windows DirectSound (0 in, 2 out)
   8 PHL 241B8Q (3- HD Audio Driver for Display Audio), Windows WASAPI (0 in, 2 out)
   9 PHL 241B8Q -2 (HD Audio Driver for Display Audio), Windows WASAPI (0 in, 2 out)
  10 Speakers (Realtek(R) Audio), Windows WASAPI (0 in, 2 out)
  11 Microphone (Realtek HD Audio Mic input), Windows WDM-KS (2 in, 0 out)
  12 Stereo Mix (Realtek HD Audio Stereo input), Windows WDM-KS (2 in, 0 out)
  13 Speakers (Realtek HD Audio output), Windows WDM-KS (0 in, 2 out)
  14 Headphones (Realtek HD Audio 2nd output), Windows WDM-KS (0 in, 2 out)
  15 PHL 241B8Q (ACX HD Audio Speaker), Windows WDM-KS (0 in, 2 out)
  16 Headset (@System32\drivers\bthhfenum.sys,#2;%1 Hands-Free%0
;(Bose NC 700 HP)), Windows WDM-KS (0 in, 1 out)
  17 Headset (@System32\drivers\bthhfenum.sys,#2;%1 Hands-Free%0
;(Bose NC 700 HP)), Windows WDM-KS (1 in, 0 out)
  18 PHL 241B8Q -2 (ACX HD Audio Speaker), Windows WDM-KS (0 in, 2 out)
  19 Headphones (), Windows WDM-KS (0 in, 2 out)
Select the desired input device (Microphone):
```

Figure 7: Starting MooveTaf

Once you selected your input and output device, the program will start recording automatically.

2.1. Baseline recordings

When recording a bird for the first time, you need to perform some baseline recordings before training a classification network. With the first start of MooveTaf, the folder **".moove"** is

created. It contains all your recorded data, trained models and config settings (Fig. 9). Note that, on Linux and MacOS (and sometimes Windows) the ".moove" folder is hidden. On MacOS, it can be made visible pressing `< CMD + shift + . >` while in your user folder, on Linux it can be done using `< Ctrl + H >`. If you still can't find it, search for "how to unhide folders".

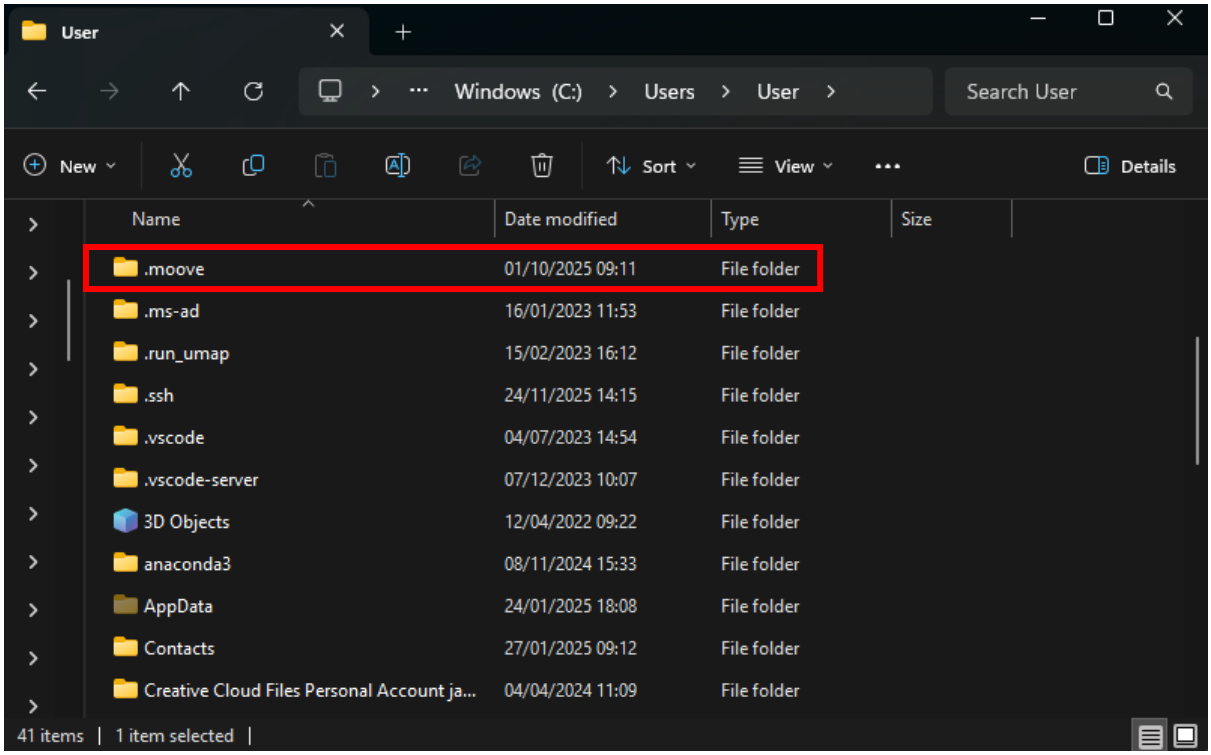


Figure 8: Location of your ".moove" folder

By default, it is saved within your **username** folder under **C:/Users/<YourUsername>/**. Once created, you can move the folder to your desired location and set this location in the terminal before starting the Moove applications. For that, use the following commands depending on your system (Fig. 10), replacing the directory with your actual path. Furthermore, you have to change the *global_dir* variable in the **config file** (Fig 13, pink box) to your actual path, e.g. `C://<PathToFolder>/` or `D://<PathToFolder>/`.

Figure 9: Setting the ".moove" folder location

The folder structure for recorded data in *rec_data* is as follows and is parsed into the GUI in this way (Fig. 11). In case one folder doesn't exist and the structure changes, your data cannot be

found correctly. The names of your parent bird folder and experiment folder are set in the config (see 2.1.1).

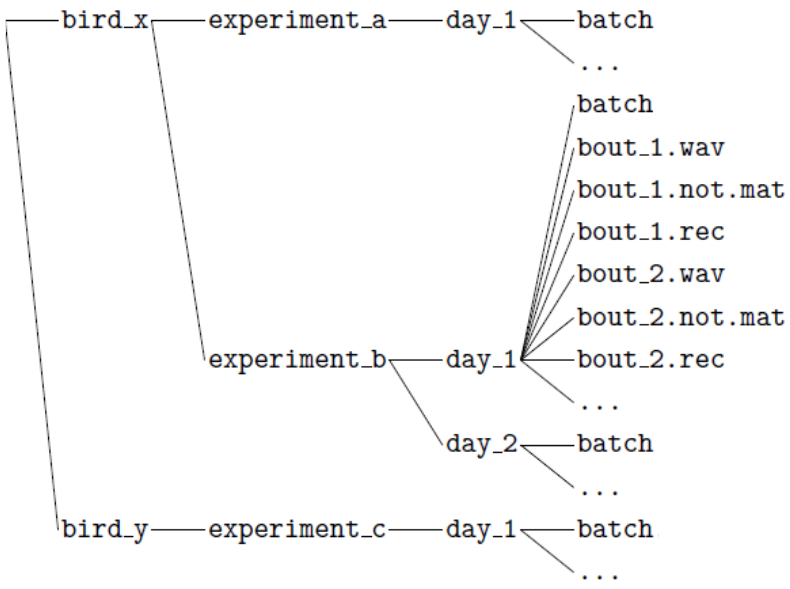


Figure 10: Folder structure in rec_data

A **batch.txt** file will be automatically created once the recording starts. It contains a list of all file names included in this day folder (Fig. 12). You can also create and modify these files using any text editor.

Figure 11: Structure of a batch file

While recording with MooveTaf, we don't recommend opening the MooveGUI. Due to parallelly accessing the batch.txt file, errors while writing the batch.txt file in MooveTaf can occur. If errors happen, you can update and rewrite the batch.txt file in the GUI (see 3.2.) or manually correct the batch.txt file.

2.1.1. Setting the config

The default version of the config file can be found in the moove folder in your AppData (see 1. Installation) (Fig. 13). However, changes in the config should always be made in the **config file** in the **.moove** folder. It is an *.ini* file that can be opened and edited with any text editor program. Remember to save your changes once you're done and before starting Moove.

The line **global_dir** lets you set your folder saving destination (pink box).

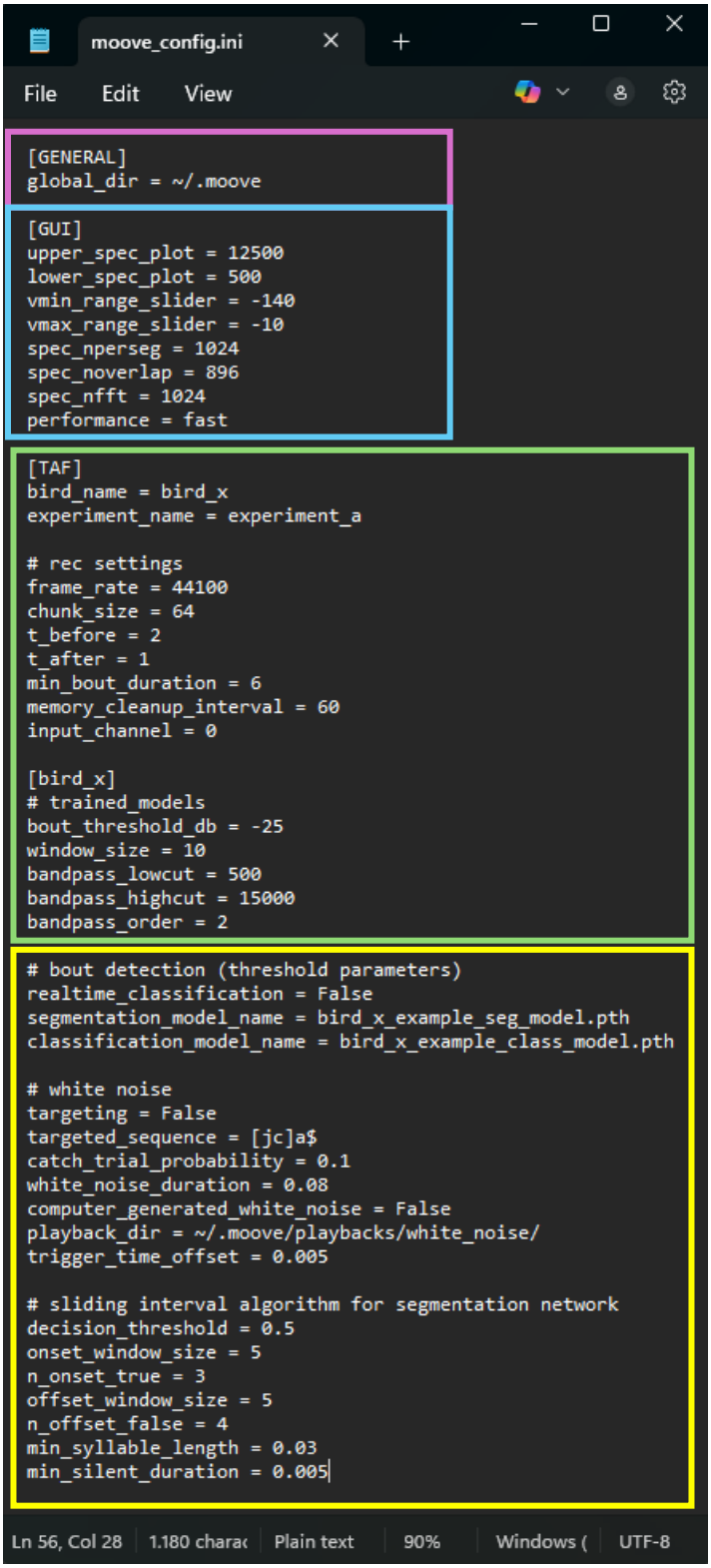


Figure 12: Config settings for baseline recordings

In the section below `[GUI]`, settings for opening the MooveGUI can be edited (blue box, see 3.1 Setting the config). In the green highlighted box below `[TAF]`, settings for using MooveTaf can be found. The parameters relevant for a first-time setup and baseline recordings are described in the table below (Table 1). For the recording, the `bird_name` and `expreriment_name` have to be individually adjusted. **Don't use spaces in the bird_name and experiment_name parameter.** The section title `bird_x` has to be renamed to be the same as the `bird_name`. For different birds, the whole section `[bird_x]` can be copied and added with the different bird names as titles. All the following parameters will be described in a later chapter (2.2. Targeting).

Table 1: Parameters in the config for MooveTaf

Parameter	Default Value	Description
bird_name	ye00pu07	Specifies the name of the bird
experiment_name	baseline	Name of the experiment
frame_rate	44100 [Hz]	Sampling rate for audio recording
chunk_size	64	Processed samples per audio chunk during digitization; smaller values for lower latency
t_before	2 [seconds]	Time before the bout trigger to include in recording
t_after	1 [seconds]	Time after the bout trigger to include in recording
min_bout_duration	4 [seconds]	Minimum duration of a bout to be saved
memory_cleanup_interval	60 [seconds]	Interval for memory cleanup to save resources
input_channel	0	Channel where the input comes in, can be one channel or two channel (0,1)
bout_threshold_dB	-30	Threshold in dB for detecting a bout; try around during the first time setup, **very setup- and bird-specific parameter**
window_size	10	Smoothing window size for the smoothing filter for the amplitude signal
bandpass_lowcut	500	Defines the lower cutoff frequency for filtering the spectrogram.
bandpass_highcut	15000	Defines the upper cutoff frequency for filtering the spectrogram.

bandpass_order	2	Defines steepness of the filter's cutoff; the higher the sharper the transition

In the yellow highlighted part, the `realtime classification` can be enabled and set up. However, for baseline recordings no model will be trained yet. Therefore, **`realtime_classification`** is by default set to false. Trained segmentation and classification models can later be added in the lines below, by default example networks are given. For setting up these parameters check out 2.2 Targeting below. You can **quit** MooveTaf by pressing Ctrl + C in the terminal window. If you trained your models on a different chunk_size then the default 64 you have to change the `chunk_size` in the config file before recording with online classification. This is not recommended since chunk_size influences classification performance and online recording performance.

2.2. Targeting

Once you trained a model on your song data for following recording sessions, you can use the trained networks to segment and classify online while recording.

Additionally, you can target specific syllable sequences to get playback. Targeting while online classification has to be turned on separately: **`targeting = True`**.

The time for processing and classifying a syllable can be adjusted by changing the chunk_size. With the default chunk_size of 64, online syllable classification will take approximately 30ms. Changing the chunk_size can lower this value but also make classification more prone to errors. If your syllables are very similar, increasing chunk_size might increase the classification accuracy, but will decrease targeting latency.

You can target different sequences and playback either computer generated white noise or different sound files (.wav), that are stored in the folder given in the section: **`playback_dir`**.

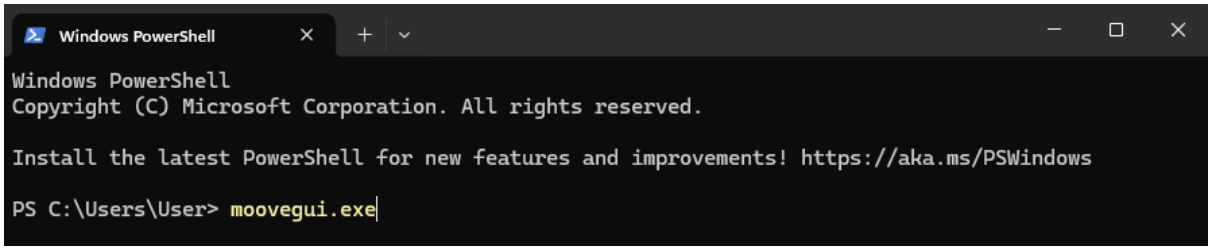
The path in the `playback_dir` points to the folder where the sound files are. They have to be .wav files. Since it will take all .wav files into consideration as a playback option, make sure that only files you want to playback are in the folder you have given. If multiple .wav files are in the same folder, the sounds will be chosen randomly for each target found.

While targeting with different sequences (**`targeted_sequence`** `= hfd$, stl$`), the script chooses one of the given sequences before each bout and plays the given sounds at random on the last syllable of the sequence (this is defined by the $ at the end of each sequence, since it follows the rules of regular expressions). You can choose any regular expression that narrows down your targeting to the exact sequence you want to target. If you are not familiar with regular expressions in python, try asking ChatGPT or Google.

Parameter	Default Value	Description
realtime_classification	False	
segmentation_model_name	bird_x_example_seg_model.pth	Name of the segmentation model
classification_model_name	bird_x_example_class_model.pth	Name of the classification model
targeting	False	Only when targeting is wanted, for realtime online targeting of the following targeted_sequence
targeted_sequence	[jc]a$	Target sequence(s) are targeted as regular expressions of the already labeled sequence. When multiple sequences are given, a target sequence is chosen randomly at the beginning of the bout.
catch_trial_probability	0.1	How many bouts are not targeted, Value between 0 and 1
white_noise_duration	0.08 [seconds]	Duration of the computer-generated white noise.
computer_generated_white_noise	False	Playback of computer-generated white noise
playback_dir	~/.moove/playbacks/white_noise/	Folder where playback files are. Playback files are taken when computer_generated_white_noise is false. Folder can contain several files, which are chosen randomly in the moment of playback
trigger_time_offset	0.005 [seconds]	Time after playback when a new trigger cannot be triggered
decision_threshold	0.5 [%]	Defines a probability threshold for detecting a syllable segment.
onset_window_size	5[chunks]	Specifies the size of the sliding window used to detect onsets.
n_onset_true	3 [chunks]	Sets the number of *True* detections within the sliding window.
offset_window_size	5 [chunks]	Specifies the size of the sliding window used to detect offsets.
n_offset_false	4 [chunks]	Sets the number of *False* detections within the sliding window.
min_syllable_length	0.03 [seconds]	Specifies the minimum duration of a syllable that a syllable segment must have.
min_silent_duration	0.005 [seconds]	Specifies the minimum silence between two syllables to be counted as separate units.

3. MooveGUI

The MooveGUI will let you work with your data, train networks on your segments and thereby classify the syllables in your songs. The MooveGUI can be started via Windows PowerShell using the command **`moovegui.exe`**.



3.1. Setting the config

Once you recorded data using MooveTaf, the folder **.moove** will be created, which contains your recorded data and trained models (see also 2.1. Baseline recordings). This folder also contains your moove_config.ini file, which can be opened and edited using any text editor program. In the section `[GUI]`, highlighted in blue here, you can set parameters for the MooveGUI (Fig 14). It's recommended to first start and look at the GUI for once and then come back to setting up the parameters as desired. The main window of the GUI is described in more detail in 3.2. Main window.

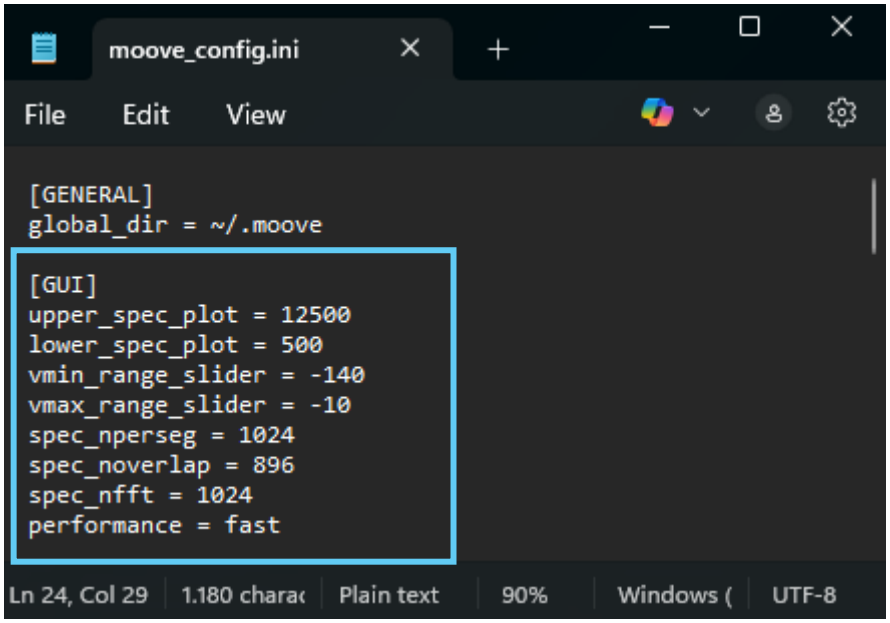


Figure 13: Config settings for the MooveGUI

The parameters that can be set in the config are described in the table below (Table 2).

Table 2: Config settings for the MooveGUI

Parameter	Default Value	Description
upper_spec_plot	12500	Upper frequency limit of the spectrogram shown in the main window of the GUI
lower_spec_plot	500	Lower frequency limit of the spectrogram shown in the main window of the GUI
vmin_range_slider	-140	Lower limit of the sliders to adjust the visual parameters of the spectrogram
vmax_range_slider	-10	Lower limit of the sliders to adjust the visual parameters of the spectrogram
spec_nperseg	1024	Defines the length of each segment for the STFT (short-time fourier transform). Shorter values lead to a better time, but a poorer frequency resolution.
spec_noverlap	896	Specifies the number of points to overlap between segments in the STFT. The smaller, the less continuous the frequency information is displayed.
spec_nfft	1024	Sets the number of points for the FFT (fast fourier transform) computation, determining the frequency resolution. Smaller values lead to a lower frequency resolution, calculation is faster.
performance	fast	Defines how the spectrogram in the GUI is calculated. fast: more bleeding (imshow) vs slow: more details but slower (pcolomesh)

3.2. Main window

With the first-time start, an example file containing a labeled song bout will open (bout_1.wav) (Fig 15). The upper plot of the GUI shows the **spectrogram** of the song file (Fig 15, ①) and the lower plot the corresponding **amplitude trace** (Fig 15, ②). The axis in between contains syllable **labels** (Fig 15, ③) and will be empty if data has not been labeled yet. On the right you can find a slider to adjust the visual parameters of your spectrogram. The minimum and maximum range can be set in the config (see 3.1. Setting the config) and when adjusted manually the current slider settings will be saved when closing the GUI.

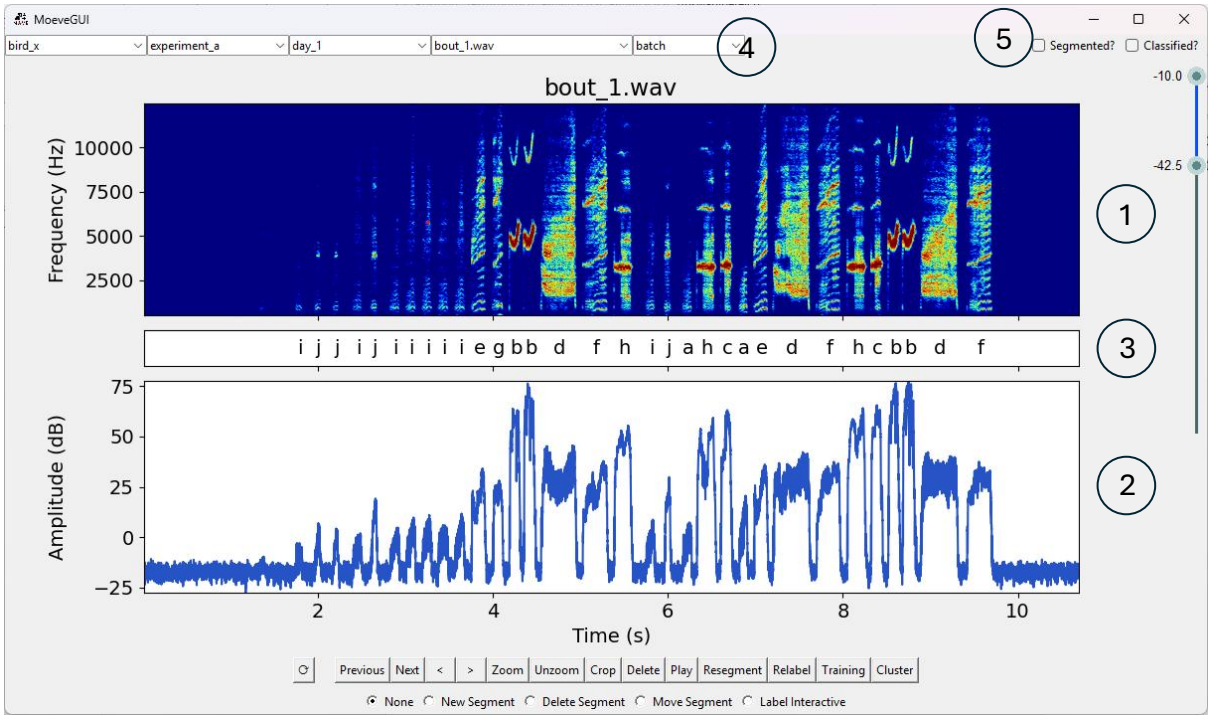


Figure 14: Main window of the MooveGUI

The top of the GUI contains a **navigation bar** (Fig 15, ④)(Fig 16), representing the current working path in your recorded data (*rec_data*), meaning bird folder (*bird_x*), experiment folder (*experiment_a*), day folder (*day_1*) and song file (*bout_1.wav*). In addition, the most-right drop-down menu shows the current batch file you're working on (by default *batch.txt*).

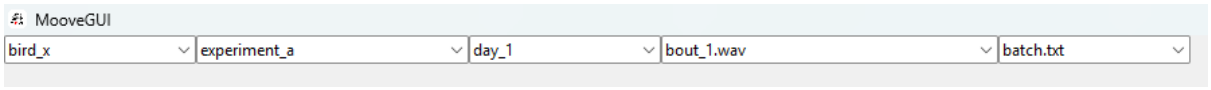


Figure 15: Navigation bar in the main window

The bottom row of the GUI contains multiple functional buttons to work on the current file (Fig 17). The **update** button '⟳' (Fig 17, ①) refreshes the GUI applying any changes performed on data folders and .rec files and updates each batch file. In case you are recording with MooveTaf on the same computer while working in the GUI, you can also press update to load your recently recorded files. The '**Previous**' and '**Next**' buttons (Fig 17, ②) enable switching between song files of one day. In case the file you are loading next contains a lot of data, the loading process might take a few seconds. The arrow buttons '**<**' and '**>**' (Fig 17, ③) let you move within the song file along the x-axis, which is especially helpful when zoomed in.

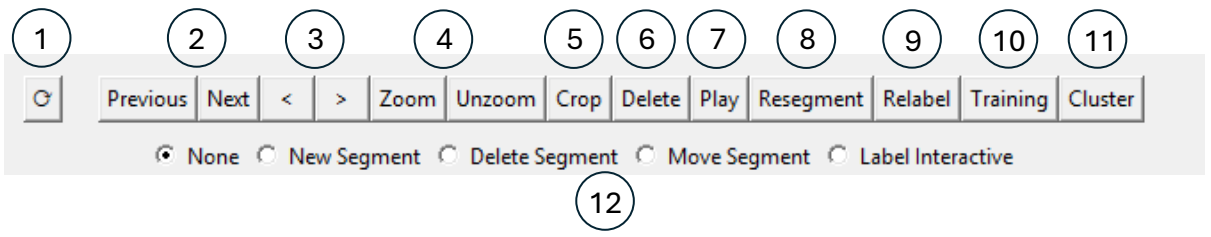


Figure 16: Bottom row of the main window containing functional buttons

Zooming can be performed on either the spectrogram or the amplitude trace plot by dragging your cursor and will be signaled by a red box (Fig 18). A rather unspecific zoom into the data can be performed using the '**Zoom**' button, while '**Unzoom**' moves back to default, showing the whole file (Fig 17, ④). Note that 'Zoom' reduces the x-axis range by 30%, staying around the center of your current axis.

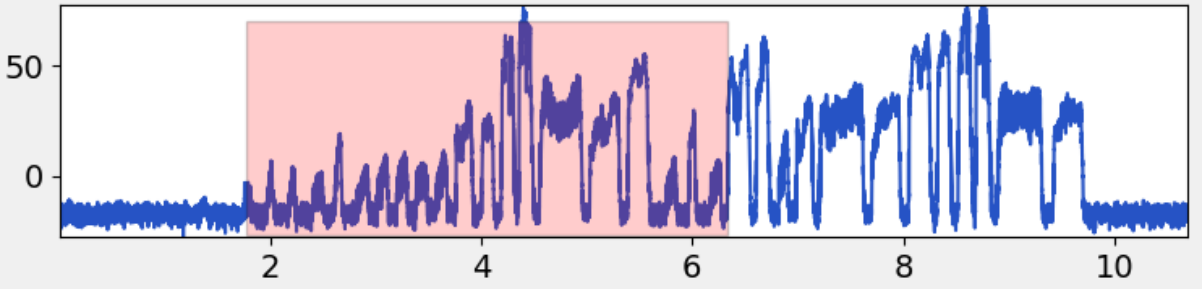


Figure 17: Zooming

Once you zoomed in, you can '**Crop**' (Fig 17, ⑤) your file to the currently shown x-axis range. Before cropping up the area, you will be asked to confirm the operation, as this will **delete** any excess data. The file names will not be changed but the title in the .rec file will show the date when the file was changed.

In case you want to delete the current file (for example noise files), the '**Delete**' button (Fig 17, ⑥) will give you the option to either remove its entry from the current batch (blue box) or remove its .wav-file, .rec-file and .not.mat-file from the current folder (red box)(Fig 19). **Note that this option will remove the file completely from your disk!**

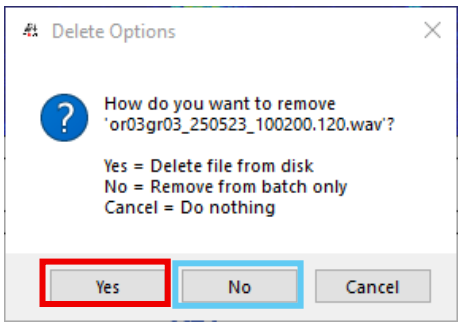


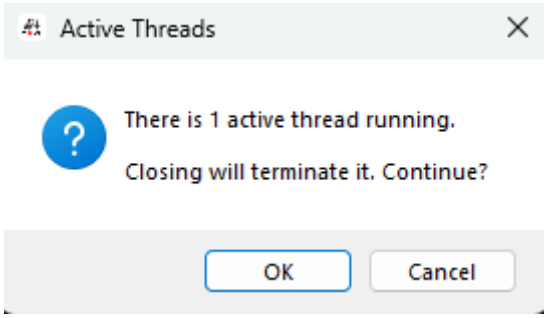
Figure 18: File delete options

The GUI will move on to the next file. In case you deleted the file from your disk, to remove the file entry from all your batch-files, simply press the **update** button. If you are working on the default batch file batch.txt, deleting the file 'from batch only' will not delete it permanently from this batch, as restarting the GUI will refill the batch.txt file with every existing .wav file in the folder. When working on the batch.txt file, only the option 'Delete file from disk' will permanently delete it from this batch file. In any other batch file removal from the batch will be permanent.

The '**Play**' button will play back the sound of the current file (Fig 17, ⑦). This playback process cannot be stopped, so if you want to listen to a specific part of a long sound file, consider zooming in on the relevant section first. It will only play the data currently shown.

The buttons '**Resegment**', '**Relabel**', '**Training**' and '**Cluster**', as well as the options below (Fig 17, ⑧ ⑨ ⑩ ⑪), are used to train networks and classify syllables and will be explained in detail in the following chapters, including the usage of the upper right check boxes in the main window.

It is highly recommended to close the GUI using the 'X' in the upper right corner, as this will save all your current settings, including the current file number and slider settings. If any processes are still running in the background the MooveGUI will ask you if you are sure to close despite the running threads. However, after using the DashGUI (Chapter 3.4.3.) and closing the DashGUI via the 'Close Dash GUI' button in the Cluster Window, MooveGUI has still pending threads open and the confirmation window opens.



In case you ever encounter an error when starting the GUI, such that it won't open at all, head to your .moove folder and delete the app_state.json file. This should only be done if needed, as the index of the file you're currently working on, as well as the slider settings will be set to default. However, all your immediate changes to a file, such as onset/ offset modification etc., will still be saved!

3.3. Syllable segmentation

3.3.1. Segmenting a file

In the first step of the data preprocessing pipeline, the individual syllables in each bout of the raw audio data need to be segmented. The '**Resegment'** button in the main window of the GUI (Fig 17, ⑧) will open the *Resegmentation window*. On the left side of the window (red box), the segmentation method provided by *evfuncs* (Nicholson, 2021) is implemented (Fig 20). The four radio buttons *Current File, Current Day, Current Experiment and Current Bird* let you decide which files the segmentation method should be applied to. This is done relative to the currently selected file. Selecting the *Current Day* button will use all the files in the directory of the currently selected day, selecting the *Current Experiment* or *Current Bird* button ensures that all files in the respective subdirectories are used. For every selection, the respective **batch files** will become visible in the drop-down menu on the right (blue box). By default, *All files* from the respective directory will be used. Choosing a specific batch file in the menu will only feed files from this batch into the dataset. With that, you have the option to load specific files from multiple days or experiments. You can also perform segmentation solely on the *Current File*.

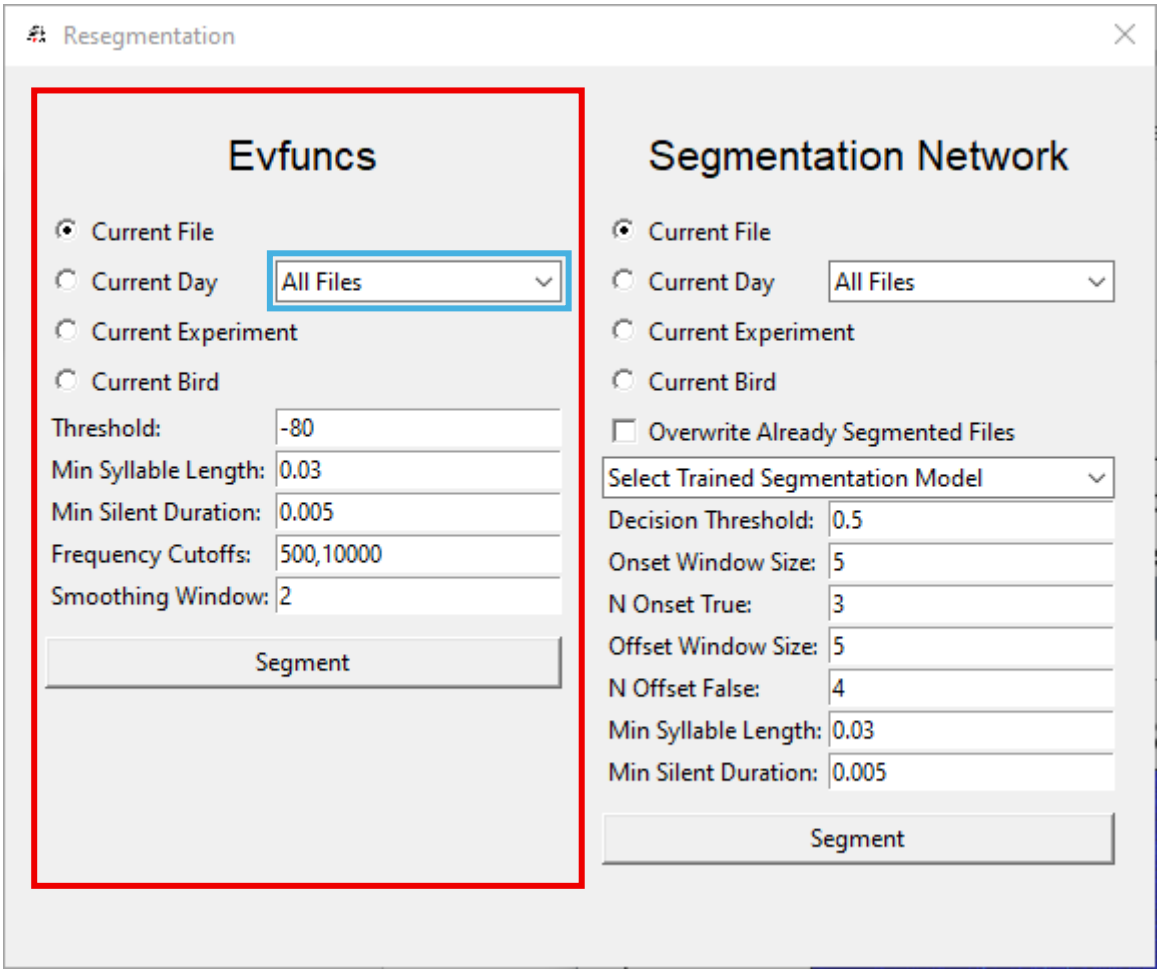


Figure 19: Resegmentation window, segmenting files using evfuncs

For the segmentation process, you can define five parameters that are explained in the table below (Table 3). Each parameter has a given default value. Pressing the button '**Segment**' will

start the segmentation process with the given parameters on the selected file(s), indicated by a green progress bar at the bottom of the window. This will determine syllable onsets and offsets in the raw audio data.

Table 3: Resegmentation parameters for using evfuncs

Parameter	Default Value	Description
Threshold	10 [Decibel]	Defines the threshold for detecting amplitude peaks as part of a syllable segment.
Min Syllable Duration	0.03 [seconds]	Sets the minimum duration of a syllable that a syllable segment must have.
Min Silent Duration	0.005 [seconds]	Sets the minimum silence between two syllables to be counted as separate units.
Frequency Cutoffs	(500, 10000) [Hertz]	Determines the lower and upper cutoff frequencies of the bandpass filter.
Smoothing Window	2 [milliseconds]	Defines the size of the time window for smoothing the signal.

Once the process is done, you will be informed, and the *Resegmentation window* will close. The syllable onsets and offsets assigned by the algorithm will become visible in the amplitude trace of the main window (Fig 21). Each segment will be labeled 'x' by default, visible in the middle plot. Furthermore, onset and offset times will be added to the .not.mat-file of each song file the segmentation has been performed on.

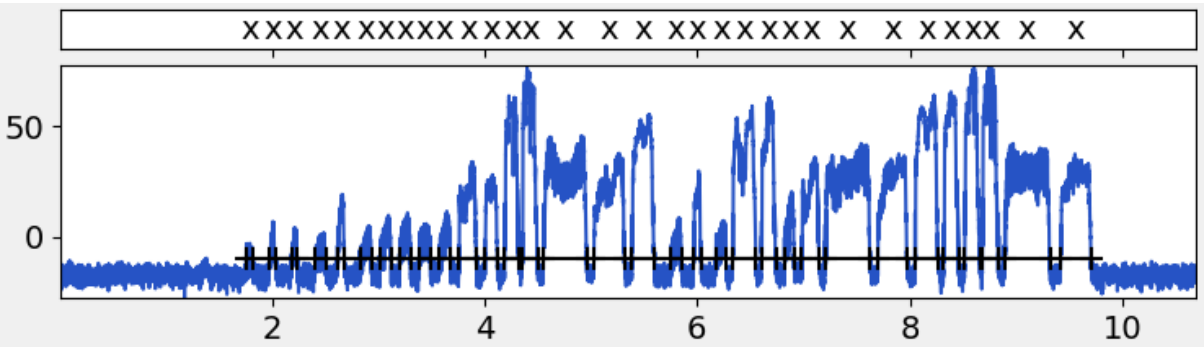
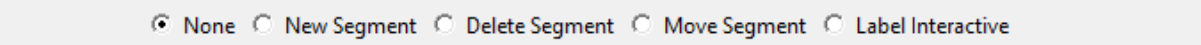


Figure 20: Segmentation of a file

The algorithm-driven segmentation might not be entirely accurate; thus, you can adjust your segments using the *segmentation bar* (Fig 17, ⑫). The options **New Segment** (shortcut "n"), **Delete Segment** (shortcut "d") and **Move Segment** (shortcut "m") are implemented as a group of radio buttons on the lower control bar of the GUI. Each option of the bar can be accessed via a shortcut, by simply pressing the respective button on the keyboard. Switching to that option will become visible in the bar. The option **Label Interactive** is part of the classification process and will be explained later.



If the **New Segment** option is selected, a syllable segment can be added with a left click for the onset and a right click for the offset in the area of the amplitude diagram. This segment is then temporarily labelled with the placeholder value 'x'. Selecting the **Delete Segment** option allows you to delete an existing syllable segment in the amplitude diagram by clicking on it. You can also click on the corresponding label to delete it. The **Move Segment** option allows the user to left-click on the marker of an existing onset or offset point in the amplitude diagram. The marker will be highlighted in red. A right-click on the desired position will move it to this new position. You can mark files for which he has manually verified the segmentation using the **Segmented** checkbox in the upper right corner of the main window (Fig 15, ⑤). This will in the following steps give you the option to specifically train a network based on previous segmentation. This information will be saved in the corresponding *.rec* file of the current song file.

3.3.2. Create a Segmentation Training Dataset

As soon as enough bouts have been segmented, the segmentation network can be trained. The training network is recommended to be an iterative process. Therefore, few bouts are first segmented manually, and the segmentation network is trained. The trained network can then be used to segment a bout that has not been segmented yet. Even if the segmentation of this bout is not yet perfect, the corresponding bout can then be corrected more quickly by hand and included in the set of bouts for the training dataset. By that, you can train the network on more and more hand-corrected segmented files. To create a training dataset out of your segmented files, press the '**Training'** button (Fig.15, ⑩) in the GUI main window to open the *Training window*.

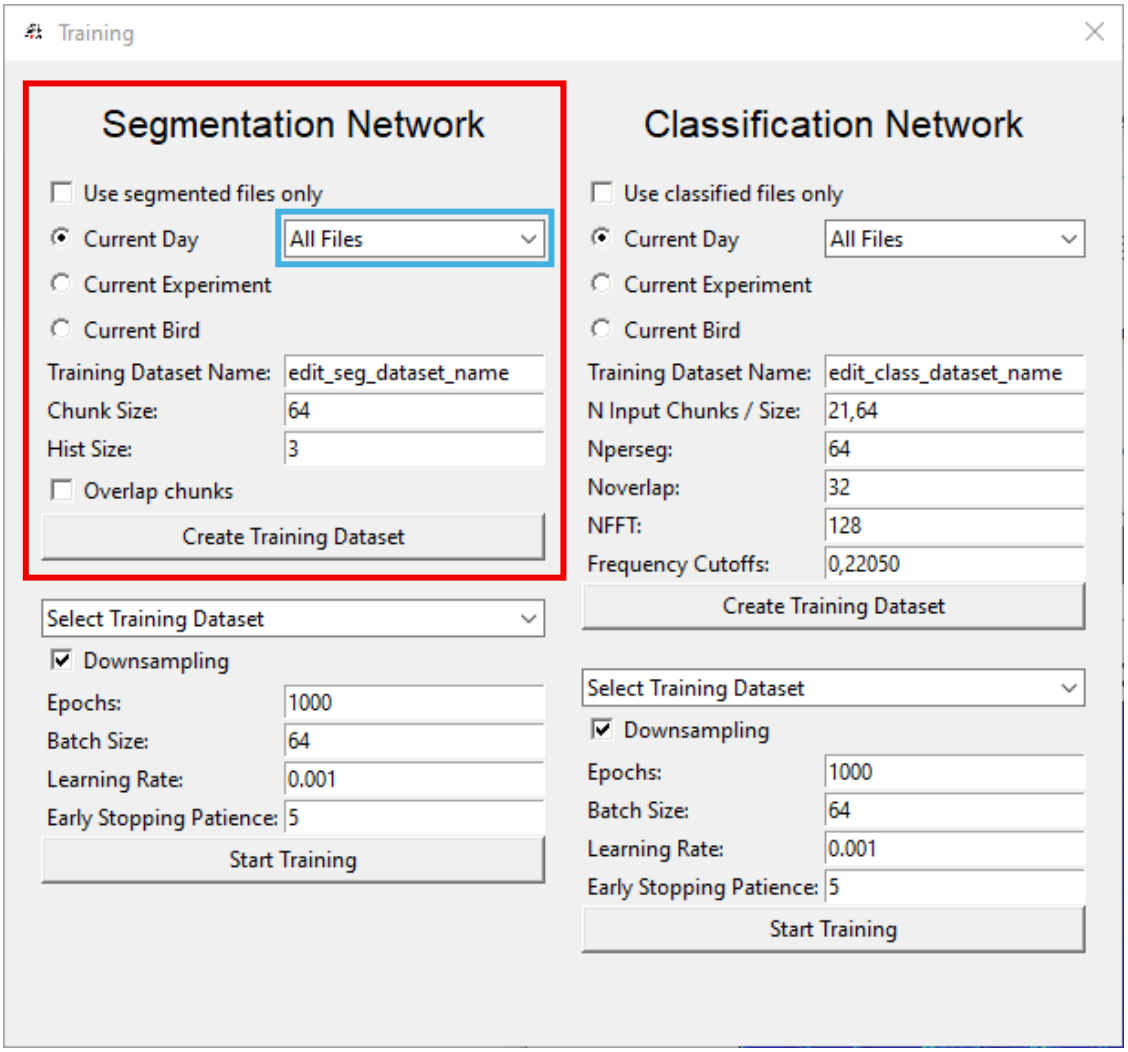


Figure 21: Creating a segmentation training dataset

The left part of the window is dedicated to the segmentation network, with the upper part enabling the creation of a training dataset (Fig 22, red box). With the upper four buttons you can choose which files to feed into the dataset. The options *Current Day*, *Current Experiment* and *Current Bird* will use all files in the respective subdirectories. For every selection, the respective **batch files** will become visible in the drop-down menu on the right (blue box). By default, *All files* from the respective directory will be used. Choosing a specific batch file in the menu will only feed files from this batch into the dataset. With that, you have the option to load specific files from multiple days or experiments. The option *Use Segmented Files Only* creates a dataset only containing the files in which the *segmentation checkmark* has been ticked (see above). This gives you the option to only feed files into the dataset that have already been manually checked or corrected. You must assign a name to the dataset in the *Training Dataset Name* field, the suffix *_seg* will be added automatically. The *Chunk Size* field below defines the step size of how many audio samples are fed to the segmentation network during inference (default value = "64"). This enables the segmentation network a fast detection of onsets, as the duration of each audio chunk is about 1.45ms using a sampling rate of 44.1kHz. The input field named *Hist Size* specifies how many previous audio chunks are added during the inference of a new audio chunk in the segmentation network (default value = 3). The checkbox *Overlap Chunks* determines

whether successive sequences of chunks should overlap in the training dataset. If selected, each new sequence is created containing chunks of the previous sequence. This will enlarge the training dataset as more overlapping data points are generated from the audio data. If the parameter is not selected, the sequences are created without overlapping so that each sequence is independent of the previous one. To avoid data leakage, this option should only be set if you feed **at least 7** segmented files into the network. You cannot create empty datasets. Pressing the button **Create Training Dataset** will start the process, indicated as '*Looking for Segments*' and followed by a green progress bar at the bottom of the window. Once the dataset is created you will be informed, and the *Training window* will close. With that, the content of the dataset, *Chunk Size* and *Hist Size* will be saved to a *.pkl* file in the folder *training_data*.

3.3.3. Train the Segmentation Network

Once a segmentation training dataset is created, the segmentation network can be trained via the *Training window* (Fig 23, red box).

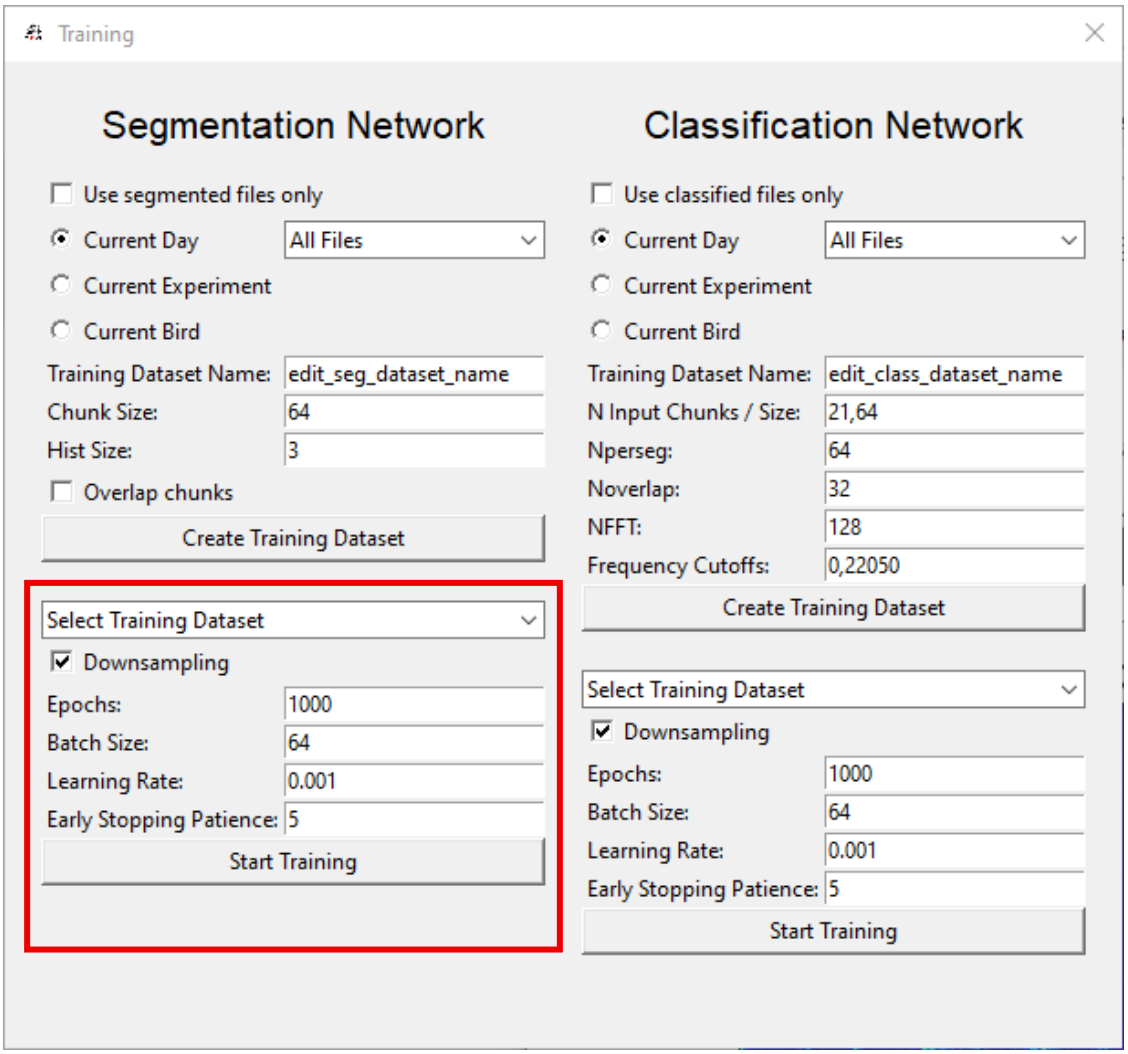


Figure 22: Training of the segmentation network

In the drop-down menu *Select Training Dataset* you can choose between your previously created training datasets. The parameters that can be set to train the network are explained in the table below (Table 4).

Note that we do not recommend downsampling if you're especially interested in 'repeats' or if your dataset contains syllables that only occur very rarely.

Table 4: Parameters for training the segmentation network

Parameter	Default Value	Description
Downsampling	True	Balances the dataset by downsampling each label to have an equal number of samples.
Epochs	1000	Specifies the number of epochs for training the neural network.
Batch Size	64	Defines the number of samples that will be propagated through the network at once during training.
Learning Rate	0.001	Controls the step size during the optimization process.
Early Stopping Patience	5	Sets the number of epochs without improvement of the validation data after which the training is terminated automatically. Higher early stopping patience can lead to overfitting.

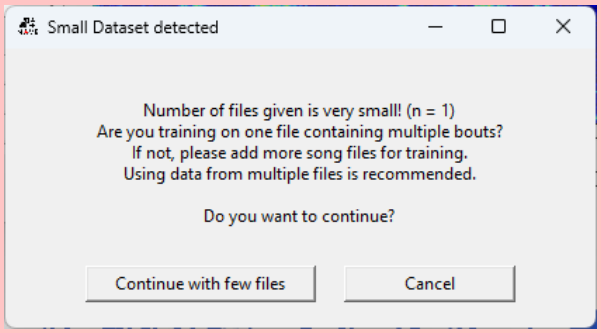
The *Start Training* button will train the segmentation network on the files from the selected training dataset. The training window will indicate the status of the training at the bottom, starting with '*Checking files*' for usability, switching to '*Training in Progress'* once you confirmed the start by pressing '*Ok*' and finally informing you when the training is finished, closing the *Training window*. The training progress can be observed in the terminal, where the current iteration of training (epoch) and the current accuracy of the network is shown (Fig 24).

```
2025-07-15 14:49:36,335 - INFO - Epoch 1/1000, Train Loss: 0.1810, Val Loss: 0.2701, Train Accuracy: 0.9259, Val Accuracy: 0.9300
2025-07-15 14:50:07,462 - INFO - Epoch 2/1000, Train Loss: 0.1321, Val Loss: 0.1028, Train Accuracy: 0.9471, Val Accuracy: 0.9550
2025-07-15 14:50:38,612 - INFO - Epoch 3/1000, Train Loss: 0.1157, Val Loss: 0.0875, Train Accuracy: 0.9537, Val Accuracy: 0.9664
2025-07-15 14:51:07,832 - INFO - Epoch 4/1000, Train Loss: 0.1085, Val Loss: 0.0881, Train Accuracy: 0.9565, Val Accuracy: 0.9647
2025-07-15 14:51:37,585 - INFO - Epoch 5/1000, Train Loss: 0.1029, Val Loss: 0.0774, Train Accuracy: 0.9589, Val Accuracy: 0.9702
2025-07-15 14:52:08,967 - INFO - Epoch 6/1000, Train Loss: 0.0975, Val Loss: 0.0908, Train Accuracy: 0.9610, Val Accuracy: 0.9637
2025-07-15 14:52:40,453 - INFO - Epoch 7/1000, Train Loss: 0.0942, Val Loss: 0.0797, Train Accuracy: 0.9628, Val Accuracy: 0.9692
2025-07-15 14:53:11,918 - INFO - Epoch 8/1000, Train Loss: 0.0885, Val Loss: 0.0684, Train Accuracy: 0.9649, Val Accuracy: 0.9736
2025-07-15 14:53:43,040 - INFO - Epoch 9/1000, Train Loss: 0.0824, Val Loss: 0.0632, Train Accuracy: 0.9678, Val Accuracy: 0.9755
2025-07-15 14:54:14,344 - INFO - Epoch 10/1000, Train Loss: 0.0802, Val Loss: 0.0610, Train Accuracy: 0.9685, Val Accuracy: 0.9757
2025-07-15 14:54:45,695 - INFO - Epoch 11/1000, Train Loss: 0.0772, Val Loss: 0.0609, Train Accuracy: 0.9695, Val Accuracy: 0.9761
2025-07-15 14:55:17,115 - INFO - Epoch 12/1000, Train Loss: 0.0766, Val Loss: 0.0630, Train Accuracy: 0.9698, Val Accuracy: 0.9748
2025-07-15 14:55:49,096 - INFO - Epoch 13/1000, Train Loss: 0.0737, Val Loss: 0.0744, Train Accuracy: 0.9707, Val Accuracy: 0.9703
2025-07-15 14:56:21,859 - INFO - Epoch 14/1000, Train Loss: 0.0716, Val Loss: 0.0585, Train Accuracy: 0.9714, Val Accuracy: 0.9770
2025-07-15 14:56:55,700 - INFO - Epoch 15/1000, Train Loss: 0.0692, Val Loss: 0.0567, Train Accuracy: 0.9725, Val Accuracy: 0.9778
2025-07-15 14:57:30,234 - INFO - Epoch 16/1000, Train Loss: 0.0701, Val Loss: 0.0566, Train Accuracy: 0.9720, Val Accuracy: 0.9779
2025-07-15 14:58:05,188 - INFO - Epoch 17/1000, Train Loss: 0.0686, Val Loss: 0.0599, Train Accuracy: 0.9730, Val Accuracy: 0.9768
2025-07-15 14:58:40,648 - INFO - Epoch 18/1000, Train Loss: 0.0669, Val Loss: 0.0589, Train Accuracy: 0.9735, Val Accuracy: 0.9768
2025-07-15 14:59:16,299 - INFO - Epoch 19/1000, Train Loss: 0.0674, Val Loss: 0.0561, Train Accuracy: 0.9731, Val Accuracy: 0.9782
2025-07-15 14:59:52,262 - INFO - Epoch 20/1000, Train Loss: 0.0653, Val Loss: 0.0587, Train Accuracy: 0.9742, Val Accuracy: 0.9773
2025-07-15 15:00:28,211 - INFO - Epoch 21/1000, Train Loss: 0.0657, Val Loss: 0.0555, Train Accuracy: 0.9738, Val Accuracy: 0.9782
2025-07-15 15:01:04,139 - INFO - Epoch 22/1000, Train Loss: 0.0647, Val Loss: 0.0616, Train Accuracy: 0.9742, Val Accuracy: 0.9761
2025-07-15 15:01:42,645 - INFO - Epoch 23/1000, Train Loss: 0.0645, Val Loss: 0.0568, Train Accuracy: 0.9743, Val Accuracy: 0.9775
2025-07-15 15:02:24,079 - INFO - Epoch 24/1000, Train Loss: 0.0631, Val Loss: 0.0559, Train Accuracy: 0.9747, Val Accuracy: 0.9783
2025-07-15 15:03:02,235 - INFO - Epoch 25/1000, Train Loss: 0.0629, Val Loss: 0.0580, Train Accuracy: 0.9750, Val Accuracy: 0.9780
```

Figure 23: Iteration of training the network

The trained model can be found as a *.pth* file in the *trained_models* directory, as well as *_mean.pt* and *_std.pt*.

To train a network, at least **8 segments** must be defined in the given files. Furthermore, if the dataset consists of at least **7 segmented files**, the data will be split between files to form the training data, validation data and test data set. Splitting data by files prevents data leakage and provides more reliable accuracy results. However, you can still train a network on less than 7 files, for example if you have very long song files containing multiple bouts and segments. The GUI will ask you whether you want to continue with only a few files.



Pressing **Continue with few files** will train a network on these files (if they contain at least 8 segments) by not splitting between files. Therefore, training data, validation data and test data sets will contain segments from the same file. This is in general not recommended and accuracy values can be less reliable. Pressing **Cancel** will bring you back to the *training window*.

3.3.4. Resegment using the Trained Network

Once you have trained your segmentation network, you can use it to segment files. For that purpose, open the *Resegmentation window* using the **Resegment** button in the main window. On the right half, the trained network can now be parameterized and applied (Fig 25, red box). Again, the options *Current File, Current Day*, *Current Experiment* and *Current Bird* let you choose which files to resegment. For every selection, the respective **batch files** will become visible in the drop-down menu on the right (Fig 25, blue box). By default, *All files* from the respective directory will be used. Choosing a specific batch file in the menu will only feed files from this batch into the dataset. With that, you have the option to load specific files from multiple days or experiments. With the tickbox *Overwrite Already Segmented Files* you can decide whether files that have already been manually segmented (and marked as *Segmented*, see above) should be overwritten and segmented by the network. Ticking the box will enable resegmentation of these files. In the drop-down menu *Select Trained Segmentation Model* you can select the desired trained segmentation model. Its content is generated from all saved segmentation models in the *trained_models* directory.

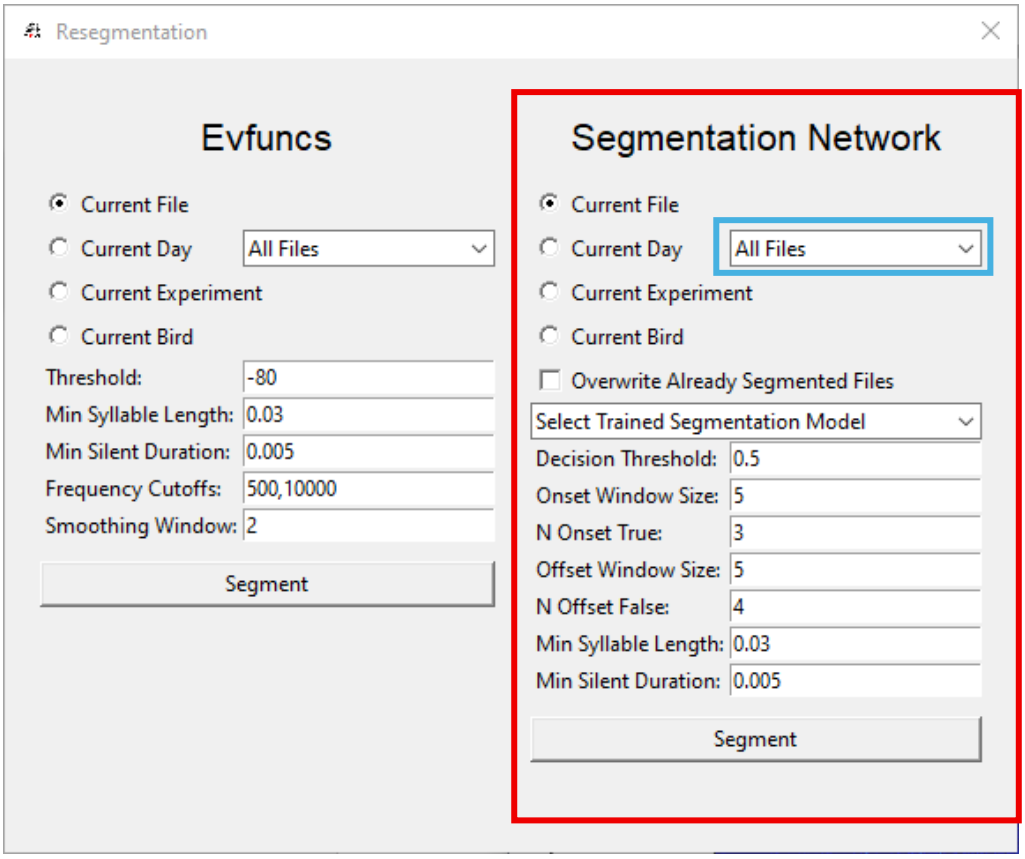


Figure 24: Resegment using the training segmentation network

The resegmentation parameters can be adjusted below and are explained in the following table (Table 5). Eventually, pressing the **Segment** button at the bottom of the window will start the resegmentation process of the selected files, indicated by a green progress bar at the bottom of the window. Once all files are resegmented, you will be informed and the *Resegmentation window* will close.

Table 5: Parameters for resegmenting using a trained network

Parameter	**Default Value**	**Description**
Decision Threshold	0.5 [%]	Defines a probability threshold for detecting a syllable segment.
Onset Window Size	5 [chunks]	Specifies the size of the sliding window used to detect onsets.
N Onset True	3 [chunks]	Sets the number of *True* detections within the sliding window.
Offset Window Size	5 [chunks]	Specifies the size of the sliding window used to detect offsets.
N Offset False	4 [chunks]	Sets the number of *False* detections within the sliding window.
Min Syllable Length	0.03 [seconds]	Specifies the minimum duration of a syllable that a syllable segment must have.
Min Silent Duration	0.005 [seconds]	Specifies the minimum silence between two syllables to be counted as separate units.

3.4. Label Clustering

3.4.1. Create a Cluster Training Dataset

To obtain the syllable labels for the training dataset of the classification network, the dimensionality reduction method UMAP is used together with a clustering algorithm. The input for UMAP consists of the individual spectrograms of the identified syllable segments. For this purpose, a cluster dataset containing the spectrograms of the syllable segments needs to be created, which can be done in the *Cluster window*, available via the **Cluster** button in the main window (Fig 17, 11).

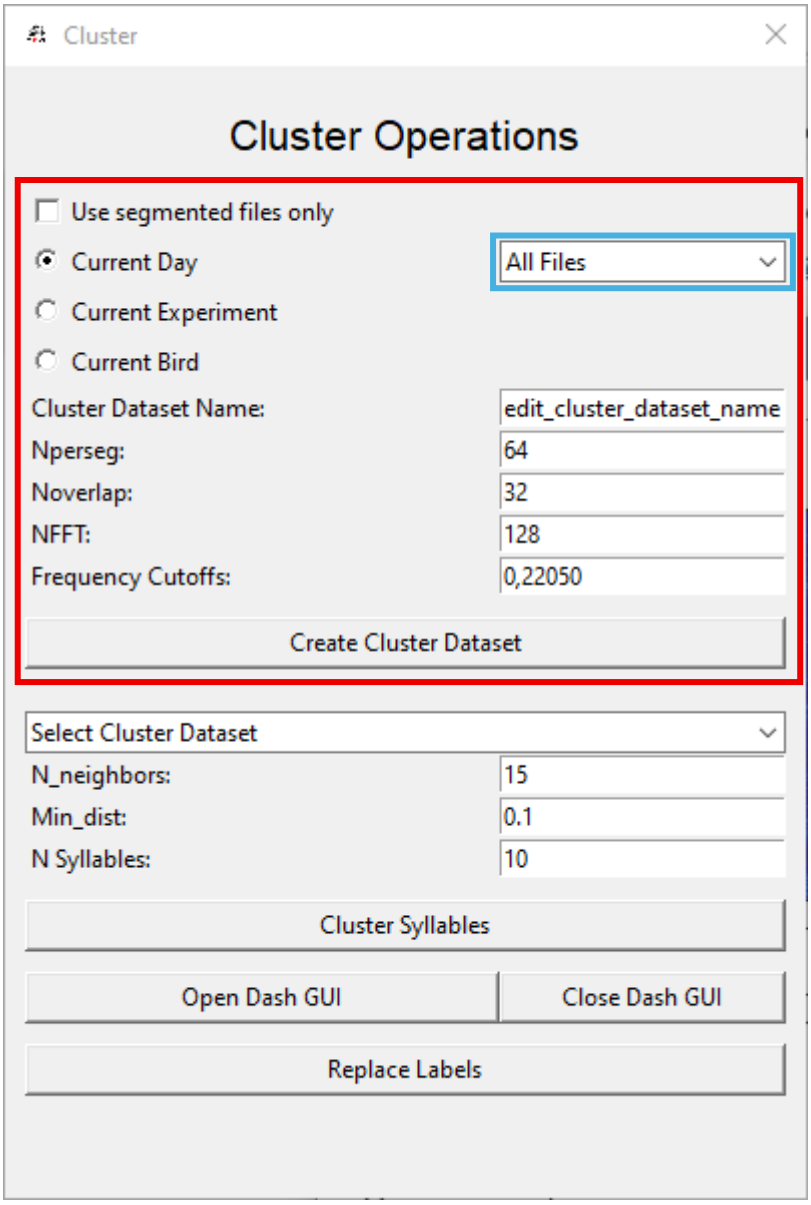


Figure 25: Creating a cluster training dataset

In the upper part of the window (Fig 26, red box), the options *Current File, Current Day, Current Experiment* and *Current Bird* define the files the clusters will be created from. For every selection, the respective **batch files** will become visible in the drop-down menu on the right (Fig 26, blue box). By default, *All files* from the respective directory will be used. Choosing a specific batch file

in the menu will only feed files from this batch into the dataset. With that, you have the option to load specific files from multiple days or experiments. The checkbox **Use segmented files only** targets only files that have already been manually segmented (and marked as *Segmented*, see above). You must assign a name in the *Cluster Dataset Name* field, the suffix *_clus* will be added automatically. The adjustable parameters will be used in the spectrogram calculation and are described in the table below.

Table 6: Parameters for creating a cluster dataset

Parameter	Default Value	Description
Nperseg	64	Defines the length of each segment for the STFT (short-time fourier transform). Shorter values lead to a better time, but a poorer frequency resolution.
Noverlap	32	Specifies the number of points to overlap between segments in the STFT. The smaller, the less continuous the frequency information is displayed.
NFFT	128	Sets the number of points for the FFT (fast fourier transform) computation, determining the frequency resolution. Smaller values lead to a lower frequency resolution, calculation is faster.
Frequency Cutoffs	0,22050 [Hertz]	Defines the lower and upper cutoff frequency for filtering the spectrogram.

Pressing the button Create Cluster Dataset will start the process, indicated by a green process bar at the bottom of the *Cluster window*. Once the clustering is done, you will be informed, and the *Cluster window* will close. The dataset will be saved as *.pkl* file in the *cluster_data* folder.

3.4.2. Cluster Syllables

Once the cluster dataset is created, the dimensionality reduction using UMAP can be started. For that, the created dataset can be selected in the lower part of the *Cluster window* (Fig 27, red box).

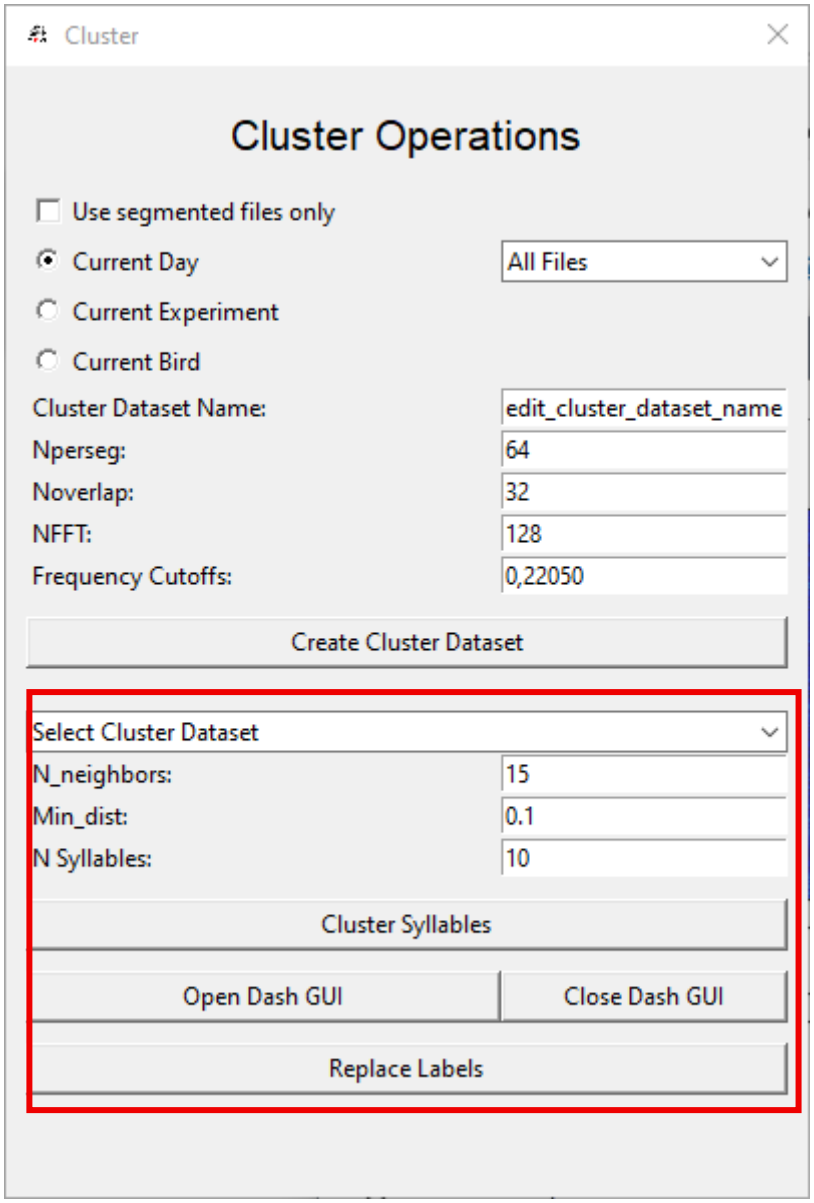


Figure 26: Clustering of syllables

Below, the input parameters for the UMAP algorithm and the following k-Means algorithm can be set (Table 7). The button **Cluster Syllables** will start the process, indicated by the '*Running*' label at the bottom of the window.

Table 7: Parameters for clustering syllables

Parameter	Default Value	Description
N neighbors	15	Determines the number of nearest neighbors *k* when constructing the high-dimensional graph.
Min _dist	0.1	Controls the minimum distance between points in the low-dimensional space. A smaller value leads to denser clusters.
N Syllables	10	Defines the number of syllable clusters to be formed for the k-Means algorithm. Re-adjust if the number of clusters created is not the yellow from the egg.

Once the clustering is completed, the results will be saved to the *.pkl* file, together with a same-named *.png* file of the 2D UMAP space containing the syllable clusters. An interactive version of the UMAP clustering can be opened with the button **Open Dash GUI**, which will start in a separate thread in your browser.

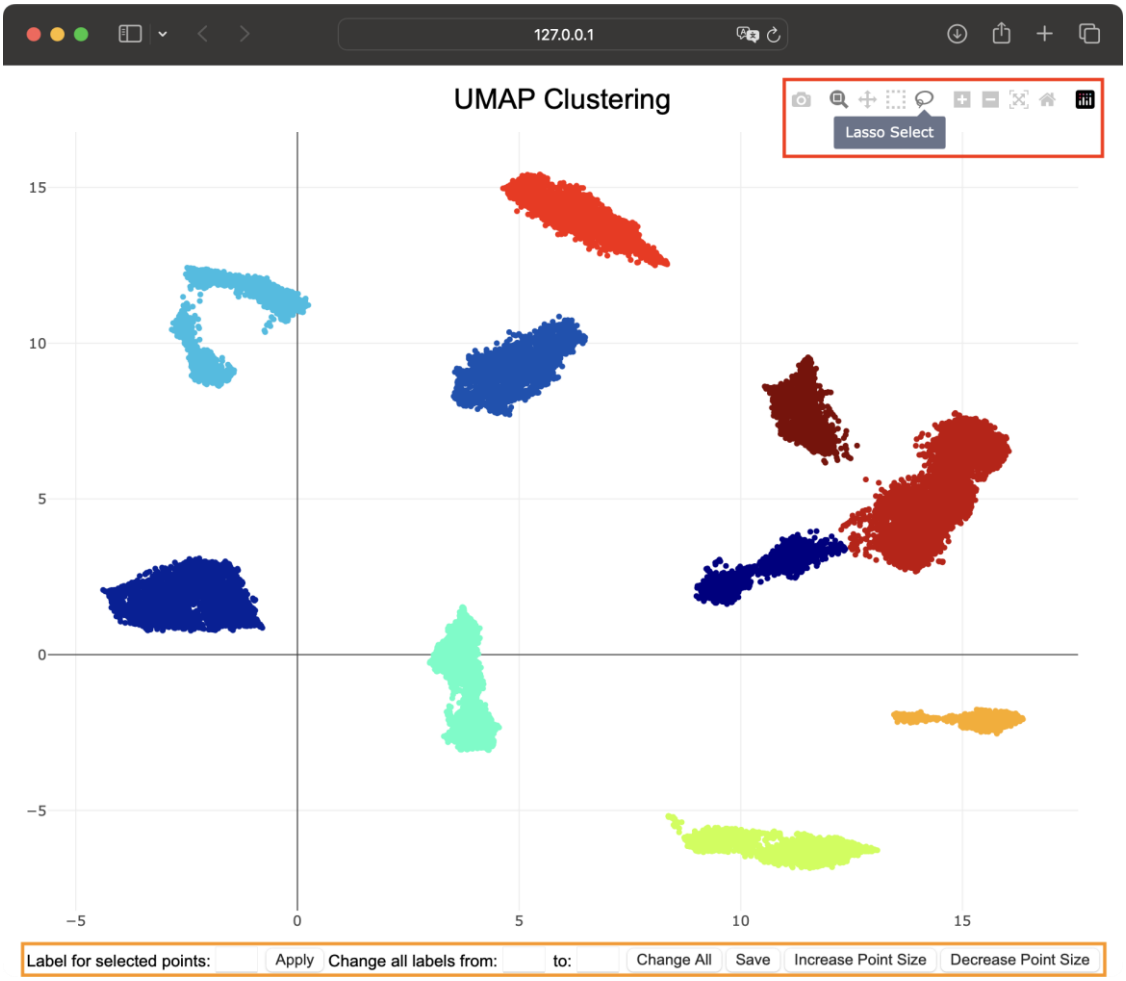


Figure 27: Dash GUI containing your syllable clusters

3.4.3. Dash GUI

Each colored cluster in the space represents one classified syllable type, each dot represents one syllable. The clusters will be labeled with letters, starting at '*a*'. The Dash GUI offers multiple ways to interact with the data and to reassign the cluster membership of the data points as desired. In the tool bar located in the upper right corner (Fig 28, red box), you can activate the **zoom** tool (Fig 29, 2) to closely inspect the data, **move** around (Fig 29, 3) the plot or **zoom in and out** (Fig 29, 6+7) more directly. The **autoscale** button (Fig 29, 8) will move the plot to your clusters position and **reset axes** (Fig 29, 9) will reset the plot. The **save** button (Fig 29, 1) will save the plot as a *.png* file.Also, **double clicking** the plot will zoom out, autoscale to your clusters position and remove any existing selection boxes. To relabel specific data points, the option **Box Select** (Fig 29, 4) lets you draw a box around a specific set of dots, and **Lasso Select** (Fig 29, 5) lets you draw a free form.

Figure 28: Options in the Dash GUI

Once you selected points, you can relabel these points directly by typing the new label into the *Label for selected points* field at the left bottom of the Dash window (Fig 28, orange box). Pressing the **Apply** button will change the label of the selected points and depending on the letter distance to the other syllables, the color space is adjusted, possibly leading to a different color mapping than before. Furthermore, you can change all dots from one label at once, by typing the current label of the cluster in the *Change all labels from* filed, and the new desired label in the *to:* field next to it. Pressing **Change All** will change the label of this cluster. In the lower right corner, the buttons **Increase Point Size** and **Decrease Point Size** give you the option to change the dot size for better visibility. When you applied your desired changes, press the **Save** button in the middle bottom part of the Dash GUI to overwrite your previous cluster data. Saving the data will be indicated by a message showing up in the MooveGUI. Once you're done, press the **Close Dash GUI** button in the *Cluster window* of the MooveGUI. This will shut down the Dash server and you can then close the browser window. The Dash GUI **must be closed** using the **Close Dash GUI** button before it can be reopened, as the server will not be available otherwise.

Finally, you can apply your newly acquired syllable labels to your data by pressing the button **Replace Labels** in the *Cluster window*. This will replace all previous placeholder labels 'x' (or other labels) in the *.not.mat* files that have been fed into the dataset (as defined in *Create a Cluster Training Dataset*) and the new labels will appear in the GUI.

3.5. Syllable classification

3.5.1. Create a Classification Training Dataset

Although the classification by the process with UMAP, k-Means and a following manual adjustment of the cluster memberships is very accurate, individual syllables can be classified incorrectly. You can semi-manually review all files and correct any mislabelled syllables using the *Label Interactive* option in the main window (Fig 17, 12). To do so, clicking on the label you want to change will highlight it in red, and typing on your keyboard will replace it. Valid label characters are basic letters and numbers (no capital letters and special characters). After relabeling one syllable, the highlight jumps to the next syllable, enabling continuous relabeling till the end of the file. Corrected labels get automatically saved in the *.not.mat* file of the corresponding bout. Analogous to the procedure for the segmentation network, you can use the **Classified** checkbox in the top right corner (5, see above) to mark these files. This information will be saved in the corresponding *.rec* file of the current song file. Note that while you are in the *Label interactive* mode, using shortcuts to switch modes is not possible, as the keys will be used for relabelling.

Once you all labels are correct, a training dataset can be created in the right upper part of the *training window* (Fig 17, 10)(Fig 30, red box). With the upper four buttons you can choose which files to feed into the dataset. The options *Current Day*, *Current Experiment* and *Current Bird* will jump to the respective folder direction and load all batch files found in those. For every selection, the respective **batch files** will become visible in the drop-down menu on the right (blue box). By default, *All files* from the respective directory will be used. Choosing a specific batch file in the menu will only feed files from this batch into the dataset. With that, you have the option to load specific files from multiple days or experiments. The option *Use Classified Files Only* creates a dataset only containing the files in which the *classification checkmark* has been ticked (see above). This gives you the option to only feed files into the dataset that have already been manually checked or corrected. You must assign a name to the dataset in the *Training Dataset Name* field, the suffix *_class* will be added automatically.

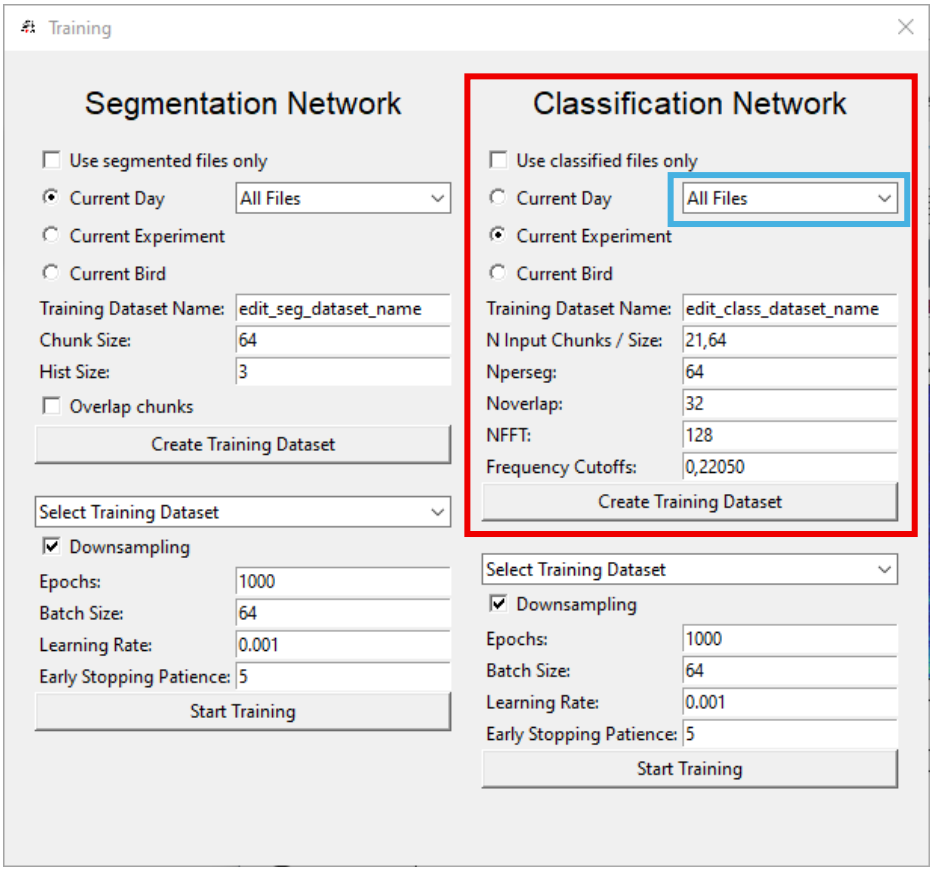


Figure 29: Creating a classification training dataset

The training dataset for the classification network is generated from the spectrogram data of the individual syllable segments and their corresponding syllable label. For the classification network, only a fixed time interval after a detected onset is used as input (*N Input Chunks/Size*). This parameter can be set below among others, as described in the table below (Table 8). You cannot create empty datasets.

Table 8: Parameters for creating a classification training dataset

Parameter	Default Value	Description
N Input Chunks/Size	21,64	Length of time interval after onset that is used as input for classification (~30.48 ms at 44.1kHz)
Nperseg	64	Defines the length of each segment for the STFT (short-time fourier transform). Shorter values lead to a better time, but a poorer frequency resolution.
Noverlap	32	Specifies the number of points to overlap between segments in the STFT. The smaller, the less continuous the frequency information is displayed.
NFFT	128	Sets the number of points for the FFT (fast fourier transform) computation, determining the frequency resolution. Smaller values lead to a lower frequency resolution, calculation is faster.
Frequency Cutoffs	0,22050 [Hertz]	Defines the lower and upper cutoff frequency for filtering the spectrogram.

Pressing the button **Create Training Dataset** will start the process, indicated as '*Looking for Syllables'* and followed by a green progress bar at the bottom of the window. Once the dataset is created you will be informed, and the *Training window* will close. With that, the content of the dataset will be saved to a *.pkl* file in the *training_data* folder.

3.5.2. Training the Classification Network

Once a classification training dataset is created, the classification network can be trained via the *Training window* (Fig 31, red box).

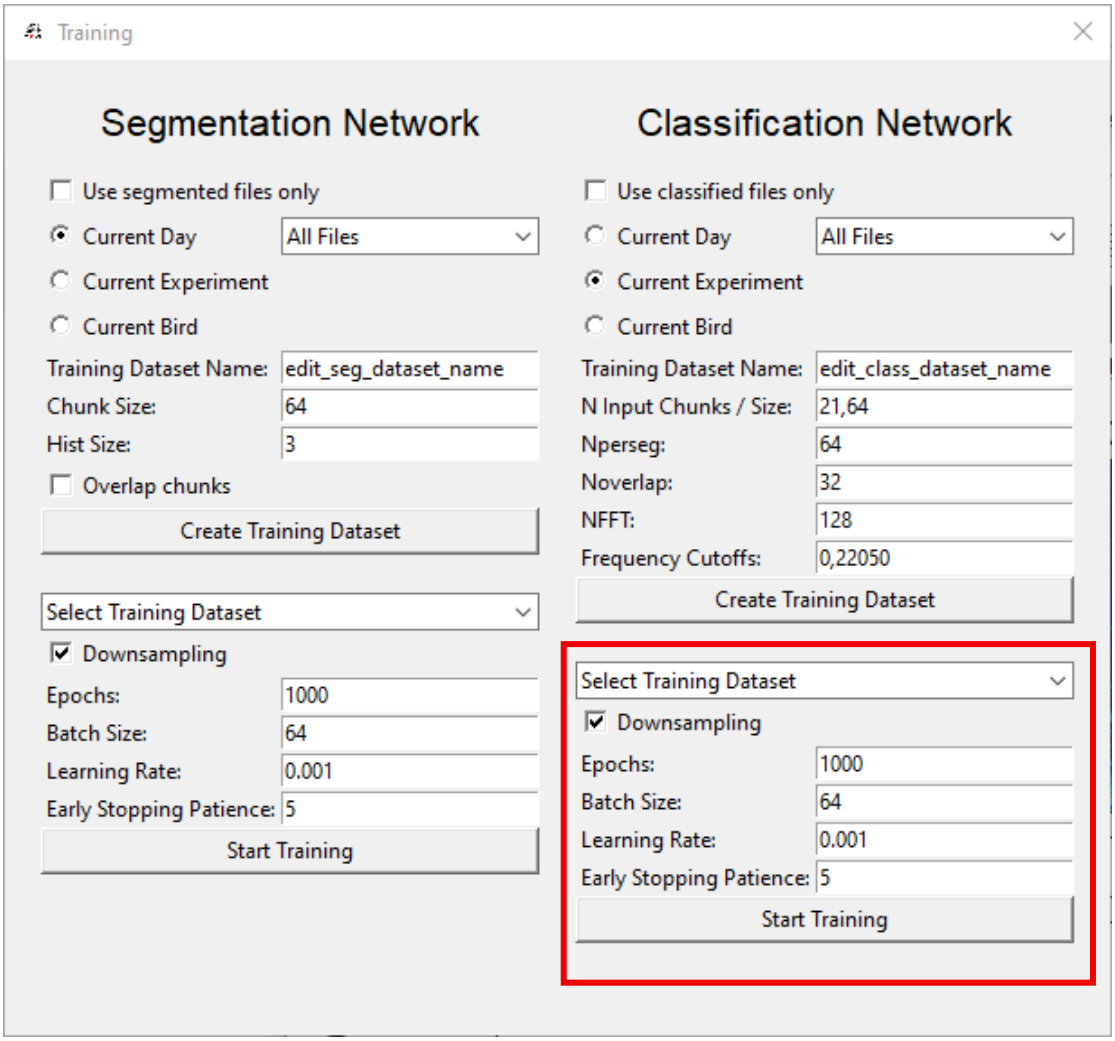


Figure 30: Training the classification network

In the drop-down menu *Select Training Dataset* you can choose between your previously created training datasets. The parameters that can be set to train the network are explained in the table below.

We do not recommend downsampling if you're especially interested in 'repeats' or if your dataset contains syllables that only occur very rarely.

Table 9: Parameters for training the classification network

Parameter	Default Value	Description
Downsampling	True	Balances the dataset by downsampling each label to have an equal number of samples.
Epochs	1000	Specifies the number of epochs for training the neural network.
Batch Size	64	Defines the number of samples that will be propagated through the network at once during training.
Learning Rate	0.001	Controls the step size during the optimization process.
Early Stopping Patience	5	Sets the number of epochs without improvement of the validation data after which the training is terminated automatically. Higher early stopping patience can lead to overfitting.

The *Start Training* button will train the classification network on the files from the selected training dataset. The start of the training requires confirmation via pressing the '*OK*' button. The training window will indicate the status of the training at the bottom, starting with '*Checking files*' for usability, switching to '*Training in Progress*' once the training has started and finally informing you when the training is finished and closing the *Training window*. The trained model can be found as a *.pth* file in the *trained_models* directory, together with a *.svg* file containing the classification matrix. This matrix shows the performance of the network as the accuracy of the predictions for each type of syllable on the test subset.

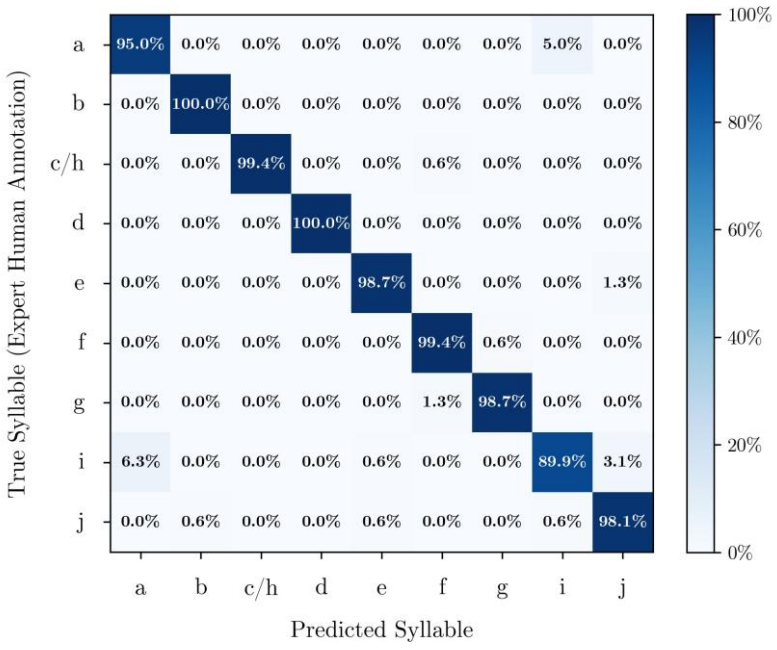
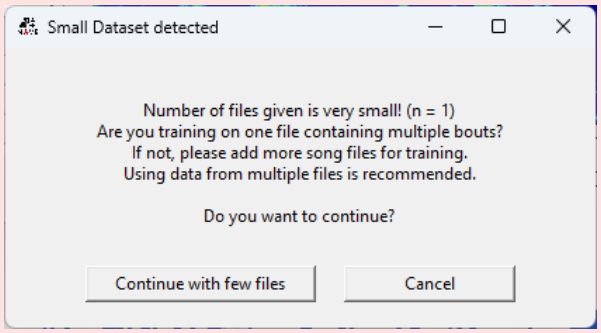


Figure 31: Example classification matrix

To train a network, at least **6 syllables per syllable type** must be defined in the given files. Furthermore, if the dataset consists of at least **7 classified files**, the data will be split between files to form the training data, validation data and test data set. Splitting data by files prevents data leakage and provides more reliable accuracy results. However, you can still train a network on less than 7 files, for example if you have very long song files containing multiple bouts and syllables. The GUI will ask you whether you want to continue with only a few files.



Pressing **Continue with few files** will train a network on these files (if they contain at least 6 syllables per syllable type) by not splitting between files. Therefore, training data, validation data and test data sets will contain syllables from the same file. This is in general not recommended and accuracy values can be less reliable. Pressing **Cancel** will bring you back to the *training window*.

3.5.3. Relabel Data

Lastly, you can apply your trained classification network on already labeled data by opening the *Relabel window* using the **Relabel** button in the main window (Fig 17, 9)(Fig 33). Again, the four upper buttons define which files should be relabeled by going into the respective subdirectories, including the selection of a specific batch file (blue box). By default, *All Files* of your selection will be used. With the tickbox *Overwrite Already Segmented Files* you can decide whether files that have already been manually classified (and marked as *Classified*, see above) should be overwritten and classified by the network. Ticking the box will enable relabeling of these files. In the drop-down menu *Select Trained Classification Model* you can select the desired trained classification model. Its content is generated from all saved classification models in the *trained_models* directory. When pressing the **Relabel** button, the replacement of labels will be started indicated by a green progress bar at the bottom of the *Relabel* window. Once all labels are replaced, you will be informed and the *Relabel window* will be closed.

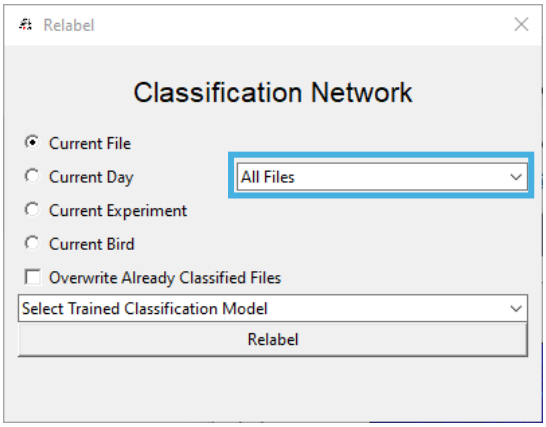


Figure 32: Relabel data using the classification network

4. REC file and Feedback Information from Training

While recording with MooveTaf, information about each recorded file will be saved in a .rec file within the specific *rec_data* folder, along with the .wav file and the .not.mat file. This .rec file contains specific information set in the config (see 2. MooveTaf) as well as information about the song file generated during the recording (Fig 34).

The first line shows the exact date and time point at which the file was created, thus when the bout was saved. The pink highlighted part describes the time at which the recording started, **begin rec** (usually 0 ms). **trig time** describes the time at which the recording was triggered, which can be set in the config (see above). It describes how much time is included before the recording was triggered. The end of the recording is shown as **rec end**. The blue boxes describe general parameters such as the recording frequency **ADFREQ** (here: 44100Hz), the number of channels **Chans** (here: 1), the number of recorded **Samples** (here: 312320) and the time included before (**T before**, here: 2ms) and after (**T after**, here: 1ms) recording was triggered, which can be set in the config (see above). Below in the green box, the lines **Hand Segmented** and **Hand Classified** (here: 1) depict whether you ticked the checkboxes for manual segmentation and classification in the MooveGUI for this file (see 3. MooveGUI). Once the checkbox is set, the line will be set to 1 else it is 0. The yellow boxes depict feedback that was directly generated during the recording and training session. You can check whether the current file was labeled as a catch file during training, and therefore no feedback was given (**Catch Song**, here: 0 and 1). This will be set to 1 in case of a catch file and 0 in case of not-catch files with information about the feedback if feedback was triggered. The amount of catch trials during recording can be set in the config (see above). In case feedback was given during this song bout, the **time points** of the feedback will be written below. Furthermore, following the **FB #** the name of your **used feedback file** (as given in the config) will be shown, and the number of the template (by default 0).

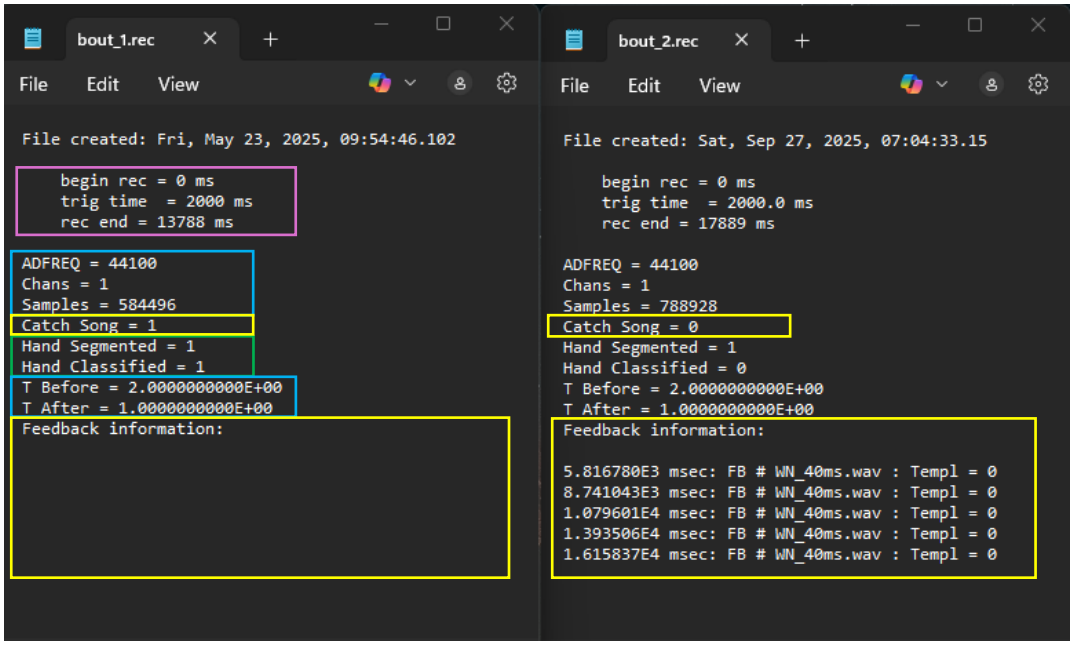


Figure 33: Rec files

The feedback given during the training will also be directly shown when opening the file in the GUI. **Catch files** will have a red box drawn around the syllable labels, and additionally the word ‘catch’ will be shown in the upper left above the spectrogram (Fig 35).

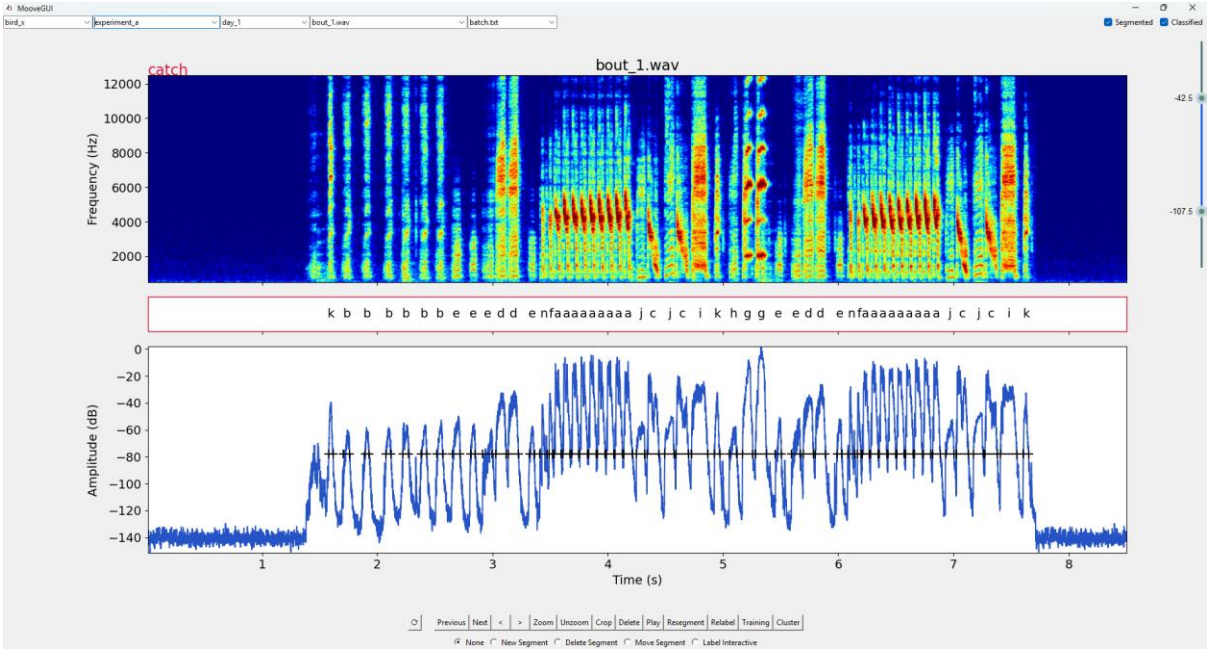


Figure 34: Marking of catch files

In the case of **feedback**, for example white noise playback, the exact time points of the feedback will be shown as small red triangles below the spectrogram, showing the beginning of feedback playback (Fig 36).

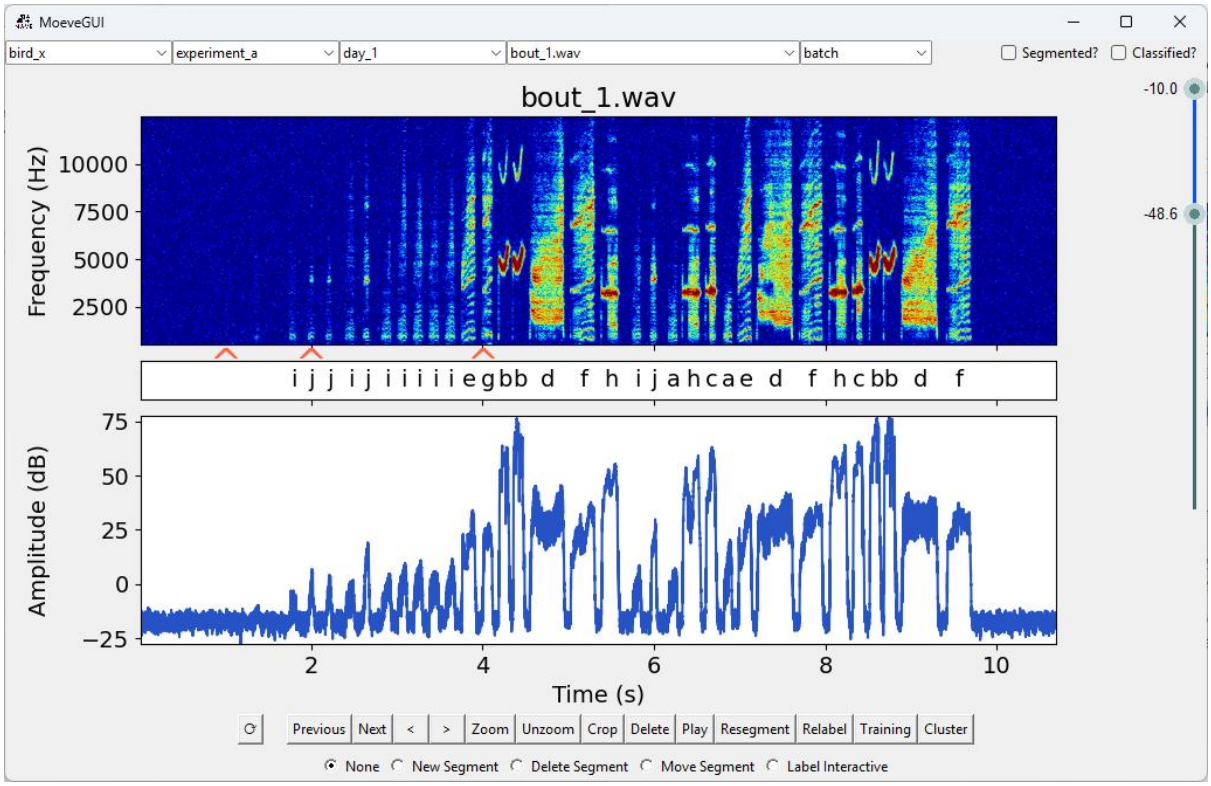


Figure 35: Display of feedback information

5. Loading previously recorded data

In case you previously recorded song data with EvTAF (Tumer & Brainard, 2007), or with any other program producing either *.wav* or *.cbin* files, they can be loaded into the MooveGUI as well. The only requirement is a respective *.rec* and *.not.mat* file for each song file.

Furthermore, the .rec file needs to be slightly modified to be processed correctly. The lines including the parameters **Hand Segmented** and **Hand Classified** (see 4. Feedback) need to be added. This can be done easily with a self-written python script, inserting the two lines at the specific positions. If you are unsure about how to do this, feel free to contact us.

Eventually, if not done so before, move the data folder into your *rec_data* folder in .moove (pay attention to the correct folder structure, see 2. MooveTaf) and start the MooveGUI. Note that in order to show feedback and catch trials correctly, the info lines in the *.rec* files must be formatted identically to the ones created by MooveTaf (see 4. Feedback).

As your old files will presumably contain different dB-values, you might want to adjust the slider values to correctly display the spectrogram (see 3. Setting the config).

6. FAQ

6.1. What if my segmentation looks ugly?

6.1.1. Case: Repeats

Offline segmentation

Segmenting repeats offline can be challenging and requires adjustment of the parameters for training the segmentation network. In figure 38, two special example cases of repeats are depicted. Case 1 shows repetition of a very short syllable (approx. 20ms long) with 'normal' sized gaps in between, and case 2 a repeat syllable with rather normal duration but very short gaps in between. Both cases will not be captured when using the default parameters of the segmentation network, as case 1 will not be detected at all and case 2 will be merged into one long syllable. This is due to the architecture of the segmentation network (for more details take a look at our paper). In short, the sliding window algorithm takes a bunch of audio chunks (default: 64) and checks how many times a syllable is detected within these chunks (default: 3 out of 5 times). When this is fulfilled, an onset is detected. Offset detection works similarly (default: 4 out of 5 times no syllable detected). With a sampling frequency of 44100Hz, one bin is only 1.45ms long. However, onset and offset detection can still be difficult. Therefore, if you see your network not catching up on repeats in the correct way you can adjust the parameters within the *Training* and the *Resegmentation* window.

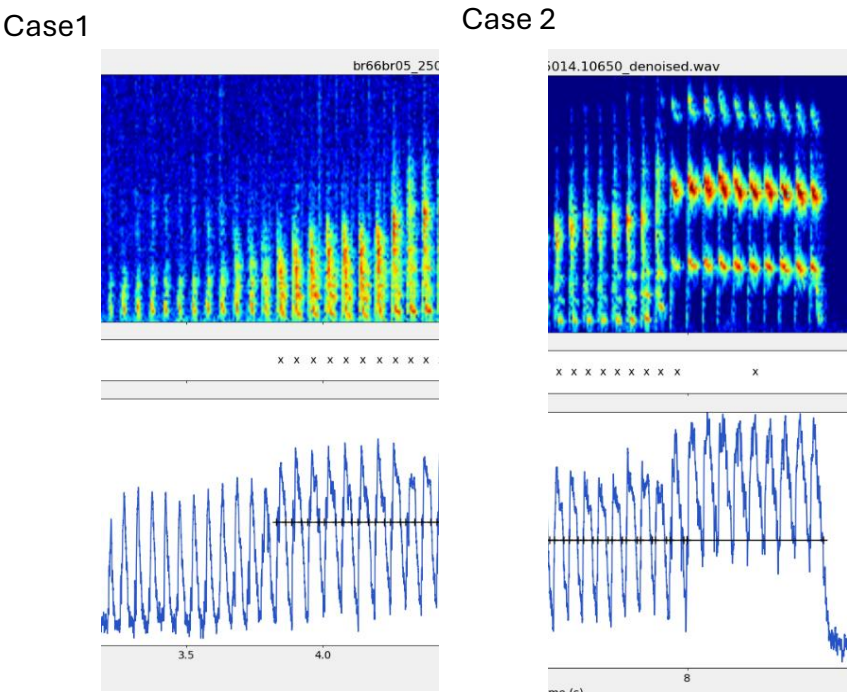


Figure 36: Example cases for repeat segmentation.

For that, you first have to hand-segment the repeats in the correct manner, create a dataset and train a network using the *training* window (3.3 Train the segmentation network) on '**Segmented files only**' (Fig 39, red box). Ensure that downsampling is **NOT activated** to capture every single

repeat syllable. Adjusting the parameter Batch Size would allow you to change the number of audio chunks, but decreasing could easily lead to incorrect segmentation.

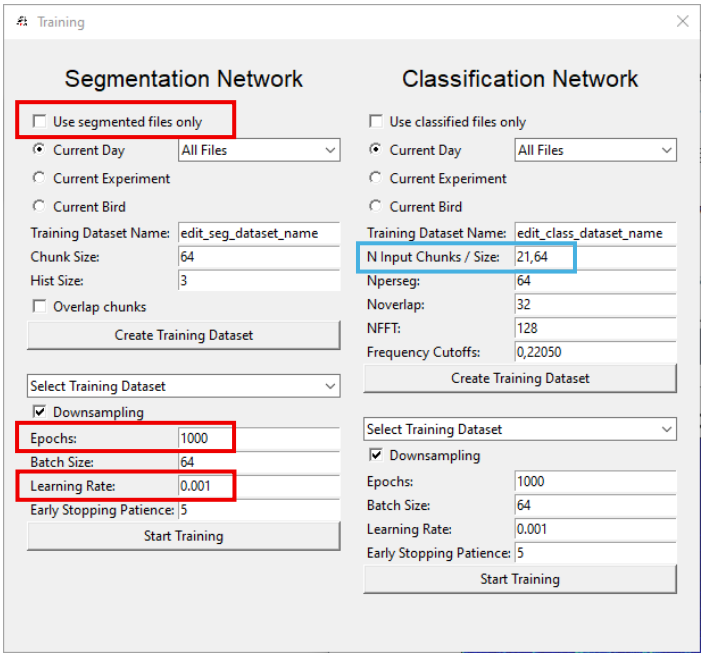


Figure 39: Training window

Once the network is trained, you can adjust the parameters for onset and offset detection parameters as described above when resegmenting your files (3.3 Resegment using the trained network). Also, adjusting the *min syllable length* and *min silent duration* can increase segmentation accuracy. Always make sure to **NOT tick the box saying 'Overwrite already segmented files'** to preserve your hand-corrected data.

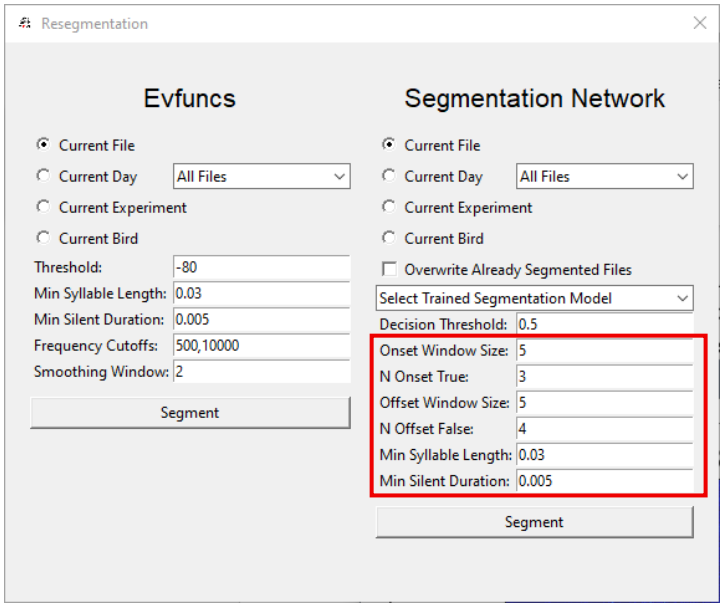


Figure 40: Resegmentation window

Once all parameters are adjusted, the repeat syllables should be segmented correctly. In this example case, parameters were set as follows.

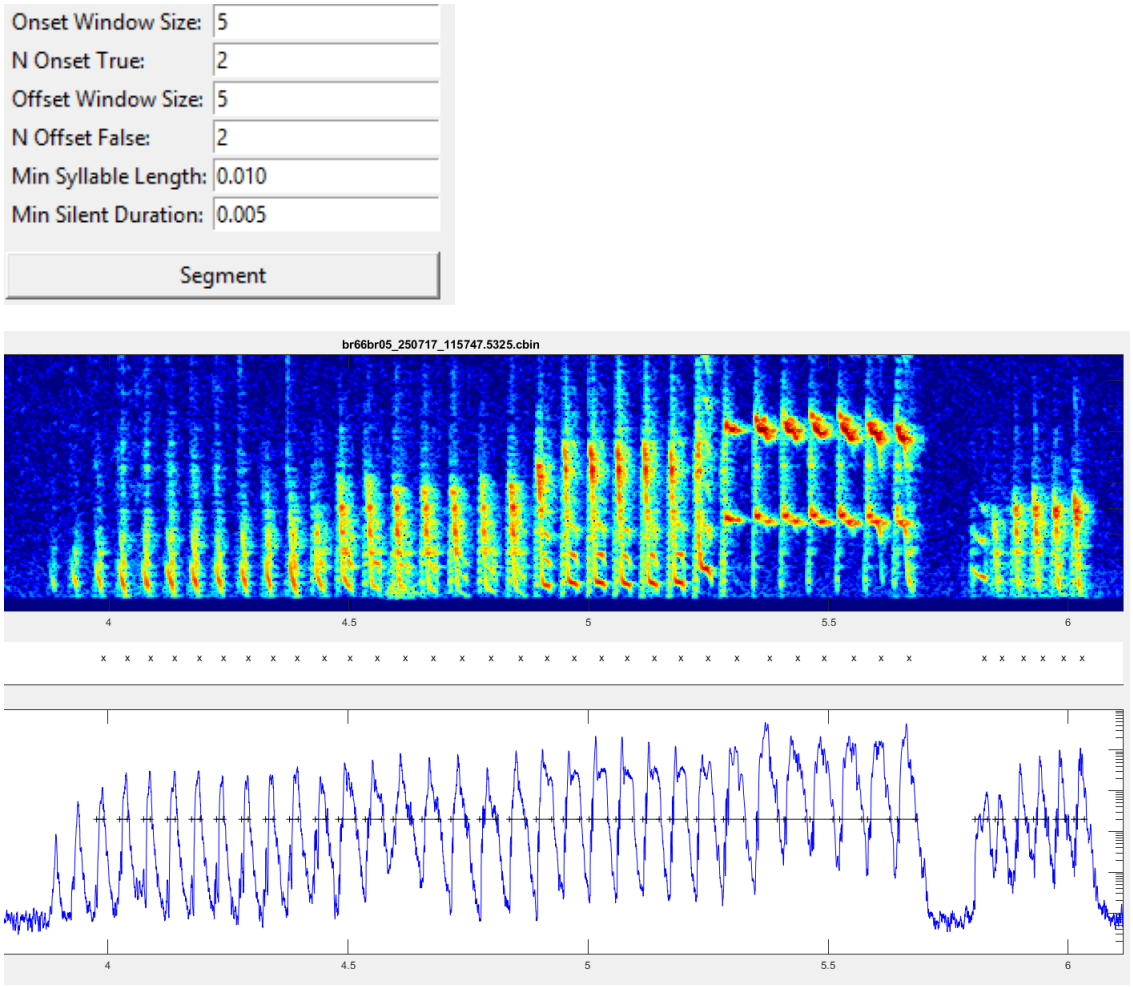


Figure 41: Example case for repeat segmentation.

Online classification

Online classification of syllables is per default performed within 30ms after syllable onset. In case you want to perform online classification on such very short syllables, you can reduce this time window. For that, you can change the size of audio chunks processed at once, as well as the number of input chunks (Fig 39, blue box). Per default, audio chunks have a size of 64 and 21 input chunks are processed. With a sampling frequency of 44100Hz and therefore a chunk duration of 1.45ms, the online classification for one syllable takes 1.45ms x 21 = 30.45ms. Adjusting these parameters can decrease this time but might reduce accuracy. Onsets and offsets of syllables shorter than 30ms will still be detected correctly with default parameters.

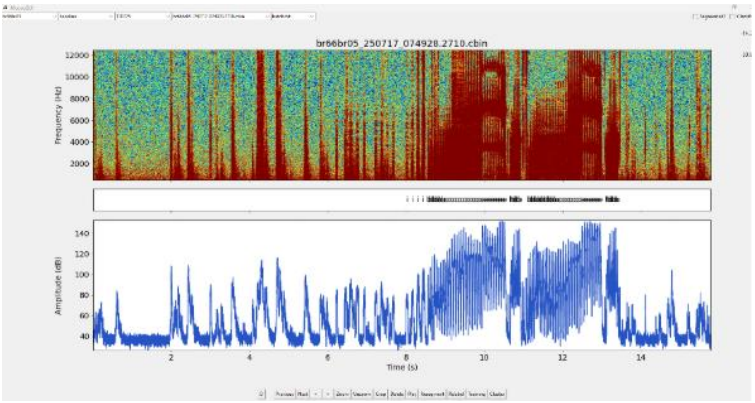
6.2. I can't find my .moove folder, where is it?

Per default, your .moove folder will be saved directly in your user folder. However, folders with names starting with a '.' are per default hidden under Linux and MacOS. On MacOS, they can be made visible pressing CMD + shift + . while in your user folder, on Linux it can be done using Ctrl + H. The folder location can be moved as described in *2. MooveTAF*.

6.3. I want to use my trained models on a Linux computer, but they are in a .zip folder?

On Linux, when copying the models, they will appear as a .zip folder. However, you can simply extract the files, which will leave you with a folder you can ignore and the desired *model.pth* files. The path to these two files can then be put into the config for online classification.

6.4. Why does my data look like this?



If you can't recognize your song nicely in the GUI, or you don't see anything in the spectrogram at all, you most likely have to adjust your sliders, as the dB-values will be incompatible with the default values. This can be done in the config-file (see 3. Setting the config). Especially cbin-files will need some adjustments, we had the best results with sliders between +40 and +100. If you can't find any values that work for your data, feel free to contact us.

7. References

Bencina, R., & Burk, P. (2001). PortAudio-an open source cross platform audio API. ICMC.

Geier, M. et al. Portaudio-binaries. GitHub. https://github.com/spatialaudio/portaudio-binaries

Geier, M. et al., (2020). Sounddevice. GitHub. https://python-sounddevice.readthedocs.io/en/0.5.0/

Nicholson, D. (2021). *Evfuncs* (Version 0.3.2.post1) [Computer software]. https://doi.org/10.5281/zenodo.4584209

Tumer, E., & Brainard, M. (2007). Performance variability enables adaptive plasticity of 'crystallized' adult birdsong. *Nature*, *450*, 1240–1244. https://doi.org/10.1038/nature06390