Deep learning in PAMGuard

Last tested on PAMGuard 2-02-15a (dev version pre-16-release) 13-Feb-2025

Table of Contents

Introduction

The continued increase in the storage capacity and battery life of acoustic recorders is allowing industry and academia to collect passive acoustic monitoring (PAM) data over ever larger spatial and temporal scales. With this wealth of new data comes a problem - what do we do with all the hugely complex data that is collected by acoustic recorders? How do we accurately extract the already variable vocalisations of our target species within the dynamic and highly variable soundscapes? In the past, manual or semi-manual validation of data might have been an option (humans are still the best at pattern recognition) but with such large quantities of data this is no longer feasible. We need accurate automated algorithms, but over the past decades developing such algorithms which are applicable to a wide range of environments, species and soundscape contexts has proven to be extremely difficult...until recently.

Supervised machine learning is when you feed an algorithm training data (i.e. data that are labelled with the correct answer to whatever we are trying to solve) and it automatically constructs an appropriate classifier model that can then be used to analyse new, unlabelled data. **Deep learning** is a subset of machine learning which uses artificial neural networks to train classifiers. Deep learning is a popular buzzword today, and it has real benefits for acoustics compared to previous machine learning methods. Deep learning is scalable, i.e. it increases in accuracy with the more training data you feed it. You can also "top up" classifiers with new training data if required. Another important aspect of deep learning algorithms is that they automatically extract features from data. A previous machine algorithm might have required a list of features, such as peak frequency, length, amplitude, etc., but a deep learning algorithm can ingest raw spectrogram images or even waveforms and work out the best features to extract for classification itself (in reality though this is a little more complicated). The final, and perhaps most important, aspect is that deep learning methods are now used everywhere - in advertising, in your photo apps, in the military, by NASA, etc., which has meant that a massive ecosystem of code and services have been developed around using deep learning technologies. This means we have free access to state-of-the-art deep learning tech (from Microsoft/Google, etc.) which probably has had more R&D spending in the last few years than the sum total of all funding in bioacoustics research, ever. What a time to learn about these techniques!

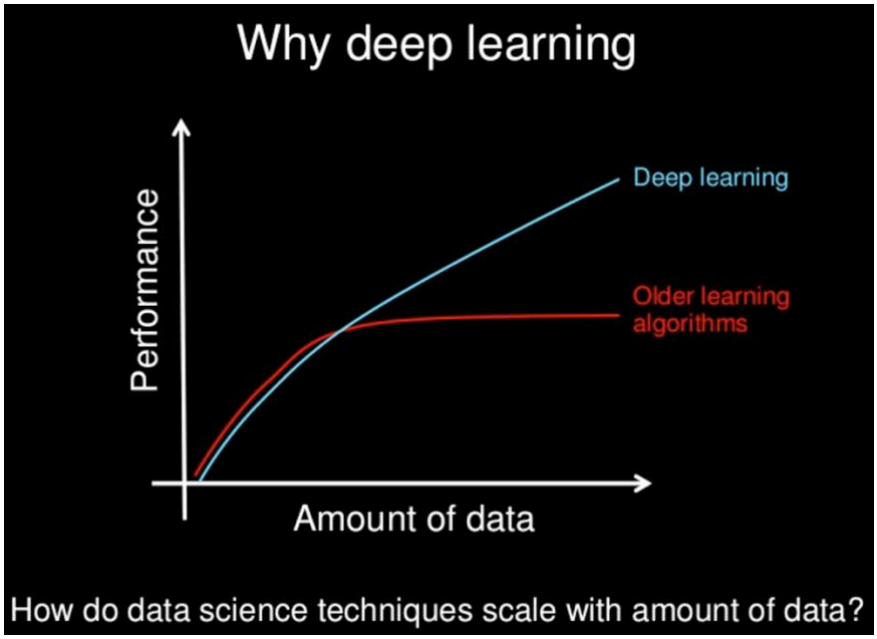


Figure 1. A graphic from a slide by Andrew Ng. This shows the advantage of a scalable deep learning approach compared to older machine learning methods.

In the context of practical application to PAM, deep learning allows us to train highly accurate classifiers which can cope with a large variation in temporal and spectral properties of signals within complex soundscapes. In fact, the approach is so good at acoustic analysis that it's solving age-old analysis problems that acoustic researchers have been tackling (with little progress) for decades; one example is automatically detecting right whale calls (Shiu et al. 2020). However, whilst deep learning has the potential to provide extremely powerful algorithms there are caveats. Running deep learning classifiers can be very processor intensive compared to more simplistic detection and classification, they require large quantities of training data and, like all automated algorithms, they can still be vulnerable to unexpected inconsistencies in data for which they have not been trained. In addition, despite the numerous research papers, huge code ecosystem, and hype around this undoubtedly effective approach to automated acoustic analysis, training and then running deep learning classifiers is still not straightforward or accessible, usually requiring coding in Python. These technical barriers restrict the uptake of deep learning to specialised research groups and thus its impact so far in marine acoustics has been limited.

PAMGuard is one of the few (perhaps only) pieces of open-source bioacoustics software which integrates generic deep learning methods. In the next exercise, we will use the ***Raw Deep Learning*** module to automatically detect both right whale upsweeps and the calls of different bat species.

Detecting right whales in noisy environments

In this example we will detect right whales in noisy acoustic recordings. Low frequency right whale upsweeps are notoriously difficult to accurately detect, especially given their spectral overlap with low frequency shipping noise, and when confounded by more numerous humpback whale vocalizations.

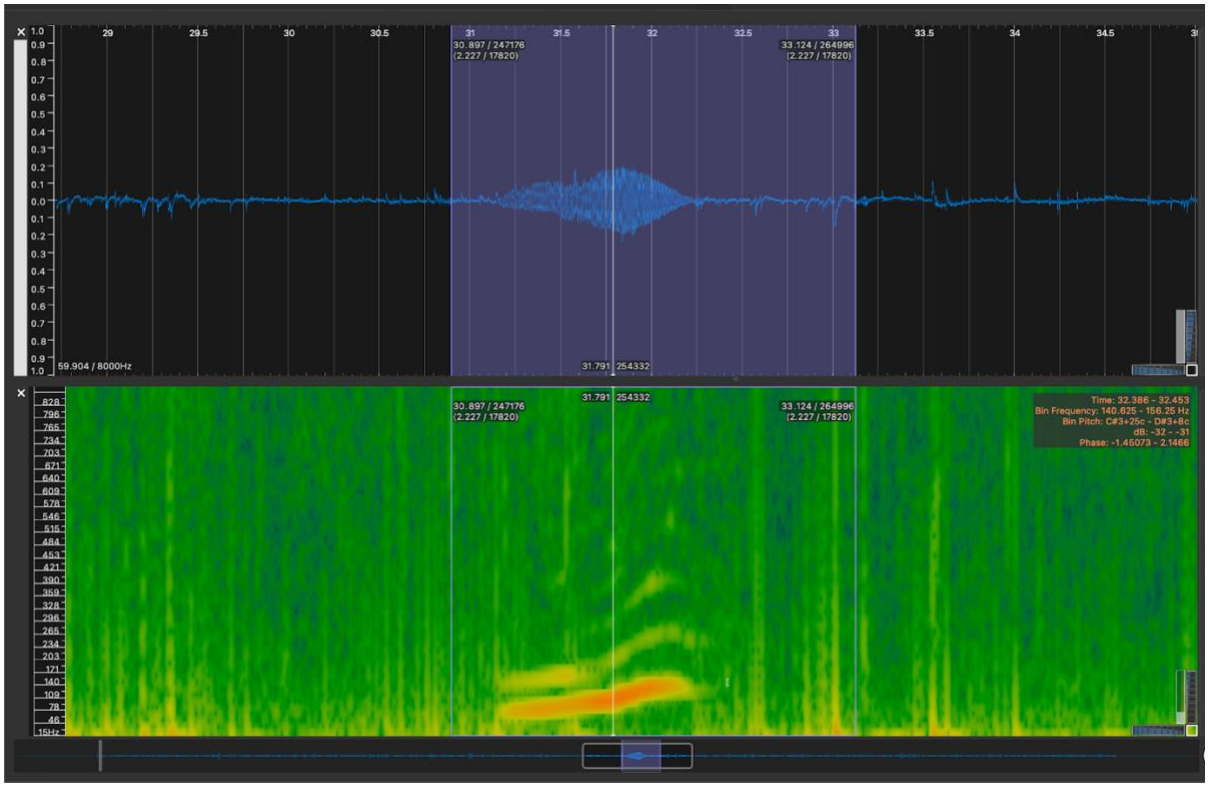


Figure 2. An example of a right whale upsweep generated in Sonic Visualiser.

A research group led by Professor Marie Roch at San Diego State University released a landmark paper on detecting right whale calls using deep learning….

Shiu, Y., Palmer, K.J., Roch, M.A., Fleishman, E., Liu, X., Nosal, E.-M., Helble, T., Cholewiak, D., Gillespie, D., Klinck, H., 2020. Deep neural networks for automated detection of marine mammal species. Scientific Reports 10, 607. https://doi.org/10.1038/s41598-020-57549-y

One of the unique aspects of this paper was that Shiu/Palmer et al. (2020) used a dataset that many other bioacoustics researchers had also been using as standardised test data to quantify the performance of automated algorithms for right whale upsweeps. Leading bioacoustics research groups in the US and the UK had been attempting to create accurate right whale upsweep classifiers for decades and tested their effort at the DCLDE conference in 2013 (Detection, Classification, Localisation, and Density Estimation). Shiu/Palmer et al.'s deep learning approach outperformed all those efforts by a very wide margin (see Figure 1 in their paper) and so they had, for the first time, proven that an accurate and automated right whale upsweep classifier was possible.

In this exercise we are going to use the Shiu/Palmer et al. (2020) classifier to detect some right whale calls within PAMGuard.

Importing and visualising data

To begin, open a blank PAMGuard configuration file (start PAMGuard and create a new filename when prompted). You will be greeted by the usual PAMGuard blank canvas – we are going to add the modules required to create an acoustic workflow which analyses sections of raw data using the deep learning module and save chunks of sound data where the prediction (the probability that a detected sound is a right whale) is above threshold of 0.8.

We will be analysing acoustic data and so the first module we need is Sound Acquisition to import the wav files. This module can also be hooked up to a DAQ (data acquisition) device if we wanted to implement this acoustic workflow in real time. Add the Sound Acquisition module by selecting ***File->Add modules -> Sound Processing -> Sound Acquisition.*** Now open the settings dialog by clicking ***Settings -> Sound Acquisition…***. Select ***Audiofile folder or multiple files*** from the ***Data Source Type*** selection menu. Click on ***Select Folder or Files*** and browse to the folder that contains wav files in the right whale tutorial data folder *(../right_whale_tutorial/wav/* The sample rate should show up as 2000Hz with 1 channel. Everything is now set up, so click ***Ok***.

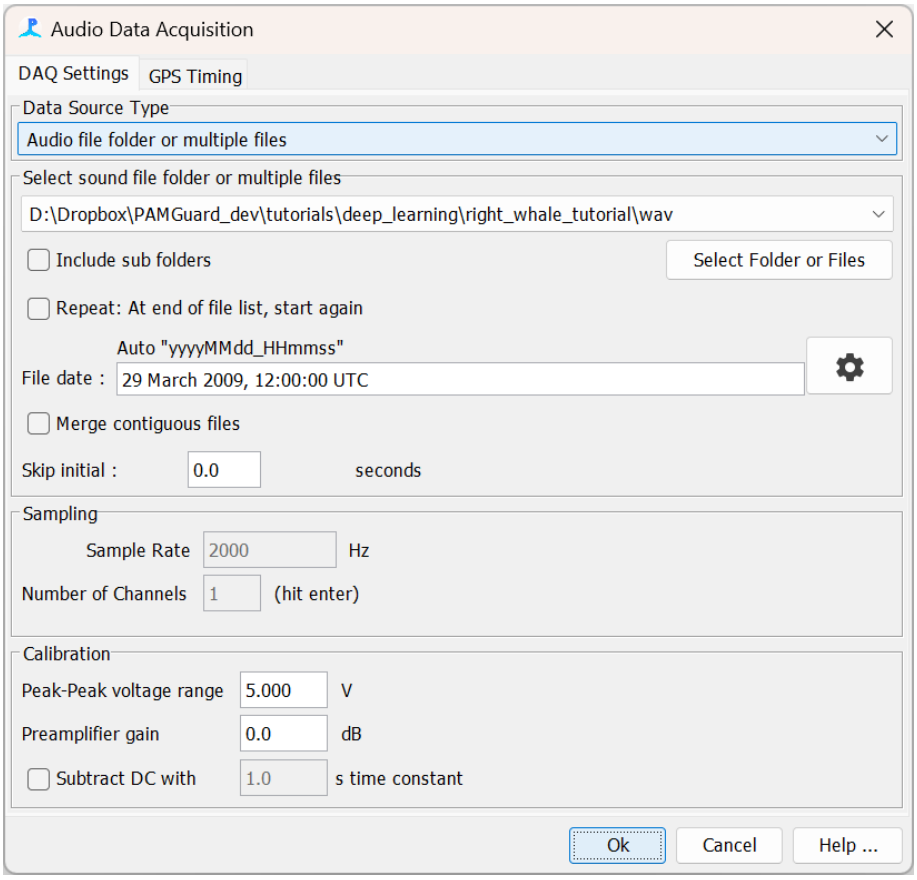


Figure 3. The Sound Acquisition module after it has been set up correctly.

The raw deep learning classifier only requires raw sound data as an input, however, we will want to visualise results on a spectrogram so will add the *FFT module*. Add the FFT module by selecting ***File->Add modules -> Sound Processing -> FFT (Spectrogram) Engine.*** Next open the settings dialog for FFT by selecting ***Settings->FFT (Spectrogram) Engine settings…***

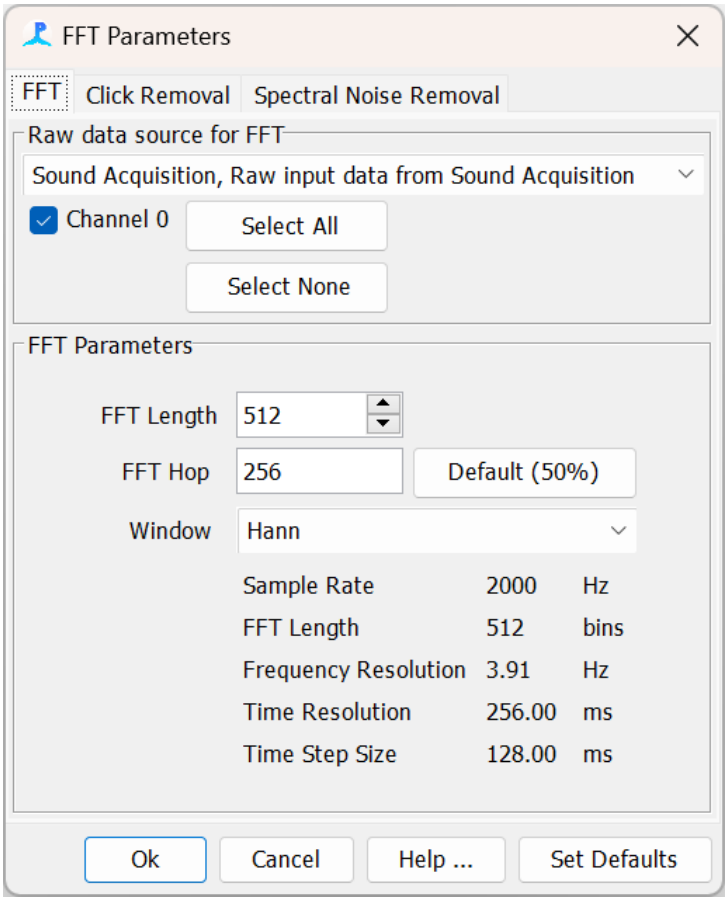


Figure 4. The FFT (Spectrogram module) settings dialog after it has been set up correctly.

The FFT settings dialog allows you to change the usual properties associated with a spectrogram e.g., FFT length, Hop size and window. The best settings for visualising right whale calls are a 512 ***FFT length*** and 128 or 256 ***FFT Hop***. Change these settings and click **Ok**. Note that the FFT settings has no bearing whatsoever on the deep learning module as it's input is raw sound data – thus whatever time resolution you select for FFT Length and FFT hop only has a bearing on how we visualise the data i.e. how a spectrogram looks.

So, next we will need a display to show the spectrogram. First of all, we need a User Display tab. Select ***File->Add modules -> Displays -> User Display.*** This will create a display window tab. Click on the tab and in the top menu select ***User Display -> New Time base data display fx.***

The Time Base Display allows users to visualise and overlay any data stream in PAMGuard with a time dimension e.g. spectrograms, click detections, whistle detections, deep learning detections, etc. It also allows users to change the y-axis so that data can be visualised in a variety of ways. For example, click detections can be viewed as FFT spectra with a frequency axis or scatter points on an amplitude bearing or bearing axis. Here we will be viewing spectrogram data (which only has one y-axis: frequency) but we need to add it the Time Base Display first. Click the ***+Add*** button and select ***FFT (Spectrogram) Engine***. The display will now have a frequency y-axis ready to display spectrogram data.

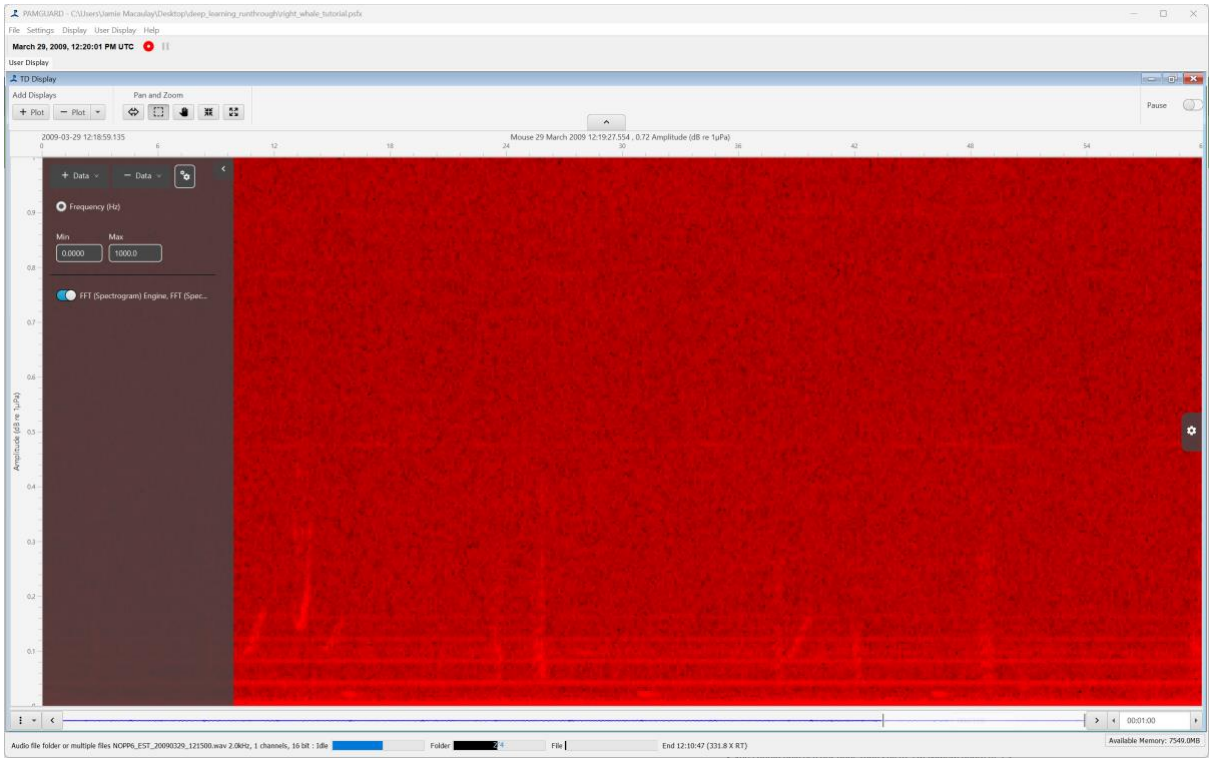


Figure 5 The Time Base display with a spectrogram added (the display will be black until you start streaming data).

The last module in this section is Sound Output module. We need to add this because, without it, PAMGuard will process data as fast as the computer processor allows and thus visualising data occurs too quickly and becomes difficult. Add the Sound Output module by selected ***File->Add modules -> Sound Processing -> Sound Output.*** Next open the settings dialog by selecting ***Settings->Sound Output…***

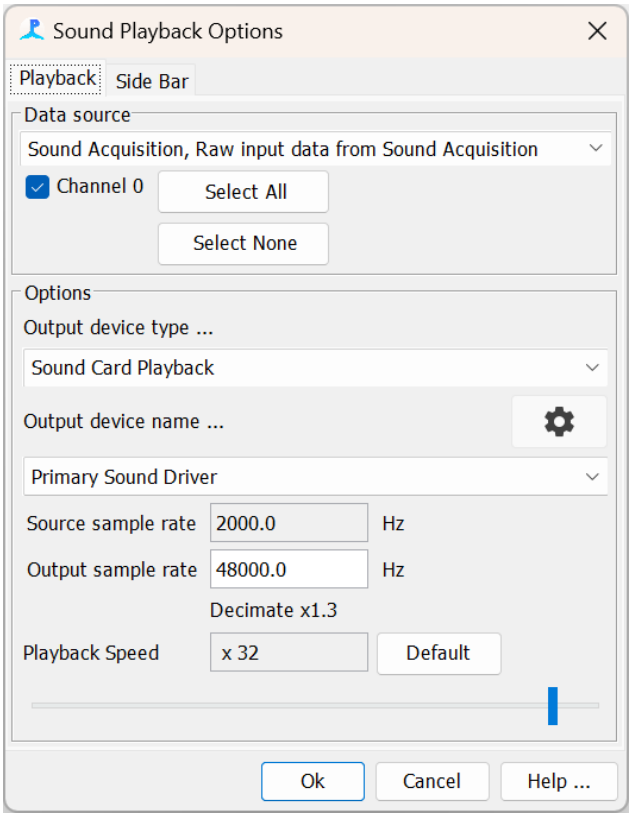


Figure 6. The sound output module.

The Sound Output module allows users to select the sound card in which sound will be played out. The default settings should be fine (but remember to check ***Channel 0*** if it is not already) but check that the ***Output device name…*** looks sensible. If you are using Bluetooth headphones for example you may wish to select them in this menu. Click ***Ok*** once you are finished.

We are now ready to test this part of the data model. First make sure you SAVE YOUR CONFIGURATION (***File->Save configuration***). Click the red button and you should see spectrogram data streaming. Play around with the playback speeds in the left side bar and the spectrogram colours in the Time Base Display.

> ***PAMGUARD QUIRK NO. 1***
> *If you want to reset PAMGuard back to the first wav file in the folder after running, open the Sound Acquisition module and click **Ok**. This is not ideal or intuitive but currently how things work.*

Adding the deep learning module

Now we will add the deep learning module and import the right whale deep learning classifier. Note that we often refer to the trained deep learning classifier as a "model" which produces "predictions" of a different "classes". This translates (in this context) to a deep learning classifier calculating the probability that a segment of acoustic data holds a target sound (e.g. right whale upsweep).

Add the deep learning module by selecting ***File->Add Modules->Classifiers -> Deep Learning Classifier.*** Next open the settings dialog by selecting ***Settings -> Deep Learning Classifier…***

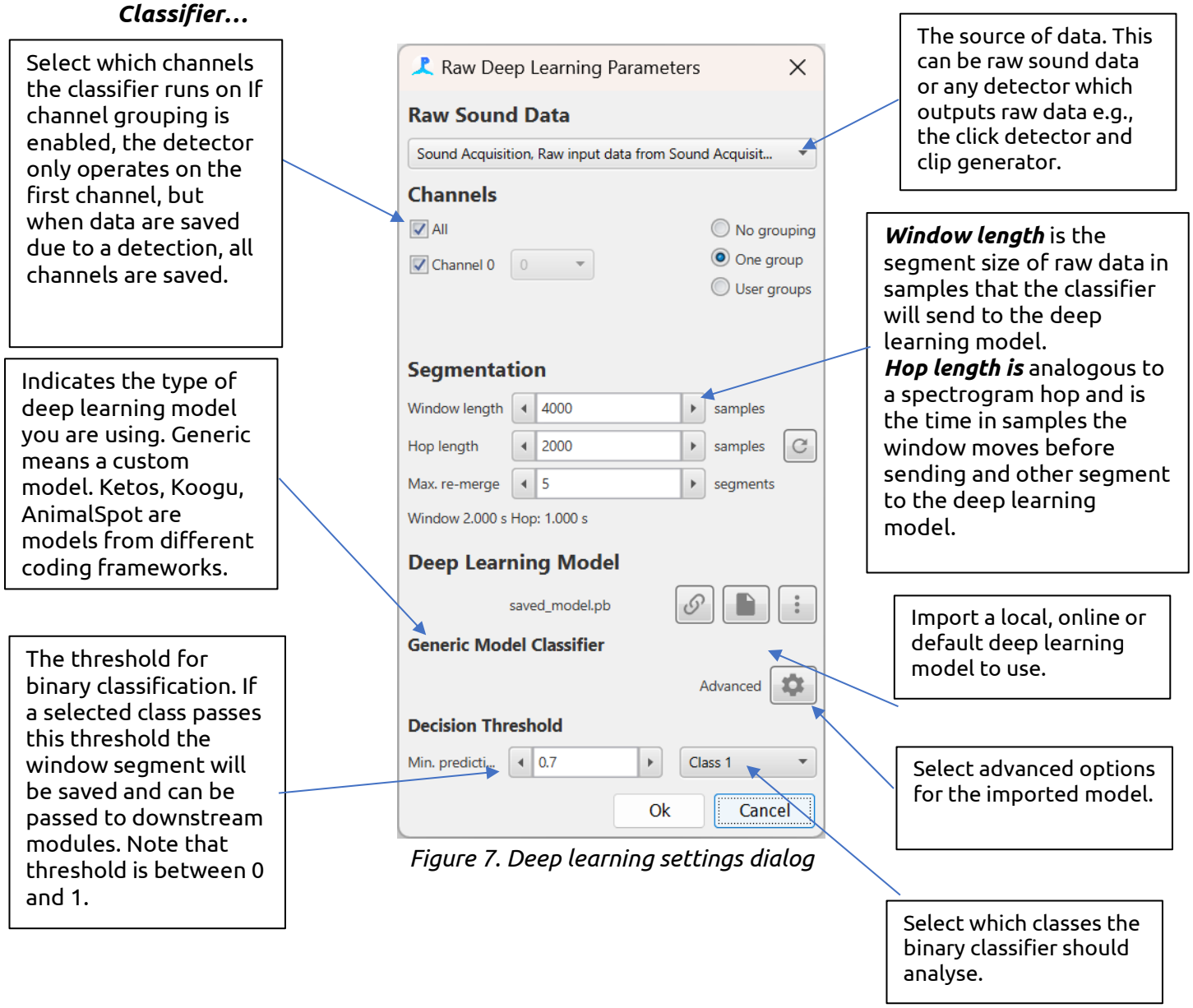


Figure 7. Deep learning settings dialog

The settings dialog is described in Figure 7 (note many of the descriptions will make more sense as you work through the tutorial). The raw deep learning classifier accepts continuous raw sound data as and input. It divides the data into overlapping segments based on a ***Window length*** and ***Hop length***. Each segment is passed to a selected deep learning model and classified as a detection if the prediction of user selected class(es) passes a threshold value. If a segment of sound passes binary classification, it is saved to a new data stream and passed to downstream processes. This allows the module to be inserted into almost any data model created in PAMGuard. For example, the module could be used in real time to localise right whales from a beam forming array (downstream module would be a beam former) or in post processing to generate clips of detected calls (downstream module would be the clip generator).

The raw deep learning module also accepts detections from other detectors as an input source. This works in almost exactly the same way as the continuous acoustic data except

the raw sound data from each detection is analysed and, instead of saving detections into another data stream if they pass a binary classification, the detections are tagged with additional metadata (e.g. prediction values) and then passed to downstream processes.

> ***"FUN" FACT NO. 1***
> Notice that the style of the Deep Learning module and Time Base Display is a little different to other modules. This is because it's built with a newer user interface (UI) technology called JavaFX. This display technology is much more flexible and can be run on many different operating systems, e.g., iPad OS, Android and even in a web browser.

Deep learning frameworks

In the introduction it was mentioned that deep learning algorithms automatically figure out what features to use from raw data – something called **feature learning**. In reality it is much easier to train an algorithm on data where some basic data cleaning and handling has been performed and so that the inputs into acoustic deep learning models are usually some sort of noise reduced spectrogram image with frequency limits in the range of the target vocalisation (i.e. we would not input a 0- 250kHz spectrogram image for right whales).

This means that each deep learning model expects a specific type of input, i.e. usually a spectrogram image a certain number of pixels between specific frequency limits and often enhanced in some way (e.g. clicks/spots removed, normalised, etc.). For PAMGuard to run models it also needs to know what these data transforms are, so it supplies the correct input.

This is where the idea of deep learning frameworks comes in. A framework is essentially a metadata structure integrated with a deep learning model that allows PAMGuard to set up the transforms required by the deep learning model. These frameworks are often based around a deep learning library (usually a bunch of Python code) which means that any model trained using the library can be imported seamlessly into PAMGuard.

There are currently 3 frameworks available in PAMGuard for deep learning.

Generic Model

The generic model framework allows users to import the most common types of deep learning model e.g., TensorFlow, Pytorch. The user has to manually define the types of transforms and expected data outputs required by the model, although these can also be imported from and exported to an external .pgtf file.

The generic model is the most complex and error prone way to set up a classifier and not recommended if possible but it does mean that we can load classifiers which have not used a standardised acoustic deep learning library to train their models (such as the right whale classifier we will be testing).

AnimalSpot

Animal spot is a species independent Python library for training and deploying deep learning classifiers. AnimalSpot is based on Pytorch which allows metadata to be stored in the model file. This means users can import one file and PAMGuard will set up all required transforms automatically.

Details on AnimalSpot can be found in

Bergler, C., Schröter, H., Cheng, R.X., Barth, V., Weber, M., Nöth, E., Hofer, H., Maier, A., 2019. ORCA-SPOT: An Automatic Killer Whale Sound Detection Toolkit Using Deep Learning. Sci Rep 9, 10997. https://doi.org/10.1038/s41598-019-47335-w

The code for training AnimalSpot models is available at https://github.com/ChristianBergler/ANIMAL-SPOT

Note: Animal Spot is no longer supported. Old models are backwards compatible but newer models will not work. Animals Spot models can be made to work as a Generic Model if needed. We are working to resolve this with the AnimalSpot developers.

Ketos

Ketos is a deep learning framework developed by Meridan. It is based on Google's Tensorflow Python library and has comprehensive documentation. If you want to learn all about deep learning in acoustics the Ketos website is the place to go.

The Ketos framework website is https://meridian.cs.dal.ca/2015/04/12/ketos/

Note: Ketos is no longer supported. Old models are backwards compatible but newer models might not work. Ketos models can be made to work as a Generic Model if needed. We are working to resolve this with the Ketos developers.

Koogu

Koogu is a deep learning framework developed by Shyam Madhusudhana. It is based on Google's Tensorflow Python library and has been used to develop blue whale and right whale detectors e.g.

Miller, B.S., Madhusudhana, S., Aulich, M.G. and Kelly, N. (2023), Deep learning algorithm outperforms experienced human observer at detection of blue whale D-calls: a double-observer analysis. Remote Sens Ecol Conserv, 9: 104-116. https://doi.org/10.1002/rse2.297

The Koogu framework website is https://shyamblast.github.io/Koogu/en/stable/

PAMGuard zip models

PAMGuard zip models are very similar to generic models except the model is saved within a zip file alongside a metadata file that sets everything up PAMGuard automatically This is a useful tool for users who have used the generic model tools correctly allowing them to then package everything together to share deep learning models in a format that is much easier for other users to load.

Importing and running the right whale deep learning model

In this example we will be using the PAMGuard zip model because the right whale classifier used by Shiu et al. 2020 was not developed using AnimalSpot, Koogu or Ketos. Press the import classifier model button  and browse to the *right_whale_model.zip* in the right whale tutorial folder. It may take up to a minute to import the model (a load icon will show). If the load has been successful then saved_model.pb will appear next to the import button.

Now that we have imported the model, we need to check the transforms. The transforms have automatically been set up because the zip file contains a small JSON file with PAMGuard metadata for this model. Click the Advanced settings cog to open the transforms pane. Select Southern Right Whale in the ***Example Sound*** menu of the "Model Transforms" tab to see a preview of the input image for the classifier as shown in Figure 8.

IF YOU ARE CURIOUS
In the advanced transforms pane, you can manually add transforms in a list and set the correct parameters. You can also import a preset settings file by selecting the ***Import…*** *button (for example right_whale_DL_settings.pgtf file). Settings can be exported simply by pressing the* ***Export…*** *button. If you zip your model up with these a working. pgtf transform settings file then the model can be opened easily by any PAMGuard user.*

Next, we must make sure we have the correct number of output classes. In the Decision Threshold section make sure that only Right Whales are ticked – we don't want our classifier to classify noise!

Now we need to ensure we are using the correct window length. In this case the model will only work with a segment length of 4000 (2s). We can set the hop size to any value – 2000 is a reasonable value for this example.) Note that enabling the ***Use default segment length*** switch will also set the correct length of window for the sample rate.

Keep the ***Max re-merge*** as 5 segments as this means our maximum detection can be 5 segments long. The default threshold is 0.9, however, 0.8 works a little better for detecting calls – set the ***Min. prediction*** to 0.8. Make sure that "right whale" (and not noise) is selected for classification. Finally make sure ***Channel 0*** is selected and Click **Ok** to close the deep learning settings.

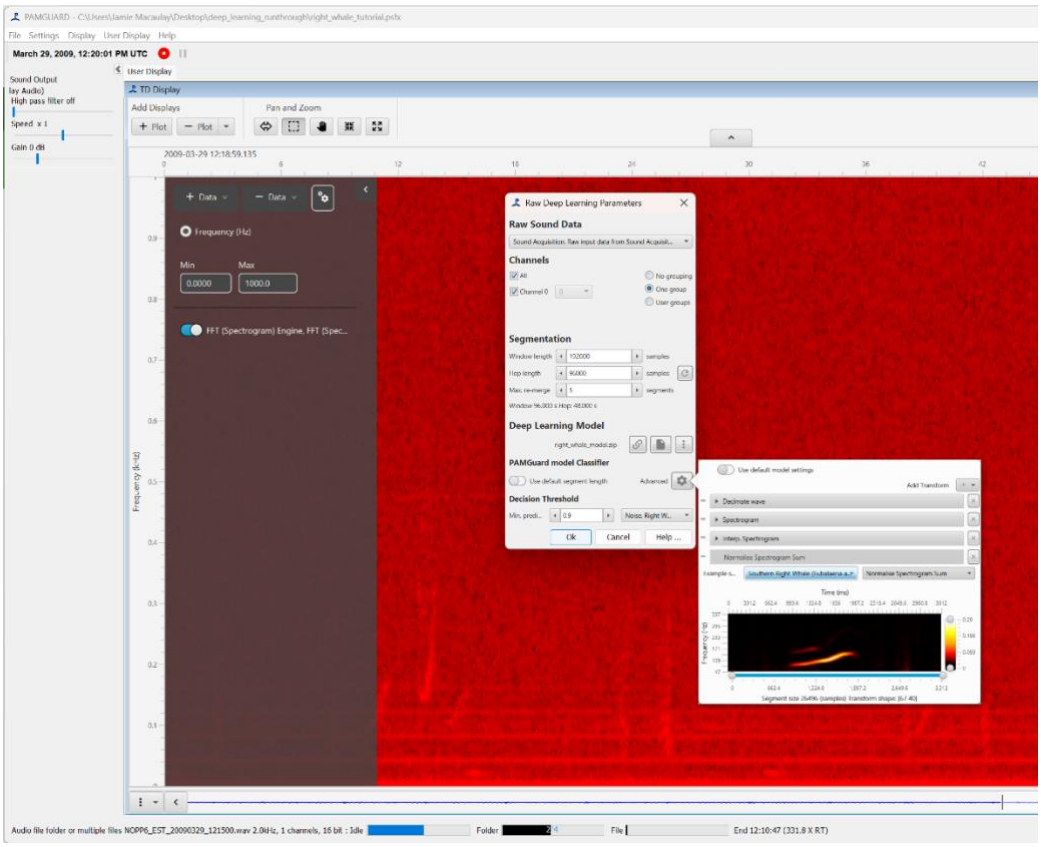


Figure 8. The raw deep learning module set up for right whale calls.

Other ways to import
There are two other ways to load this right whale model. First, it's now included as a default model in PAMGuard – clicks the ⋮ *icon and select right whale model and PAMGuard will download the model from GitHub and set it up automatically. Additionally, you could get the raw model saved in TensorFlow and select the saved_model.pb file. PAMGuard will load the model but would not know what kind of input data its need. The input transforms could then be set up in the Advanced transforms pane or imported from a .pdtf transforms setting file.*

Everything is now set up for data processing, however, we want to visualise our results so we need to add the deep learning data streams the Time Base Display. We will visualise detections which pass threshold overlaid on the spectrogram and the raw predictions for all segments in a separate plot.

Add a new plot by pressing the + button in the top Add Displays menu (note this can be hidden – press the down arrow in the middle of the top axis to show the menu). This will create a new plot underneath the spectrogram. In the new plot on the left settings pane, select the ***+ Add*** button and add the ***Prediction probability, Deep Learning Classifier*** data stream. Now go the top spectrogram plot and press ***+ Add*** button and select ***Deep learning detection, Deep Learning Classifier*** *(*Do not select **Deep learning group detections** as these are data for deep learning classifiers that use a group of detections as their input*)*

We are now ready to go. MAKE SURE YOU SAVE THE CONFIGURATION (***File->Save***) again, and then press the record button.

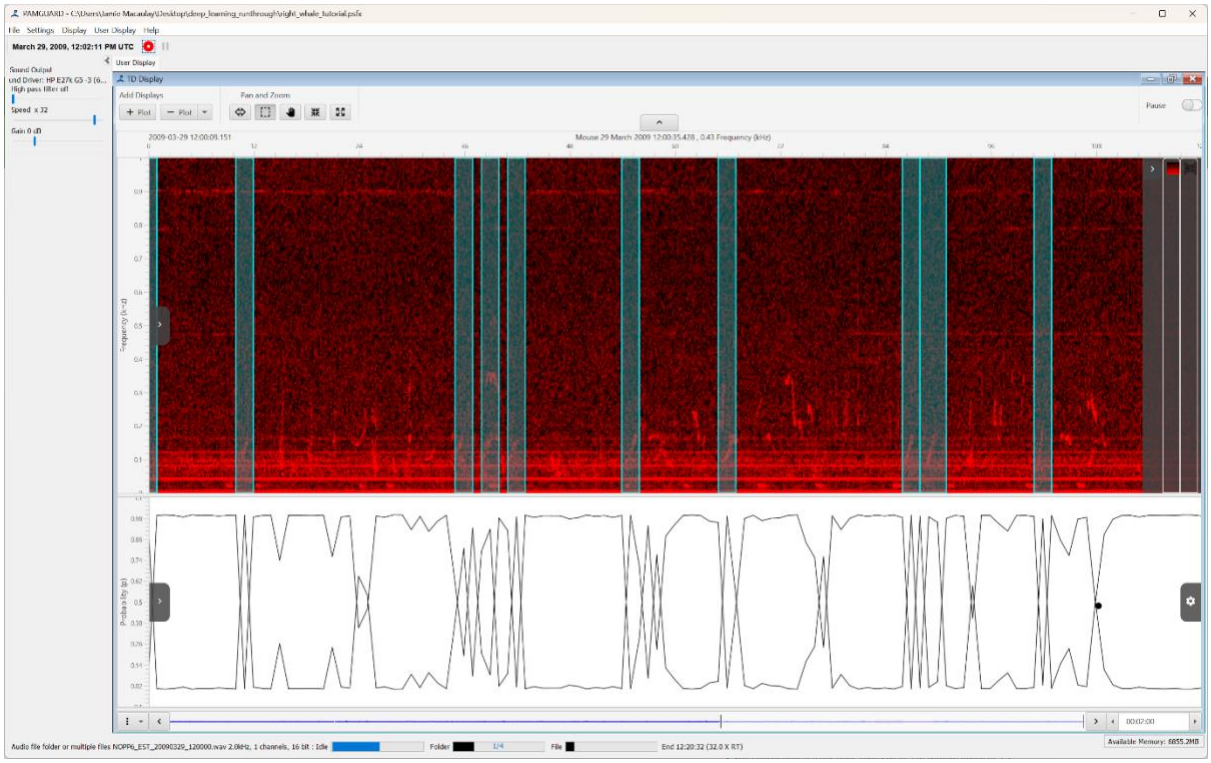


Figure 9. Running the deep learning classifier for right whale calls should look something like this.

Use the ***Speed*** slider on the left hiding bar to increase or decrease the analysis and playback speed and watch the prediction plots carefully. The two lines represent the prediction of noise and right whale. Normally the noise prediction is high, and the right whale prediction is low. However, when a right whale call is detected, these will change which is very obvious

to spot on the bottom display. If the right whale call passes the threshold prediction it is "detected" and will show as a translucent box on the spectrogram.

Saving and viewing data

Once you have explored running the deep learning model, it is time to start thinking about saving the data so that they can be viewed in PAMGuard Viewer mode and further analysed after processing.

PAMGuard saves data in three formats, raw sound files, binary files and a database. The binary files store detection data, such as the detected right whale calls and the database saves metadata and auxiliary data such as GPS, AIS, effort, etc.

> ***PAMGUARD QUIRK NO. 2***
> *"Binary files" is perhaps the worst possible name that one could give file format. All computer files are binary files – somehow this name made it out of development and stuck – now that everyone is using the binary file module it's too late to change….*

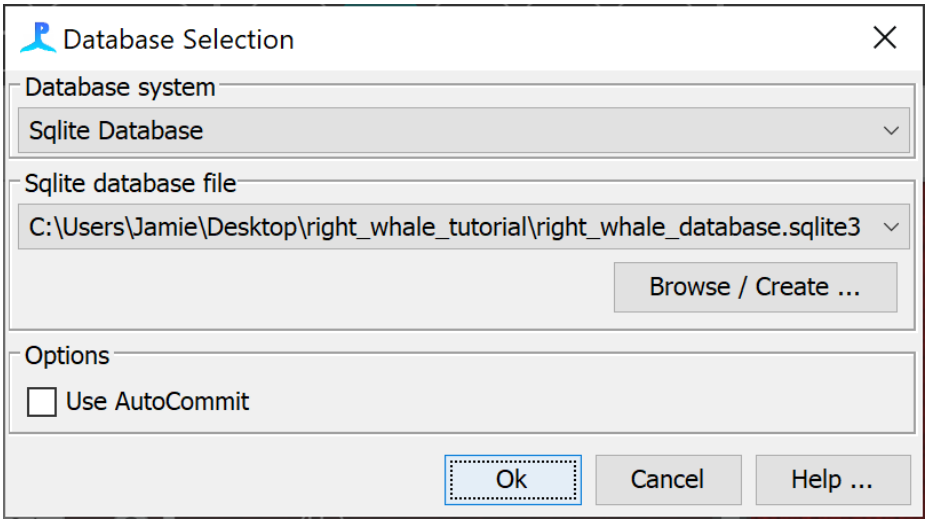
We are already analysing raw sound files so we don't need to save any, but we will want to save metadata (in a database) and the output from the deep learning detectors (in binary files). Enabling saving data is easy – simply add the relevant modules. Go to ***File -> Add Modules -> Utilities -> Binary Storage*** and then ***File -> Add Modules -> Utilities -> Database.*** Now we need to tell PAMGuard where to save the data. Go to ***File -> Binary Store*** -> **Storage options…** and press the ***Browse*** button. Use the folder selection dialog to select or create a new folder. Next go to ***File -> Database -> Database Selection…*** and in the settings dialog click ***Browse…*** and create a new SQLite database.

Now reset the raw sound files using the open Sound Acquisition and press **Ok** trick. Now click the record button again and wait for PAMGuard to run through all the wav files again.

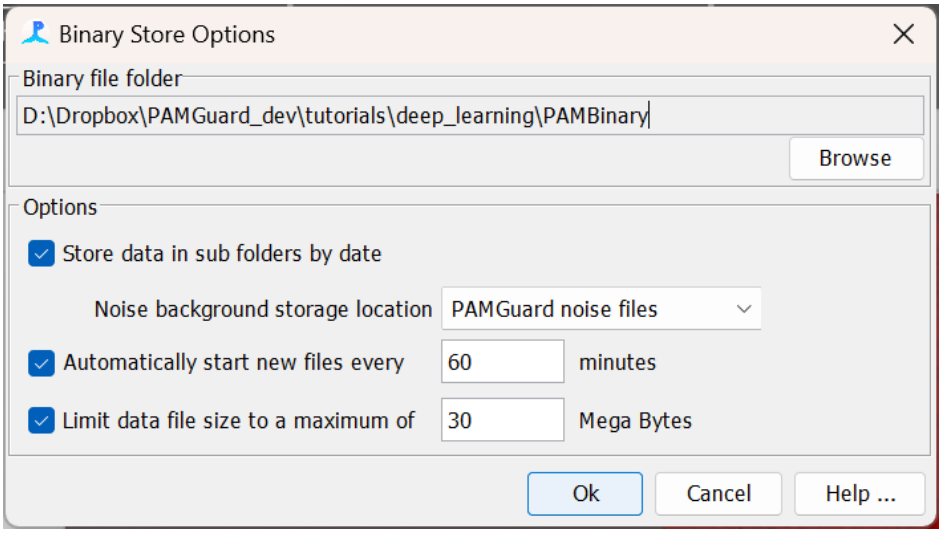
> ***"FUN" FACT NO. 2***
> PAMGuard started as a real time analysis program for marine mammal acoustic monitoring during exploration activity on seismic survey vessels. It was some years after the inception of PAMGuard that users were given the ability to view data analysed in the program in a different program called PAMGuard Viewer mode. Having two separate programs is not a user-friendly solution and there is now ongoing work to integrate PAMGaurd and PAMGuard viewer mode.

Opening the data in PAMGuard viewer mode

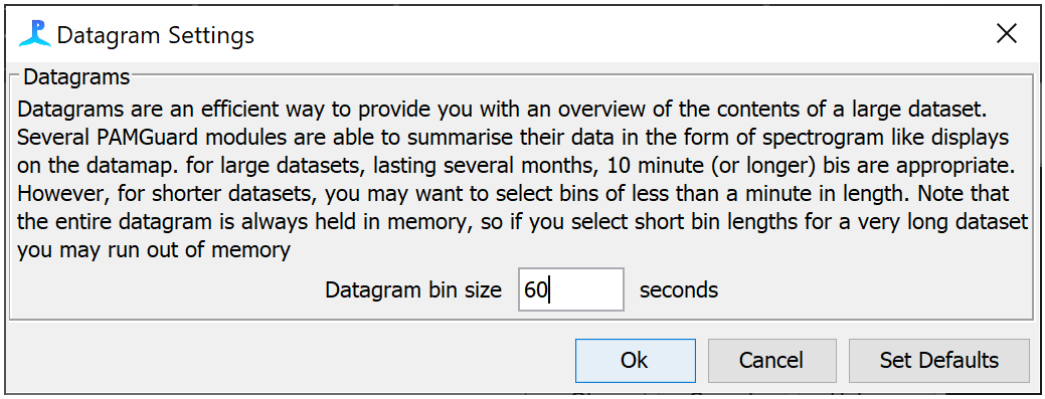
Once the data analysis is complete, we can view the results in PAMGuard viewer mode. Start PAMGuard Viewer mode and in the database selection dialog that pops up, click **Browse/Create …** and select to the database you created in the previous step.



After you have selected the correct database the binary storage dialog will pop up. The path to your binary storage has already been saved in the database so it should be correct in the ***Binary storage options*** field but double check. If it's incorrect click ***Browse*** and select the correct folder.



The first time a database is opened in viewer a datagram option will appear. This allows the user to set the time bin over which data is summarised so that long-term dataset can be visualised. Set the ***Datagram Bin Size*** to 60s.



PAMGuard viewer mode looks almost exactly the same as PAMGuard. The main difference is the **Data Map** tab. This shows the summary of detection data over the entire dataset. We have analysed a relatively small dataset here, but in reality, we could be churning through month or even years of data, in which case the data map becomes very useful for navigating through data at large temporal scales.

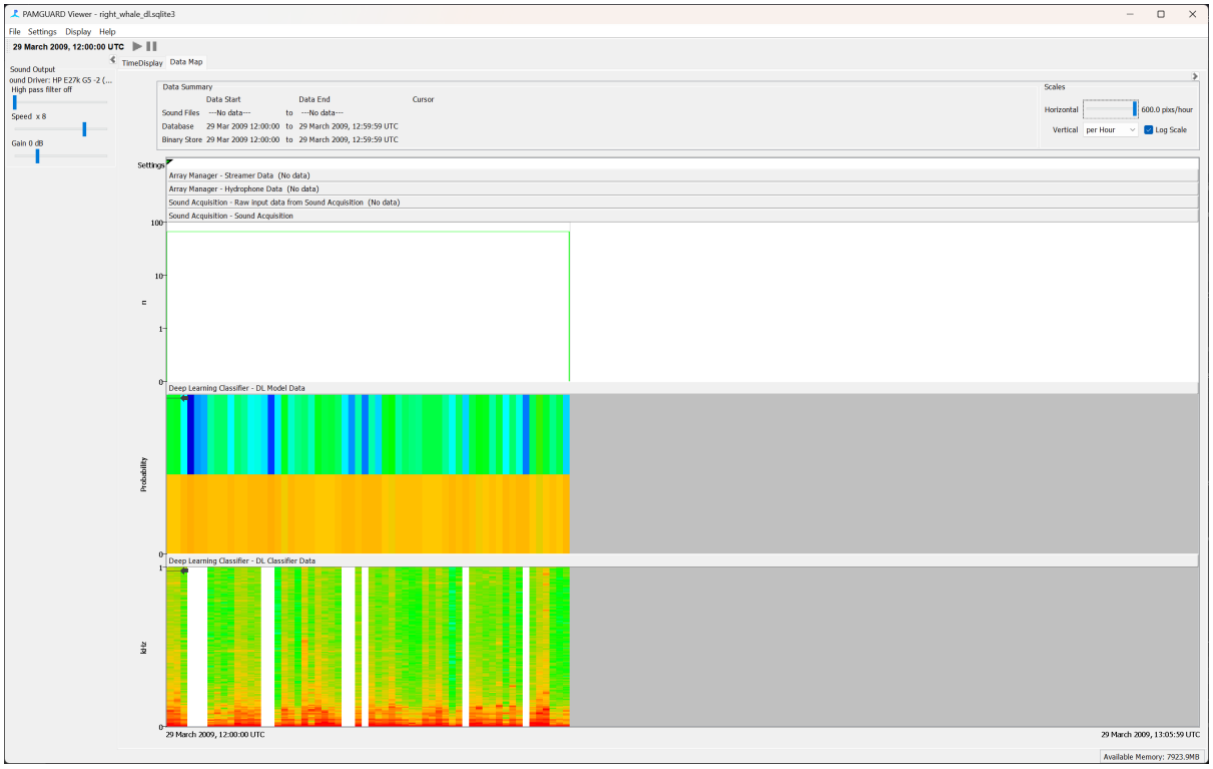


Figure 10. The data map in PAMGuard viewer mode displays different data streams at large temporal scales. It can also be used to navigate to different sections of data.

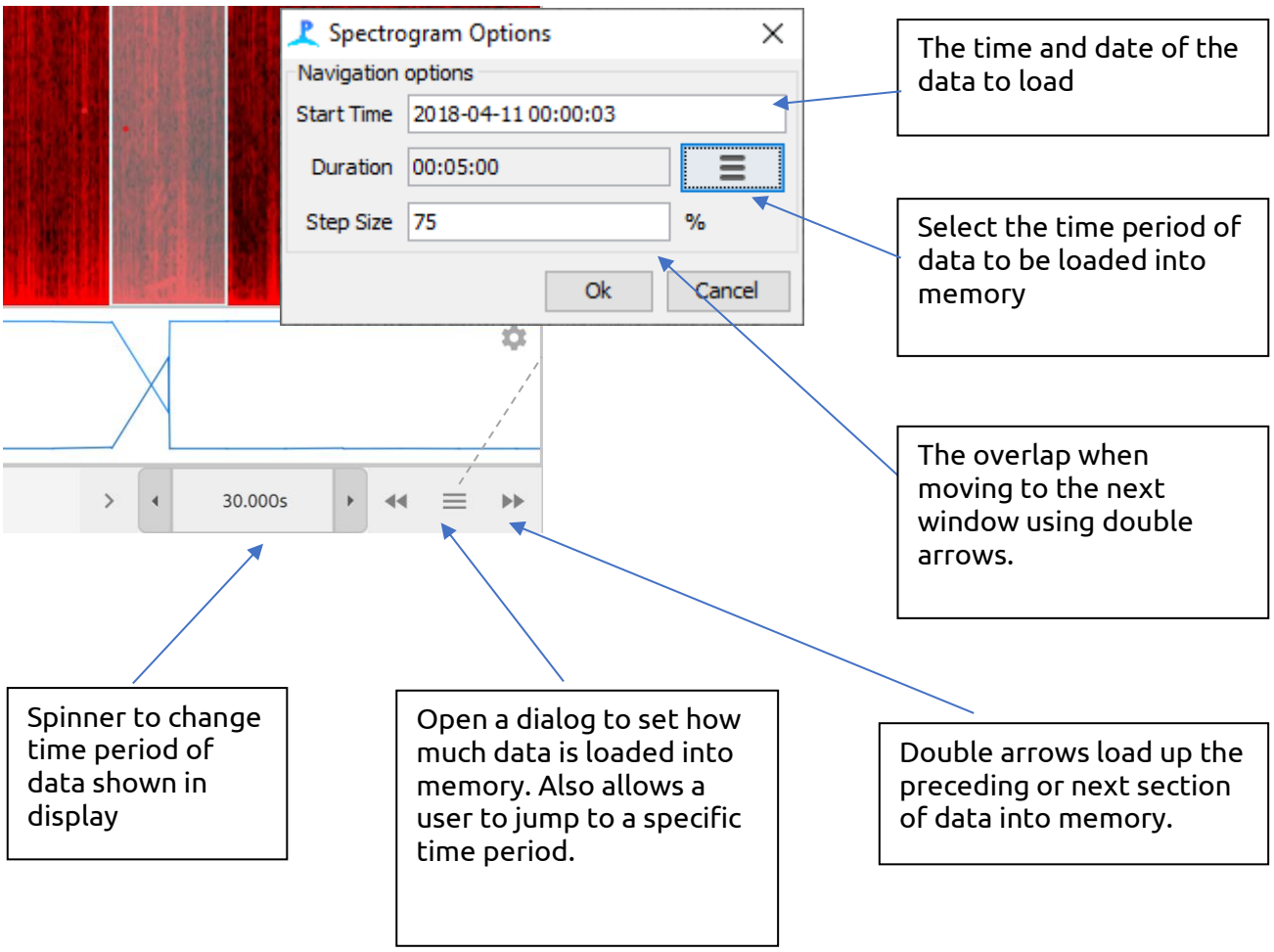
There are two data streams from the deep learning module. ***DL Model Data*** and ***DL Classifier Data***. The ***DL Model Data*** (top panel) is the raw prediction data from each analysed segment of acoustic data. The predictions are shown as colours – where blue indicates low and red high average prediction values in the datagram time bin (here 60 s). The prediction datagram is split horizontally for each class. Here we have two classes, so the lower half of the prediction datagram shows the noise class, and the upper half shows the right whale class (since there are far fewer right whales than noise than noise the upper half is mostly blue, i.e. low average prediction).

The ***DL Classifier Data*** stream shows only the window segments which have passed binary classification i.e., passed the user set prediction threshold. This datagram shows the average frequency distribution within classified clips. As you can see in this datagram there are only a few sections of data that contain right whale classifications (Note that on most datagrams in PAMGuard, white indicates effort but no data, and grey indicates no effort).

To navigate to an appropriate section of data right click on the datagram and select ***Centre Data Here.*** PAMGuard will load a section of data into memory and it will be available to explore in different displays.

> ***"FUN" FACT NO. 2***
> Even after running through a detection process the data volumes generated by PAMGuard and stored in Binary files can be too large to load into memory all at once, especially for long-term datasets. To avoid running of memory PAMGuard allows users to select which section of the data they wish to view and only loads these data into memory at any one time.

Click on the ***User Display*** tab to show the time base display. If you have clicked on a section of data with right whale detection, then these will appear in the display. There are a few extra controls on the bottom right of the time base display which can also be used to navigate through the dataset.



Note that you can increase the data loaded in the Time Display by click on the three bars icon and changing ***Duration***. In this case there is not a large quantity of data so we can easily set the duration to a few hours without causing memory issues.

The display will show the predictions and the right whale detections. Just as in real time mode, the continuous prediction results are on the bottom plot and right whale detections are on the top plot.

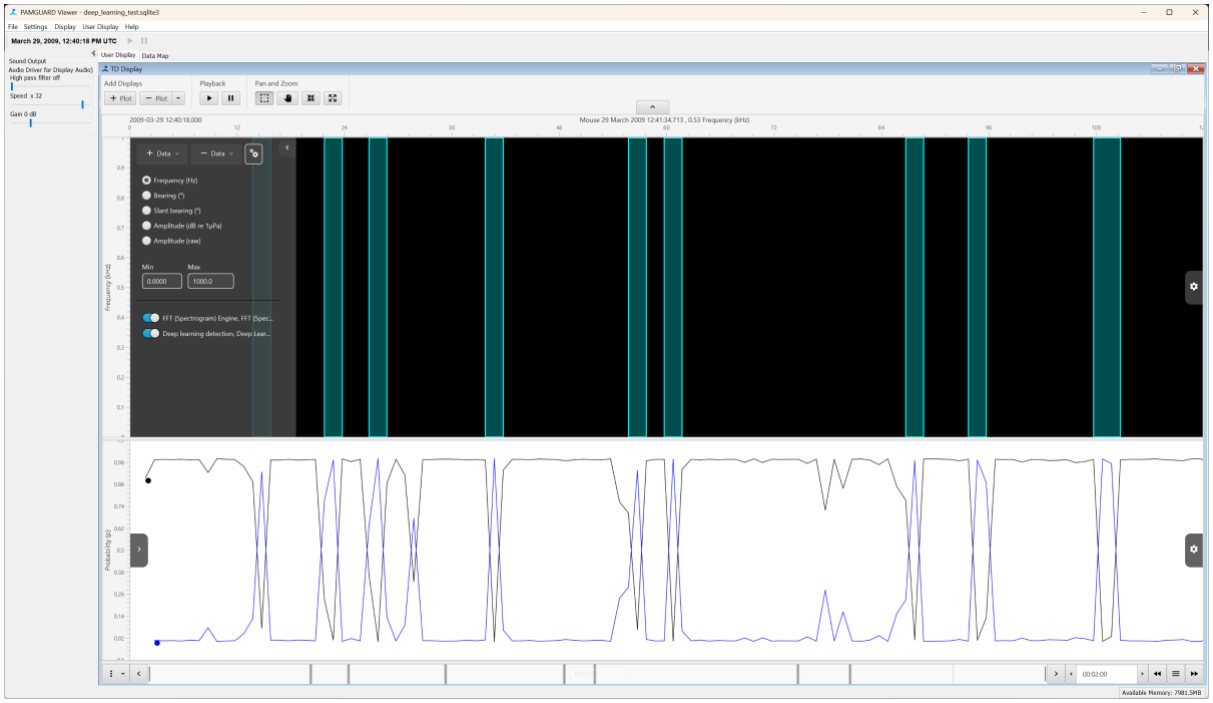


Figure 11. The time base display when first opening in Viewer mode. The top plot shows right whale detections and the bottom plots shows the continuous prediction data from the deep learning model.

Translucent boxes on a black background is not a particularly informative display for data visualisation – we can improve it in two ways, showing spectral data for each detection and/or adding in the raw spectrogram data from the wav files.

First, we can show the raw spectral data of each detection. Open the settings for the plot by clicking the cog on the top right of the plot and then clicking the deep learning icon. This will show the display settings. Click the ***Show spectrogram*** toggle switch and the translucent boxes will change to show a spectrogram. You can also play around with the ***FFT length*** and ***FFT Hop*** of the spectrogram: 512 and 256 samples respectively seems to be a good combination for visualising right whale calls.

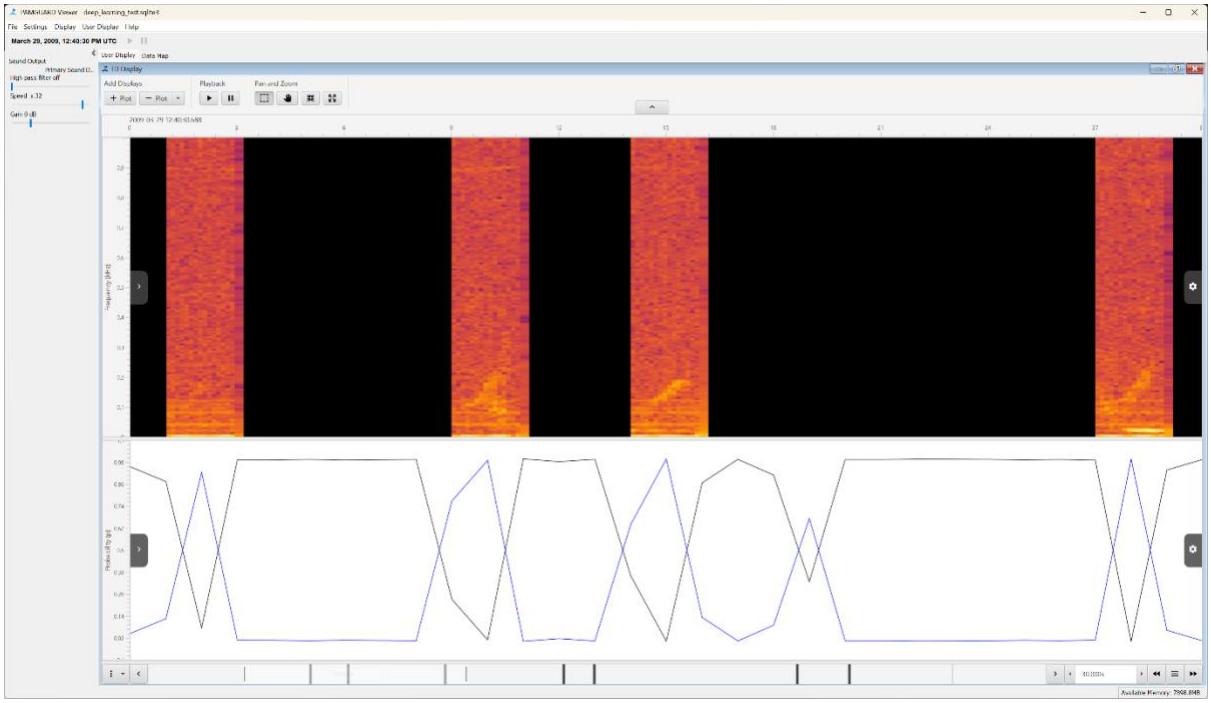


Figure 12. Visualisation of the right whale detections can be improved by showing the spectrogram of each detection rather than just a box.

Secondly, as PAMGuard only saves the raw data for detections so we cannot plot a spectrogram of all the data without going back to the raw sound files. To import raw sounds files, go to ***Settings-> Sound Acquisition.*** Select the check box ***Use offline files***. Because the file location will not have changed since you analysed the data you do not need to do anything more. However, if the sound files ever do change location, then you can simply tell PAMGuard where the new files are by selecting the ***Browse*** button.

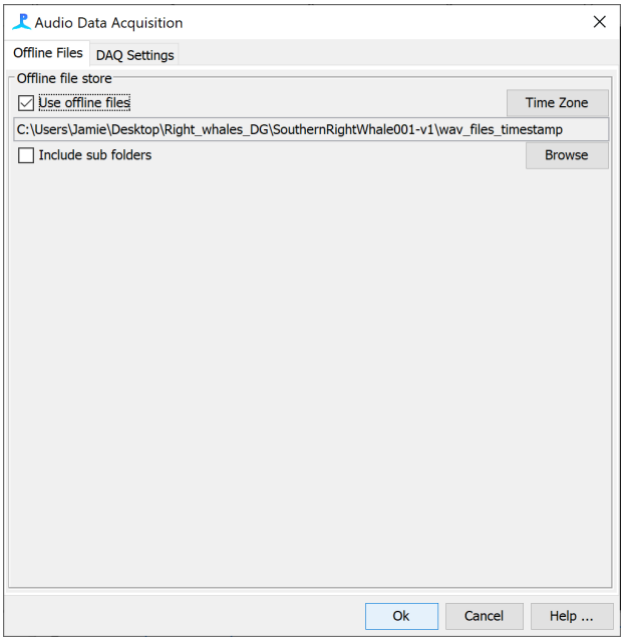


Figure 13. The Sound Acquisition dialog can be used to import raw sound data into PAMGuard viewer mode.

Click ***Ok*** and PAMGuard will map the sound files and automatically plot a spectrogram on the time base display (you may need to load a new section of data first).

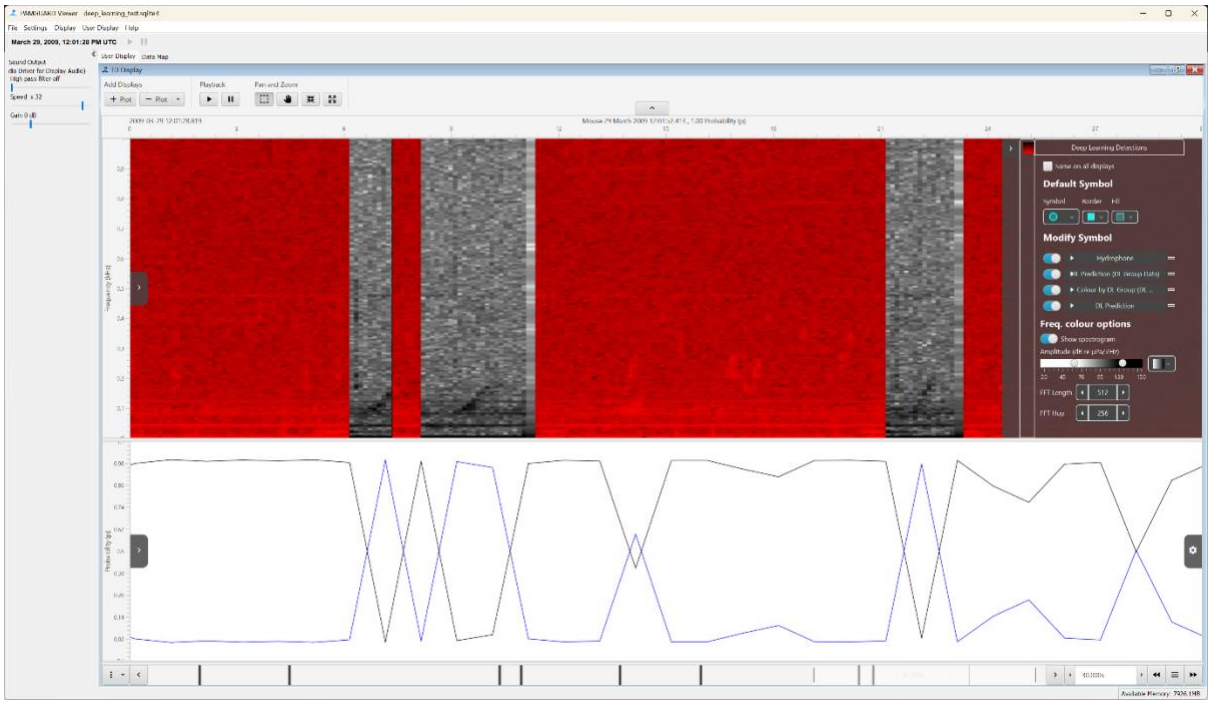


Figure 14A. Raw sound data can be imported into viewer mode and loaded as a spectrogram on the time base display.

The time base display allows closer inspection of detection data – this feature is still experimental and so is not enabled by default. To enable it, select the left settings pane

(top left arrow button), click the cog button just under ***Display Data*** and select ***Adv. Pop up menu*** in the ***Pop up menu type*** selection menu.

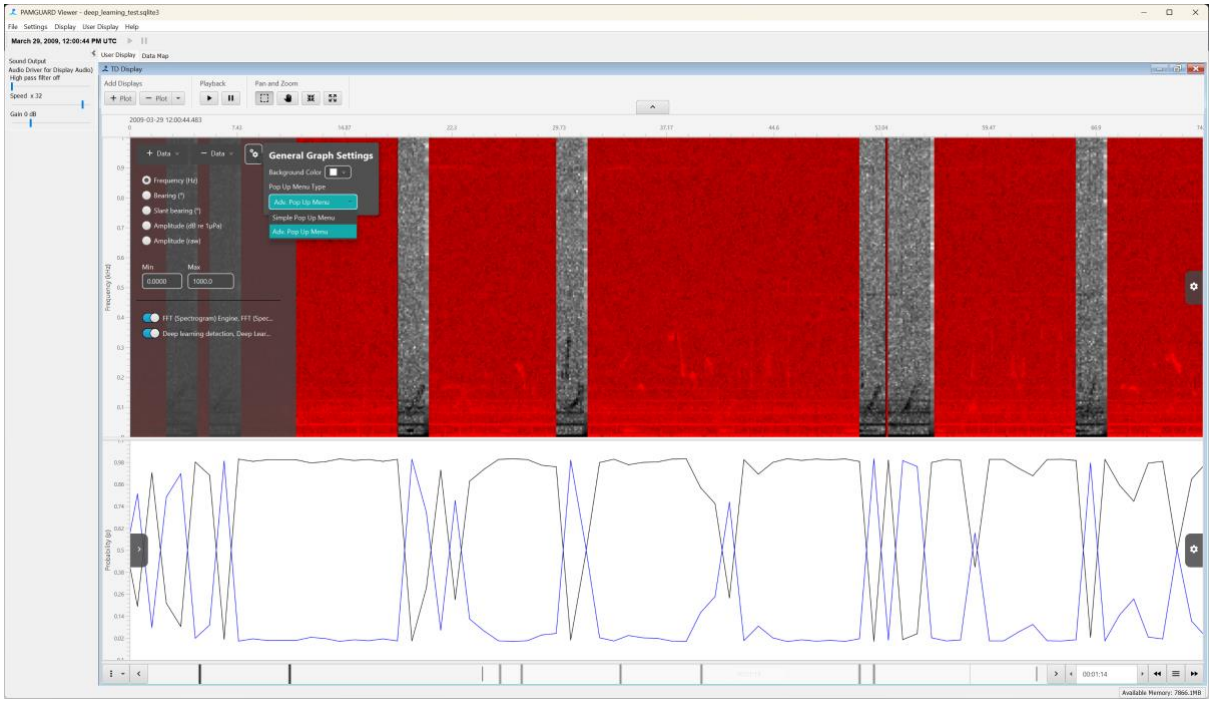


Figure 15. Enable the adv pop up menu to view detections more closely.

Right click on one of the right whale detections. This will bring up the detection preview window showing the waveform of the right whale detection. The detection preview pane allows users to visualise an individual detections in multiple ways (e.g. waveform, spectra, spectrogram), play sound, export and browse metadata.

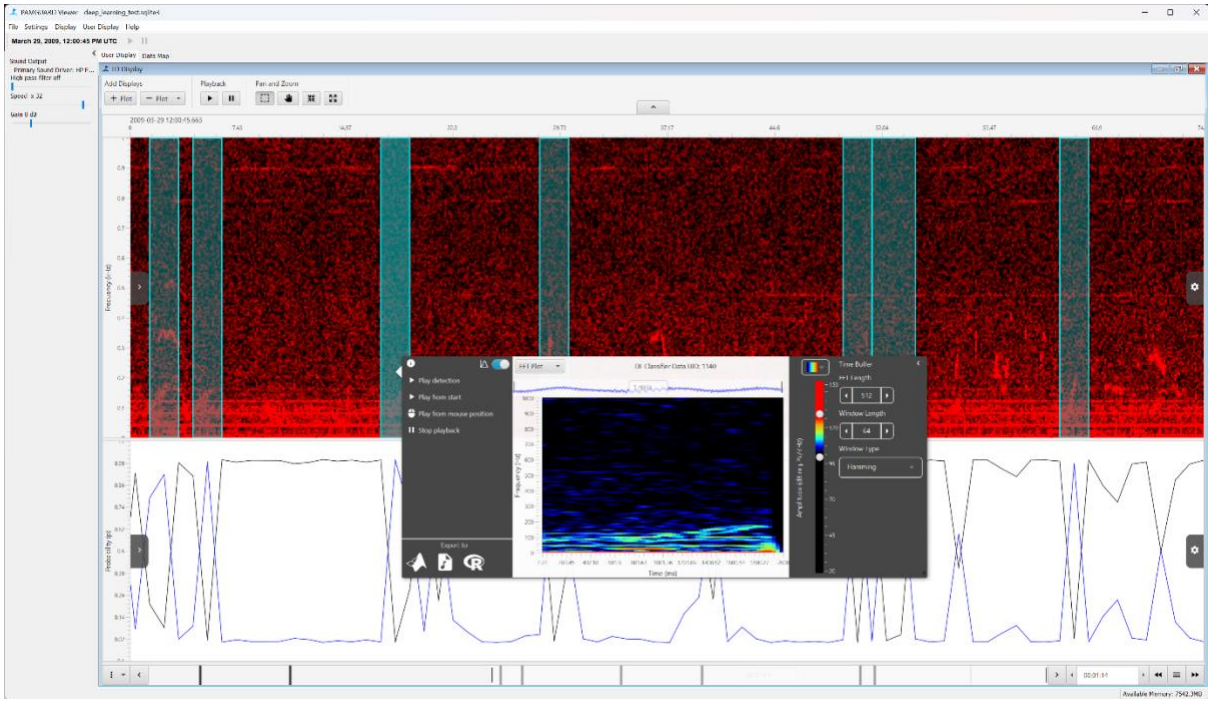


Figure 16. The advanced pop-up menu allows users to explore the waveform and spectra if detection in more detail.

Click on the information icon on detection preview panel to show the detection metadata including the preview. You can export the detection data to MATLAB, R, or a .wav file instantly by clicking the respective icon. Finally try the sound playback controls to listen to the detected right whale calls.

Longer term datasets

We have only processed an hour or so of data. PAMGuard is designed to process large datasests. In the ***folder right_whale_turotiral->long_term_dataset*** there is a long term dataset which has already been processed. Open it up in viewer mode as before and explore the data. Go to different location in the datamap and check out the indovudal detection in the time display. Also note that this dataset includes the older right whale edge detector – can you tell which performs better?

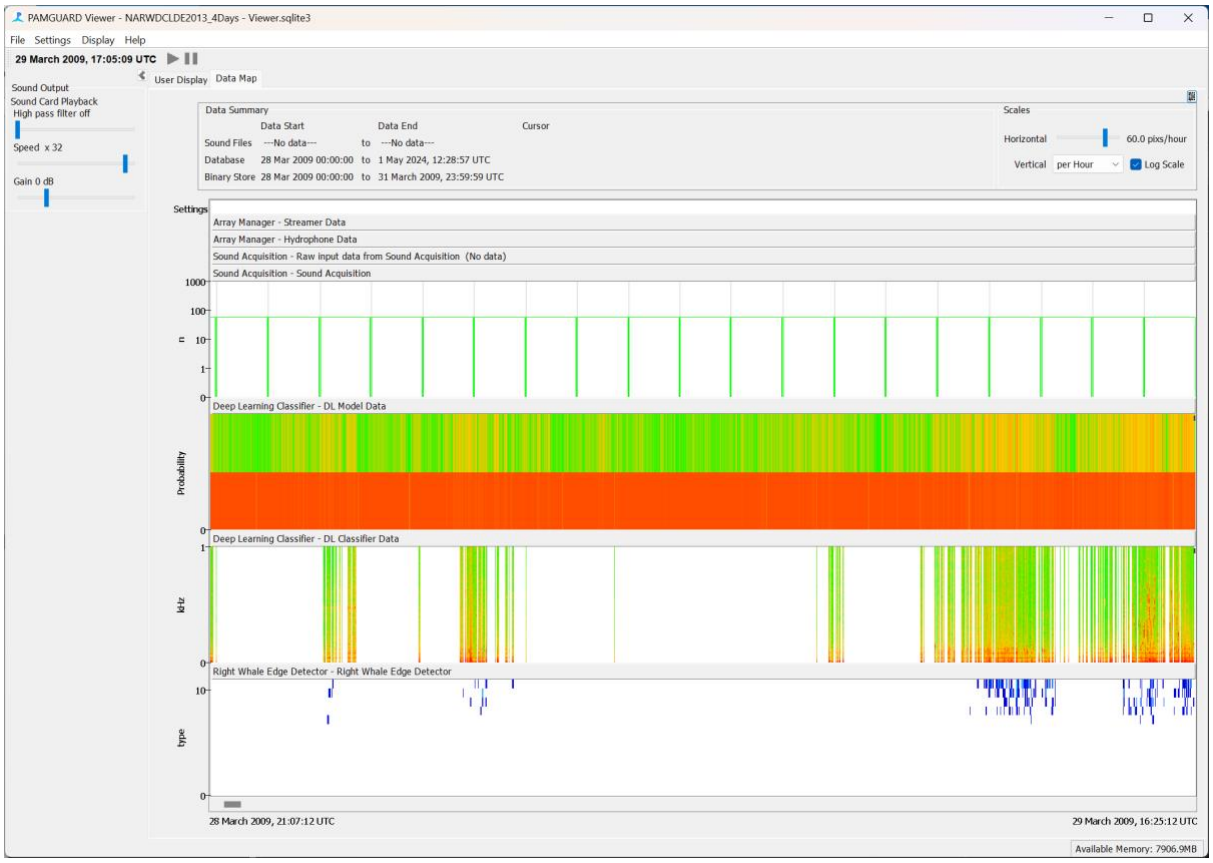


Figure 17. The datamap from a longer-term dataset. The DL Classifier Data stream indicated where detections have occurred. Note there is also a Right Whale Edge Detector data stream which shows the results from an older automated right whale detector.

Extracting detection results in MATLAB

PAMGuard provides a number of useful analysis tools but inevitably researchers will wish to export the data to another program to perform follow up analysis. PAMGuard has a several functions which make this easy for users/

We will export data into MATLAB using the PAMGuard MATLAB library. The library can be found in online at **www.pamguard.org/48_MATLABRcode.html.** Download the library and add it to your MATLAB path.

Use the `loadPamguardBinaryFolder` function to load all detections from a folder of binary files. Replace the path defined by the `folder` variable with the path to your binary folder holding the right whale deep learning detections. Run the script

```
folder = '/Users/au671271/Google Drive/PAMGuard_dev/tutorials/deep_learning/right_whale_tutorial/PAMBinary/';

dldetections = loadPamguardBinaryFolder(folder, 'Deep_Learning_Classifier_Raw_Deep_Learning_Classifier_DL_detection_*.pgdf');
```

All right whale detections will be loaded into a dldetections structure which contains metadata such as date-time, channel number, etc, the raw waveform of the detection and predictions results. Note that the second argument in the loadPamguardBinaryFolder folder is the file mask for the type of binary file that you wish to load – this allows only certain types of data to be loaded, in this case deep learning detections.

The loaded data provides all that is needed for further analysis, plots for papers etc. For example, let's plot the first 36 detected calls in a grid. Load the detections and run the following code (requires MATLAB 2020 +).

```
sR = 2000;
% plot all the spectrograms.
clf
tiledlayout(6,6)
for i=1:min(length(dldetections),36)

    nexttile

    % generate the data for a spectrgram
    [s, w, t] = spectrogram(dldetections(i).wave,512/2, 384/2,[],sR,'yaxis');

    % create the time and frequency matrices required to plot a surface
    [X, Y] = meshgrid(t,w);
    % plot the surface (divide and multiply by 1000 to show milliseconds and kHz respectively)
    surf(X*1000, Y/1000, 20*log10(abs(s))-140, 'EdgeColor', 'None')
    view([0,90])

    clim([90, 130]-140)
    ylim([0,0.5]);
    xlabel('')
    ylabel('')
```

```
    if (mod(i,6)==0)
        c = colorbar;
        c.Label.String = 'Amplitude (dB)';
    end

    %x axis only on bottom plots
    if (i>=20)
        xlabel('Time (ms)')
    end

    %y axis only on left most plots
    if (mod(i-1,6)==0)
        ylabel('Frequency (kHz)')
    end
end
```

All the detected right whale calls are plotted - it's that easy!

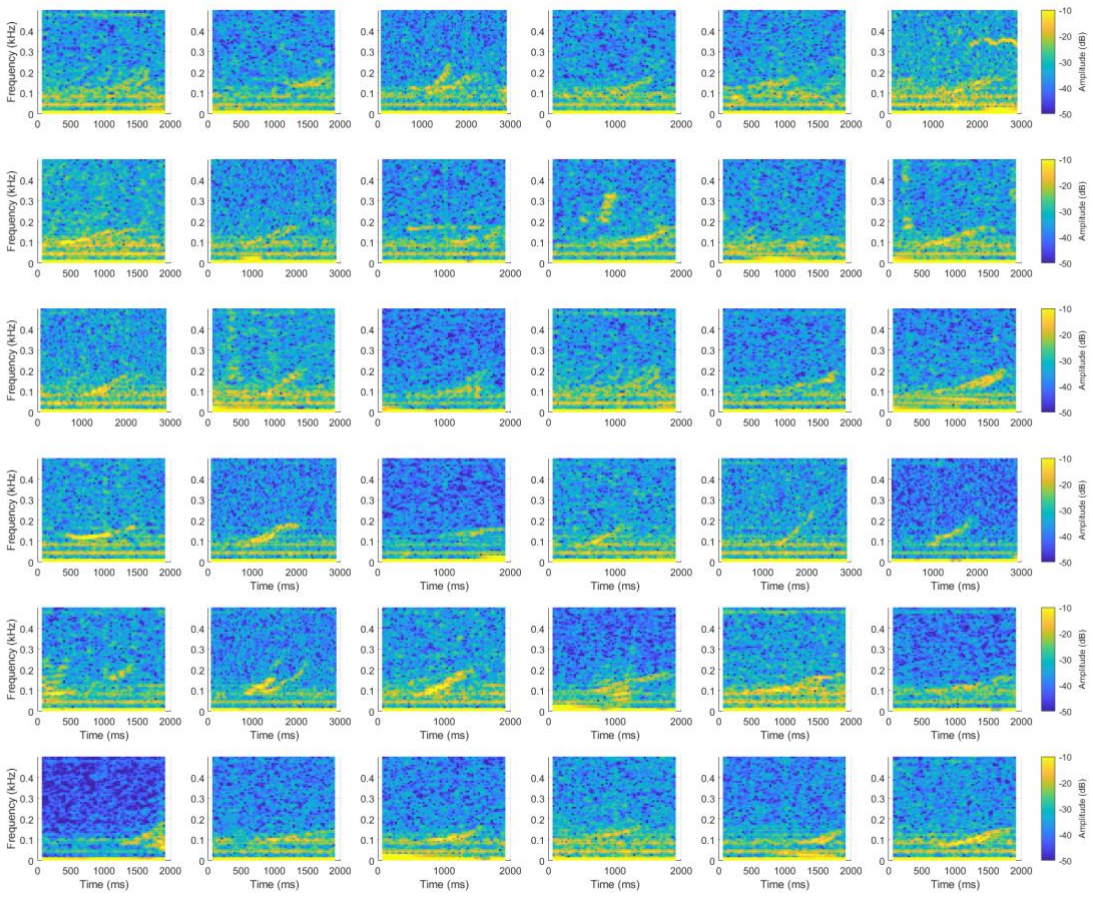


Figure 18A. Spectrograms of all the right whale calls detected by the deep learning classifier in PAMGuard. The waveforms of the calls were loaded into MATLAB using the PAMGuard MATLAB library and then plotted using MATLAB's nextile *and* spectrogram *functions.*

Extracting results in R

The PamBinaries library does not have all the functionality of the MATLAB library so the equivalent R code is a little more complex. Make sure you have downloaded the *PamBinaries* and *phonTools* packages and then run the following the code.

```
#Load right whale detections from PAMGuard deep learning

library(PamBinaries)
library(phonTools)

#count the number of classified clicks in a folder of pamguard files.

#folder to a set of binary files.
folder='C:\\Users\\Jamie Macaulay\\Desktop\\Right_whale_DL_tutorial_test\\
Binary'
```

```
# list all the file names of the
# specified pattern. IN this case we only want the detection files and NOT
the segmented predicitions.
fnames <- list.files(folder, pattern = "Deep_Learning_Classifier_Deep_Lear
ning_Classifier_DL_detection.*.pgdf$",  recursive = TRUE)

count =0; #the classified click counter.
par(mfrow=c(5,5))
for (val in fnames) {
  print(val)

  #load each click file.
  dldetections <- loadPamguardBinaryFile(file.path(folder, val))

  #iterate through the click files to count the classified clicks.

  for (adetection in dldetections$data) {

   S = spectrogram(ts(adetection$wave, frequency = 8000), fs = 8000, windo
wlength = 64,
                   timestep = 6 , padding = 0,
                   preemphasisf = 0, maxfreq = 500, colors = TRUE,
                   dynamicrange = 60, nlevels = dynamicrange, maintitle =
"",
                   show = TRUE, window = 'hanning', windowparameter = 3,
                   quality = FALSE)


  }

}
```

You will see the plot in the Plots tab in R Studio. Note that in this example there are fewer plots because the data was processed with a higher threshold of 0.9 instead of 0.8.

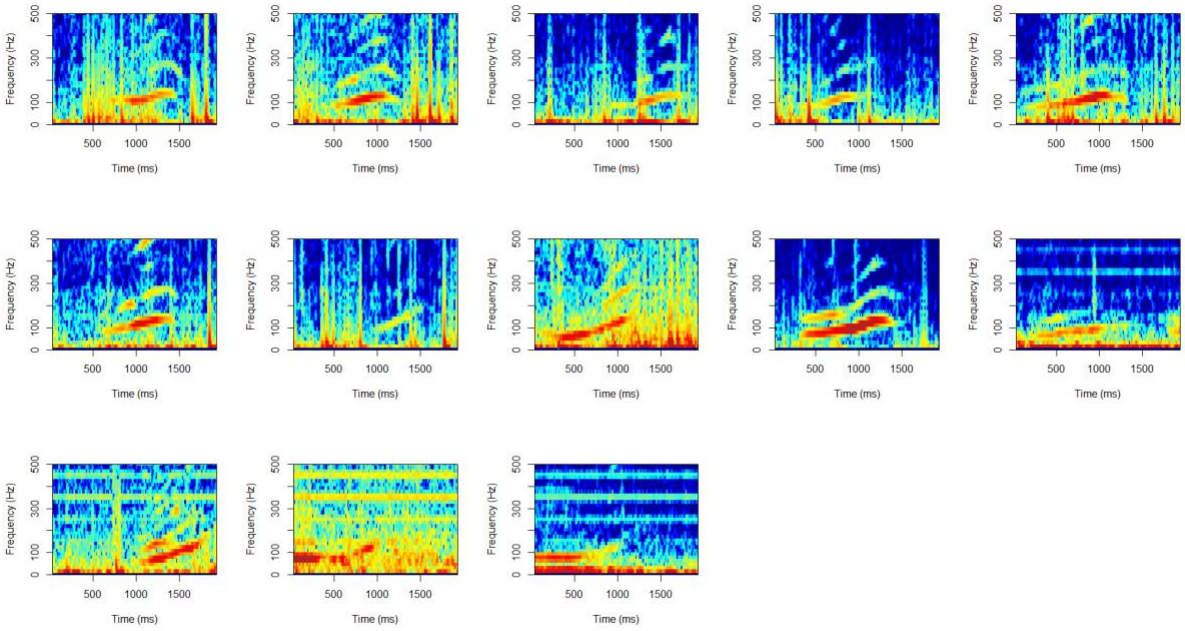


Figure 19A Spectrograms of all the right whale calls detected by the deep learning classifier in PAMGuard. The waveforms of the calls were loaded into R using the PAMBinaries library and then plotted using R's par *and phonTools* spectrogram *functions.*

Using PAMGuard's exporter

PAMGuard has a powerful in-built exporter which allows users to export in various formats, including .mat files (which can be opening in MATLAB and Python), RData (which can be opened using R) and wav files. To export, select ***File->Export Data…*** Select the folder you wish to save to using ***Browse…*** and then click the wav symbol and select ***Zero Pad***. This will export the loaded data as a zero padded wav file i.e. deep learning detections are exported to a single file (or multiple if above 1GB) with zeros placed between the detections so that they are at the correct time within the file.

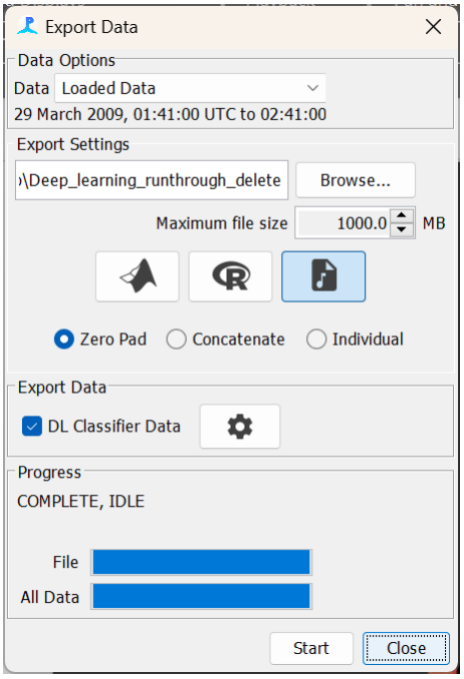


Figure 20A PAMGuard's exporter can be used to export data in multiple formats.

Select the ***DL Classifier data and*** press ***Start*** to export. Note that the settings cog opens the data selector which allows exporting of subsets of deep learning detection e.g. those that pass a higher decision threshold.

Once the data have been exported it can be opened in any sound analysis program, such as Sonic Visualiser.

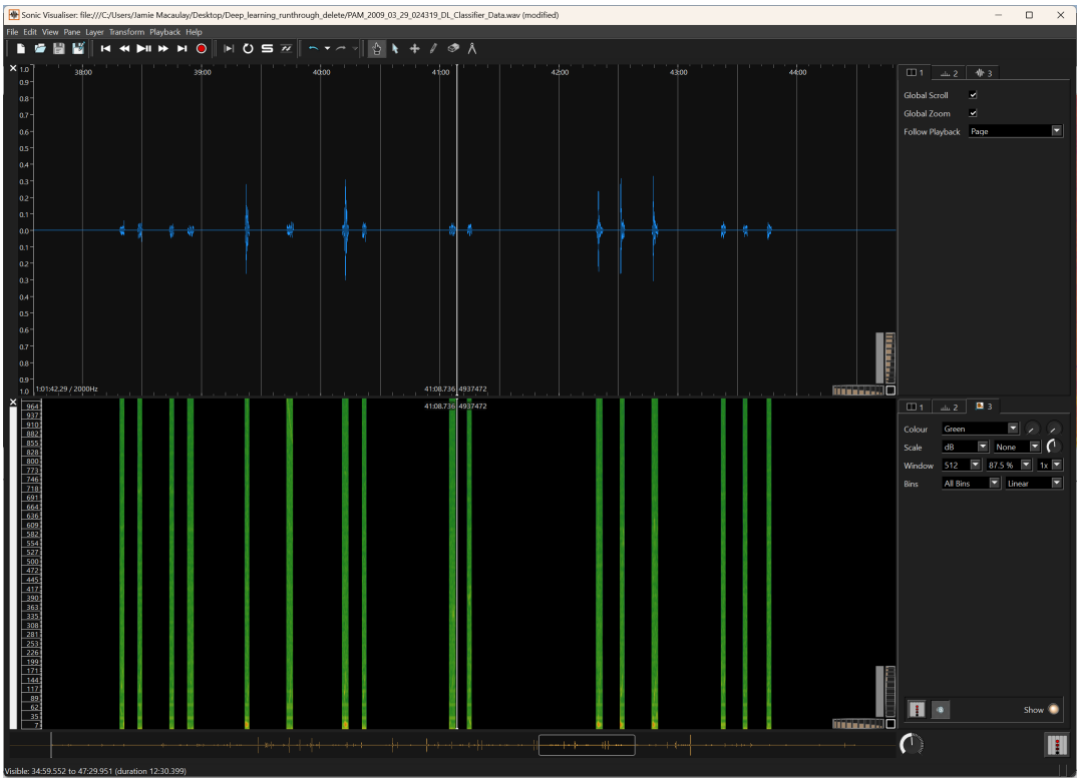


Figure 21A An exported wav file opened in Sonic Visualiser.

Acknowledgements

Thanks to Shiu et al. (2020) for making their right whale model available and open. Thanks to F. Channell, C. Tessaglia-Hymes, and D. Jaskula for deploying and retrieving MARUs, and the DCLDE 2013 organizing committee. We thank the Bureau of Ocean Energy Management for the funding of MARU deployments. Thanks to Marie Roch (San Diego State University) for support in integrating the right whale model into PAMGuard.

Analysing bat recordings using Deep Learning

Just like the marine environment, automatically detecting terrestrial species in acoustic recordings can be difficult. Bats produce directional ultrasonic echolocation calls to hunt and sense their surroundings. The narrow beam profile of bat calls means that temporal and spectral properties of calls change with the direction of the animal. Add in noise and, source level variation and the presence of multiple species and you have a classic example of acoustic data that contains a myriad of different sounds with significant intra and inter-species variation.

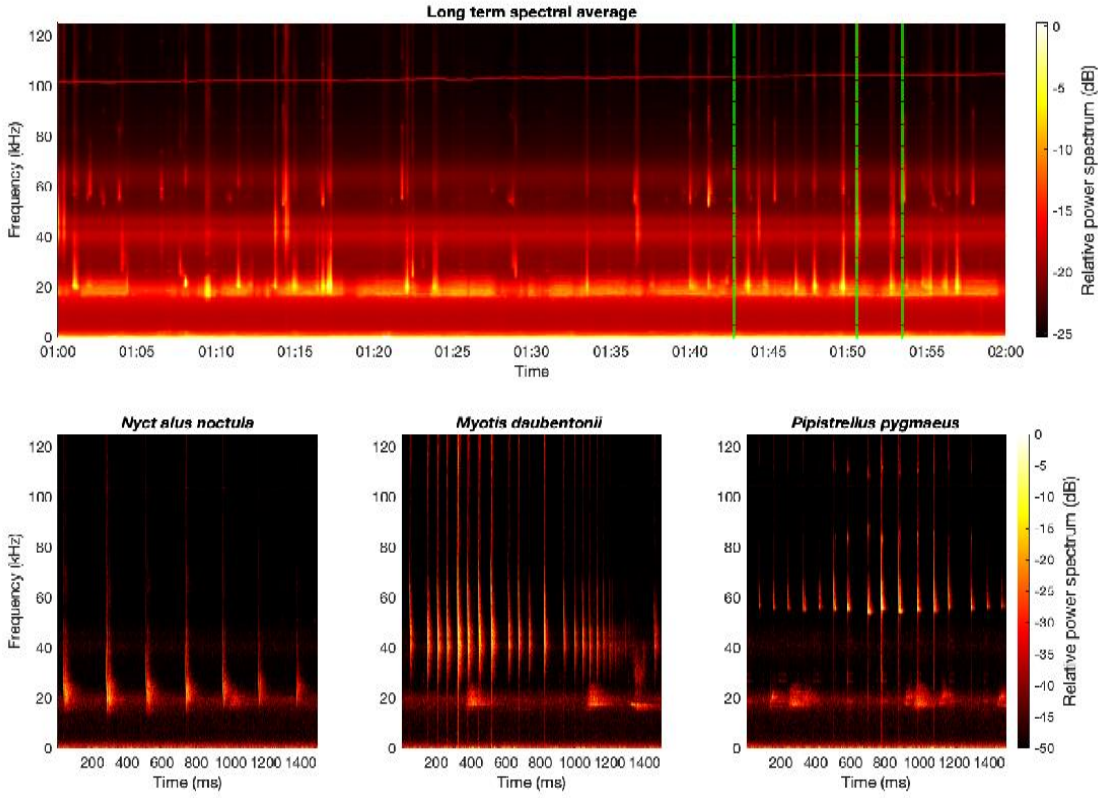


Figure 1B. A long term spectral average of one hour of recordings (top plot) with three examples of different bat species.

Deep learning classifiers have shown promise in automatically identifying bat species from acoustic recordings (e.g. bat detective) but there are some significant hurdles to overcome applying deep learning based acoustic classifiers to bats compared to other species. Bat calls are relatively short in duration (on the order of tens of milliseconds) and high frequency (~12-160kHz) compared to the vocalisations of many other species, for example baleen whale calls and delphinid whistles, are often seconds long – two orders of magnitude longer. The short length and the relatively high frequency of bat calls means that identifiable temporal and spectral features are best represented by a spectrogram image which is on the order of the call length (10ms) with a short hop size.

It is common for a deep learning classifier to split a large chunk of sound into discrete segments, with each segment converted to a spectrogram image and passed to the deep learning model for analysis. For longer vocalisations this is sensible because each segment takes significantly less time to process than the length of the segment (e.g. it might take around 100ms to process a spectrogram – if that is an image of a 2 second sound chunk then

you could process at 2s/100ms = x20 real time). However, this segmentation approach becomes problematic for shorter calls. For 10ms bat calls, the segments of sound (without any overlap) are smaller than the processing time and so splitting a sound chunk into 10ms segments will mean running at x10 *slower* than real time. So, one month of recordings will take 10 months to analyse!

This is clearly not a practical solution and so we must find a way to utilise the benefits of deep learning algorithms for bat calls whilst running at reasonable processing speeds e.g. >10 times real time i.e. we need to increase the speed by two order of magnitude without significantly compromising classification performance.

There are two possible approaches to this. The first is the brute force option – increasing speed by using more better processors (e.g., a graphics card or Apple M1 processor will give a x3 - x10 increase in deep learning performance). This can be effective for some researchers but adds significant technical and cost barriers. The second option is to reduce the data the deep learning algorithm is required to analyse by first running a simple (and thus fast) detection algorithm on the acoustic data – this is what we will do in the next exercise.

PAMGUARD TIP NO. 1
The deep learning library PAMGuard will automatically detect if there a compatible graphics card on a computer – if so, the deep learning model will be run on the graphics card instead of the CPU which greatly increases computational speed. A Nvidea CUDA enabled card is ideal for this.

Bat call detection in PAMGuard

In this exercise we will split an automated bat call identification workflow into two processes, *detection* and *classification*. For the detection we will use a simple energy detector to extract any transient sounds which will include bat calls and along with a large number of false positives. This initial detection stage will reduce the total raw data available for classification by around 99%. We will then utilise a classification algorithm based on an AnimalSpot deep learning model trained on Danish bat species; this will identify false positive detections and classify the species of detected bat calls.

The PAMGuard click detection module can be used to initially detect bat calls (i.e. will form the *detection* stage of the workflow). First, we need to acquire sound data (in this case raw wav files from an AudioMoth) using the Sound Acquisition module. Add the sound Acquisition module by selecting ***File->Add modules -> Sound Processing -> Sound Acquisition.*** Now open the settings dialog by clicking ***Settings->Sound Acquisition…***. Set up the dialog as follows…

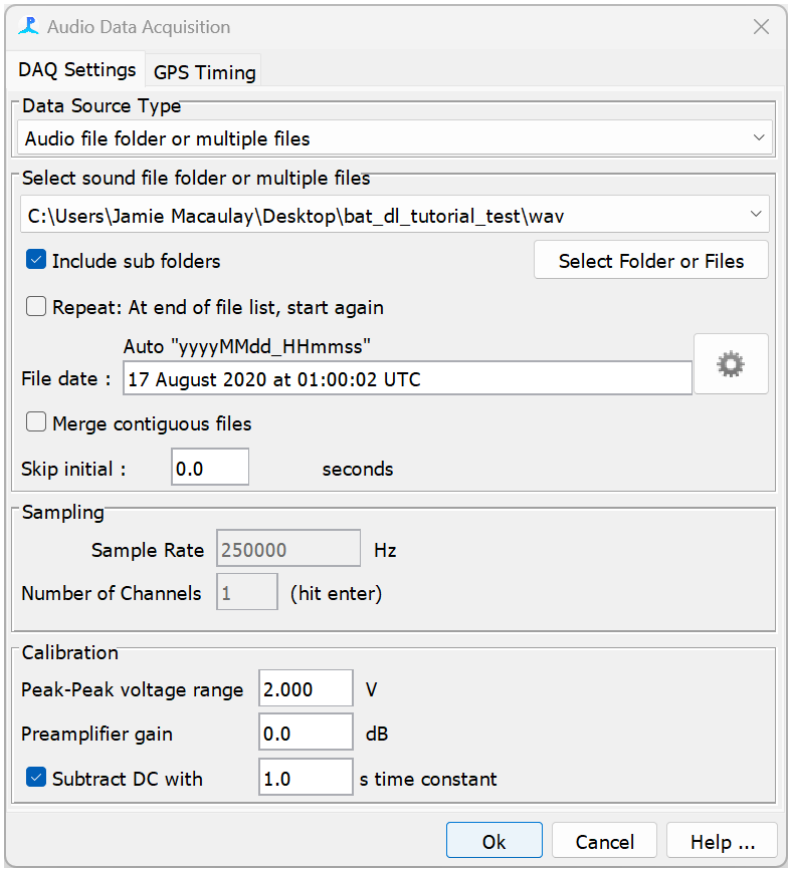


Figure 2B. The Sound Acquisition module after it has been set up correct for high frequency AudioMoth wav files.

Select ***Audiofile folder or multiple files*** from the ***Data Source Type*** selection menu. Click on ***Select Folder or Files*** and browse to the folder that contains wav files in the bat tutorial folder. The sample rate should show up as 250kHz with 1 channel. Everything is now set up so click OK.

Next, we add the click detector module. Note from the name that this was initially designed to detect the echolocation clicks of toothed whales but at its core, it's a simple energy detector which is also suitable for detecting bat calls. (see the PAMGuard help files for more

info on the algorithm). Add the click detection module by selecting ***File->Add modules -> Detectors -> Click Detector.*** Next, we need to set the click detector up for bats as it optimised by default for the much shorter echolocation clicks of dolphins and porpoises. Go to ***Settings->Click Detector - > Detection Parameters…*** Select the ***Trigger*** tab and change ***Threshold*** : 7 dB, ***Long filter*** : 0.0001, ***Long filter 2*** : 0.000001 and ***Short filter*** : 0.05. These settings are more optimal for detecting longer transients such as bat calls and lower the threshold to ensure we have a high false positive rate and so miss as few calls as possible.

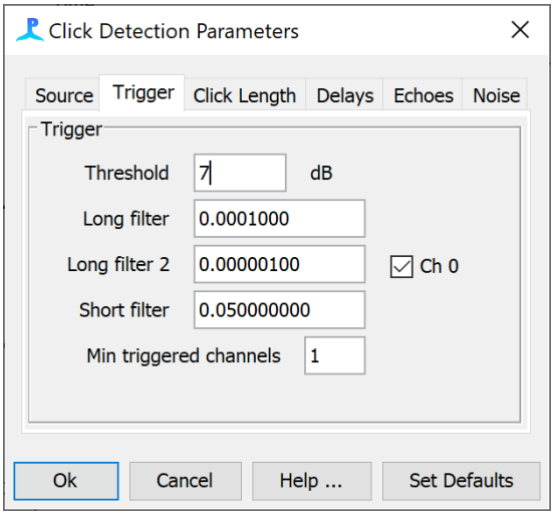


Figure 3B. The Trigger settings for the click detector module.

Once a detection has been registered, PAMGuard will save a raw snippet of the waveform with some padding before and after the detector passes above the threshold for a detection. These settings are again designed by default for the much shorter echolocation clicks of toothed whales and need to be changed for bat calls to ensure that the saved waveform snippet contains an entire bat call. Select the **Click Length** tab and change the ***Min Click Separation*** - 500, ***Max click Length*** – 10240, ***pre sample*** - 300 and ***post samples*** - 200. Click **Ok** to close the dialog.

The click detector can also be set to run on a filtered data. This is particularly useful for higher frequency species because it prevents lower frequency noise from triggering false detections. There are two filters, the pre-filter and the trigger filter. The pre-filter filters the raw data that is saved (the waveform snippets), and the trigger filter filters the data that is passed to the detection algorithm. Open the pre-filter dialog by selecting ***Settings->Click Detector -> Digital pre filter…*** Create a ***High Pass*** filter at 1kHz with ***Filter order 4*** (this setting is important because the deep learning algorithm can fail if input data is heavily filtered compared to training data).

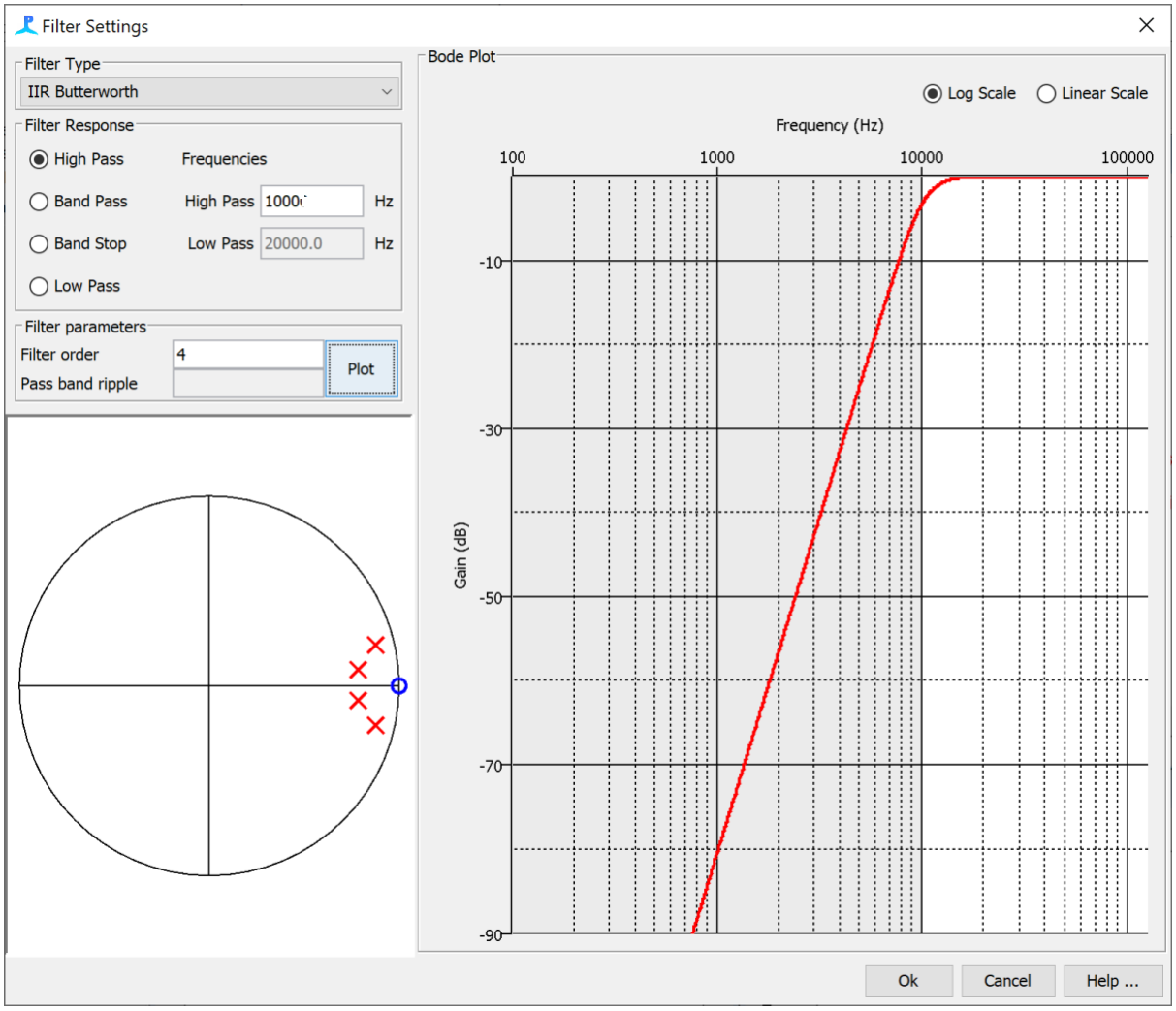


Figure 4B. the filter settings for the click detector module pre filter. Note that PAMGuard uses the same filter settings dialog throughout the program

Now open the trigger filter settings ***Settings->Click Detector -> Digital trigger filter…*** This trigger filter is critical to the functioning of the click detector and should usually be a steep band pass filter between the frequency limits of the target species. In this case we are trying to detect a large variety of bat calls so select ***Band Pass*** filter and set the frequency limits to ***High Pass*** 20000 Hz and ***Low Pass*** 90000 Hz. Finally, change the filter type to ***IIR Chebyshev*** and the ***Filter order*** to 8 to give us steeper edges and hopefully further minimise false positives from low frequency noise.

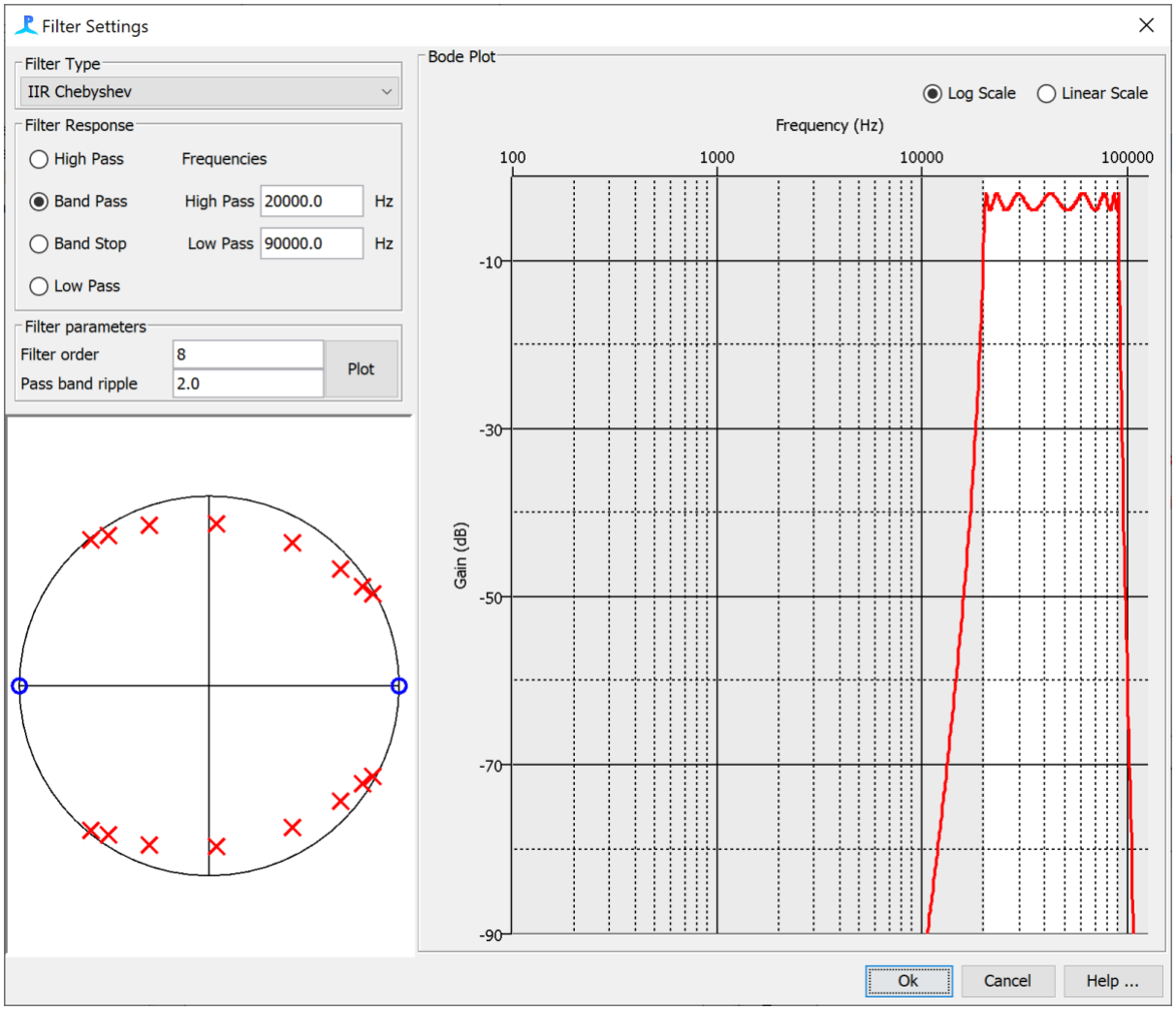
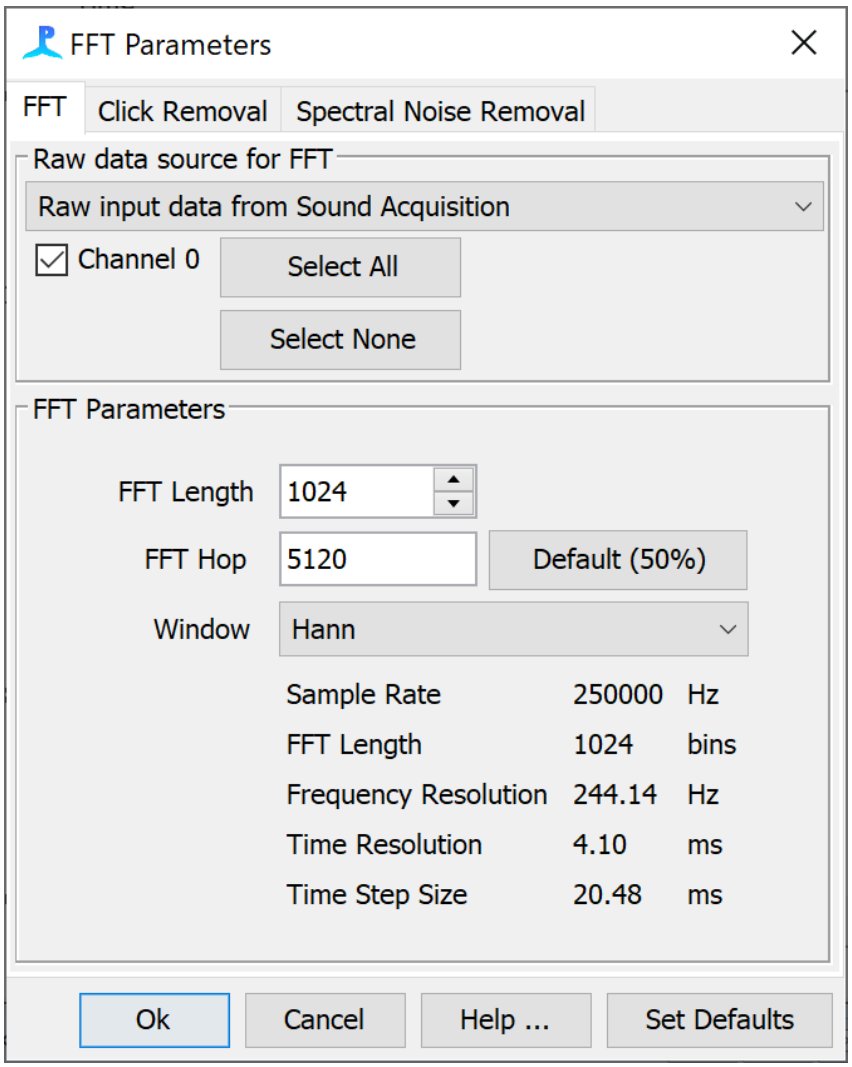
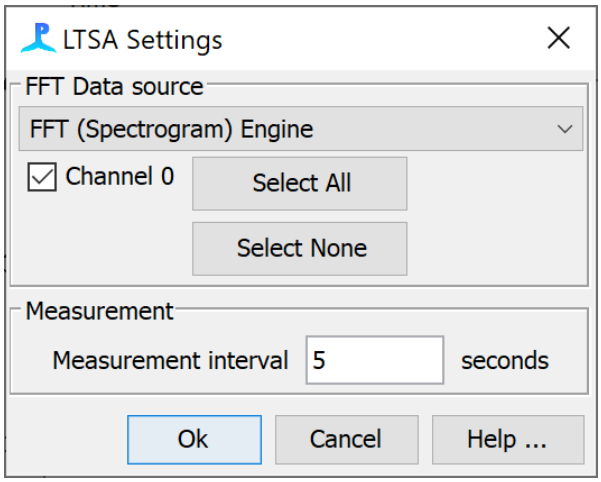


Figure 5B. The trigger filter dialog for the click detector module. The detection algorithm runs on filtered data from the trigger filter and so this filter is important when settings up the click detector module.

For long term datasets it can also be informative to calculate a long-term spectral average, (LTSA), an example of which is shown in Figure 18. To calculate an LTSA we will need to first generate FFT data so add the FFT module by selecting ***File-> Add Modules -> Sound processing -> FFT (Spectrogram) Engine***. Now add the long term spectral average module by selecting ***File-> Add Modules -> Sound processing -> Long Term Spectral Average***. Because we are calculating an LTSA and not using the FFT Module for anything else, we do not need to generate a continuous spectrogram (which is processor intensive, especially for high sample rate data). Open the FFT (Spectrogram) Module settings ***Settings-> FFT (Spectrogram) Engine settings…*** , increase the ***FFT Hop*** to 5120 and then close the dialog.



Now open the LTSA settings ***Settings -> Long Term Spectral Average settings***…, make sure ***Channel 0*** is selected and set the ***Measurement interval*** to 5 seconds. Close the dialog.



Air and water have slightly different conventions for measuring sound. dB, for example, is referenced to 1µPa in water but 20µPa in air. PAMGuard can be changed to set the default

assumptions to a terrestrial instead of marine environment. Go to ***File->Sound Medium*** and select ***Air*** -ignore the warning that pops up**.** Note that throughout PAMGuard dB will now be referenced to 20uPa, hydrophones will be renamed microphones and depth renamed height. Amplitude axis will also be scaled to new defaults to better encompass typical received levels in terrestrial acoustics.

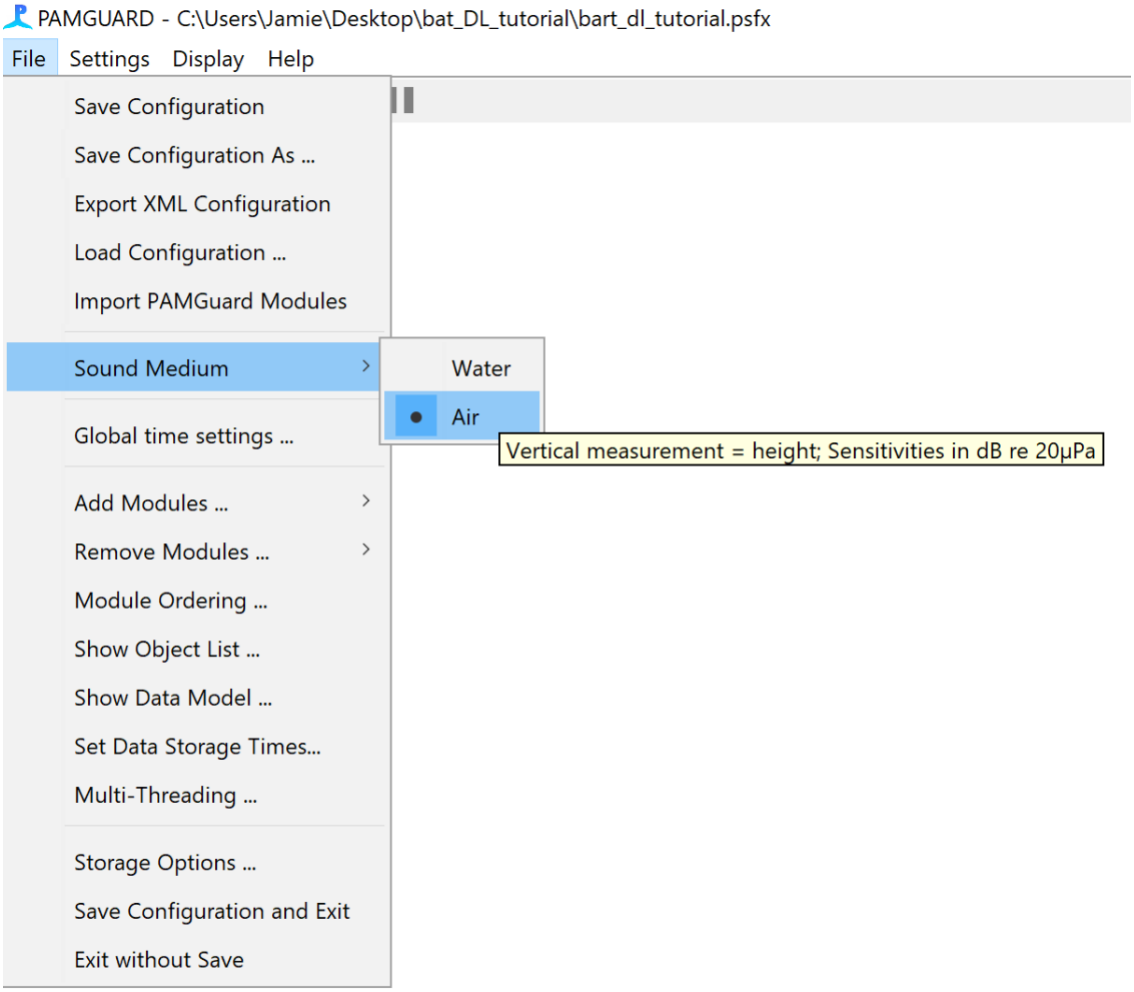


Figure 6B. Setting PAMGuard to Air mode will change the dB reference to 20uPa and change text to be more terrestrial friendly (e.g. refer to receivers as microphones instead of hydrophones)

Everything is now set up to run the detector on raw data but we still need to tell PAMGuard how and where to save our results. Go to ***File -> Add Modules -> Utilities -> Binary Storage*** and then ***File -> Add Modules -> Utilities -> Database.*** This will enable PAMGuard to save metadata to a database and the detection results to bespoke data files (called binary files). To select where the data files are stored go to ***File -> Binary Store -> Storage Options…*** and press the ***Browse*** button. Use the folder selection dialog to select or create a new folder in a location of your choice (but remember where it is). Next go to ***File -> Database -> Database Selection…*** and in the settings dialog click ***Browse…*** and create a new SQLite database.

Before running it is always a good to open the PAMGuard data model and check the modules are arranged as we have envisaged. Open the data model by selecting ***File-> Show Data Model ….*** You should see that the Click Detector and FFT module are connected Sound Acquisition module and the Long-term Spectral Average module is connected to the FFT Module.

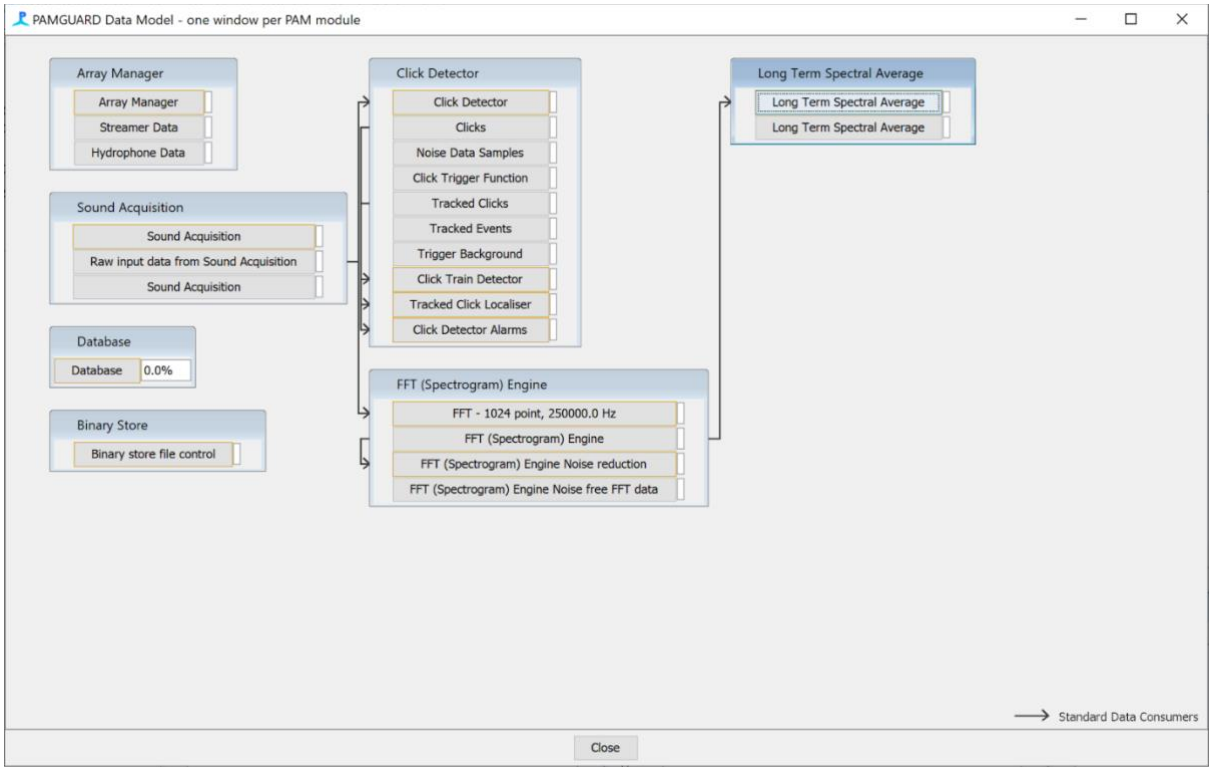


Figure 7B. The data model should look something like this.

SAVE THE CONFIURATION FILE (***File -> Save***) and press the run button (red button). PAMGuard will analyse the current wav files as fast as your computer processor allows. Note that you will want to display clicks in the click display as Amplitude/time instead of Bearing/time because the data are from one hydrophone and thus there is no bearing data. Right click on the Bearing Time Display and select ***Amplitude/Time***

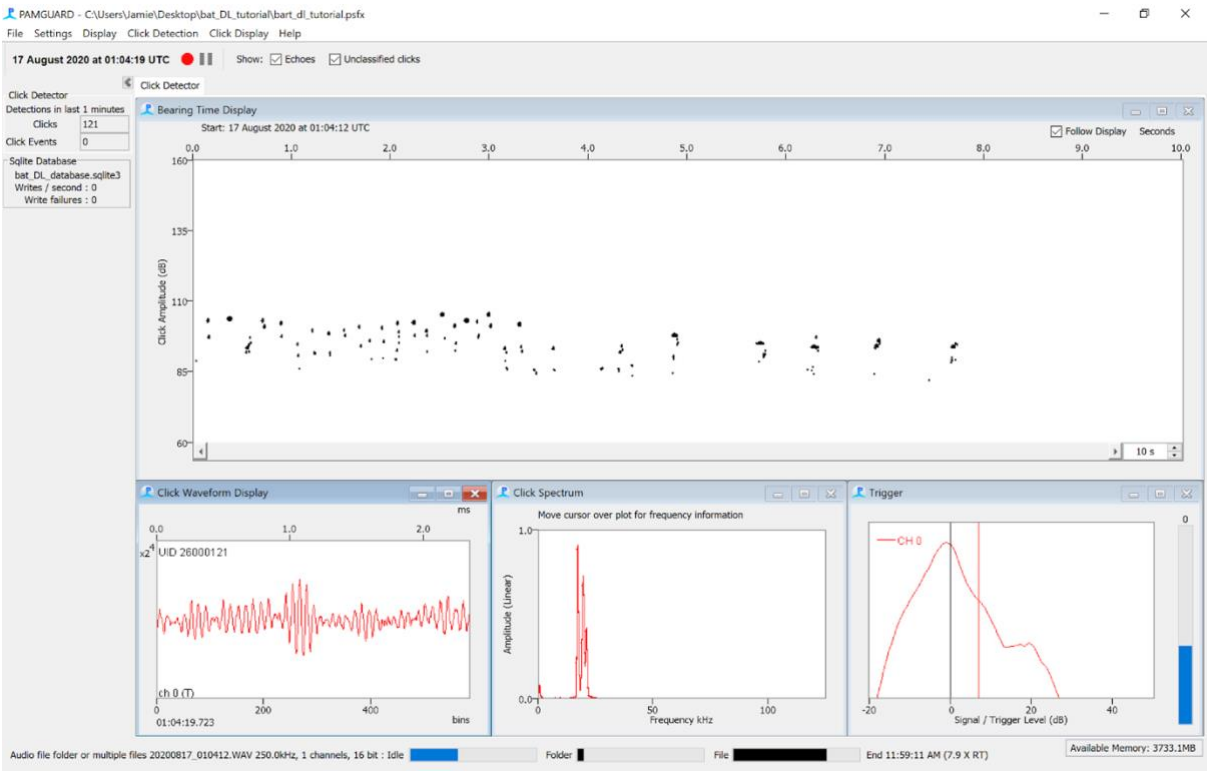


Figure 8B PAMGuard detecting bat calls.

Deep learning classification

We will now add the deep learning classifier in PAMGuard viewer mode (although it also could have been setup up to run in real time mode if we had added the module in the last section).

Once the PAMGuard has finished processing close the program and open PAMGuard viewer mode. The database selection dialog will appear, select **Browse/Create …** and navigate to the database you created in the previous section.

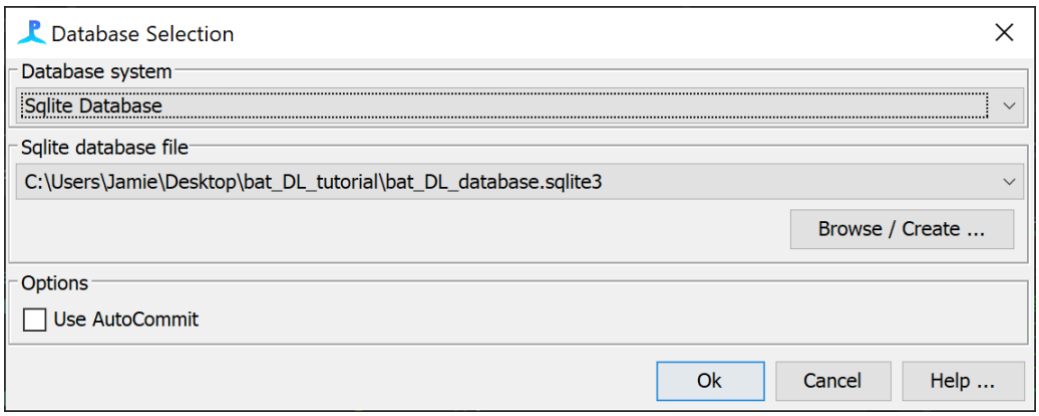


Figure 9B. The database selection dialog when PAMGuard viewer mode first opens.

After you have selected the correct database and clicked ***Ok*** the binary storage dialog will appear. The path to your binary storage folder has already been saved in the database so it

should be correct in the text field but double check. If it's incorrect click ***Browse*** and select the correct folder.

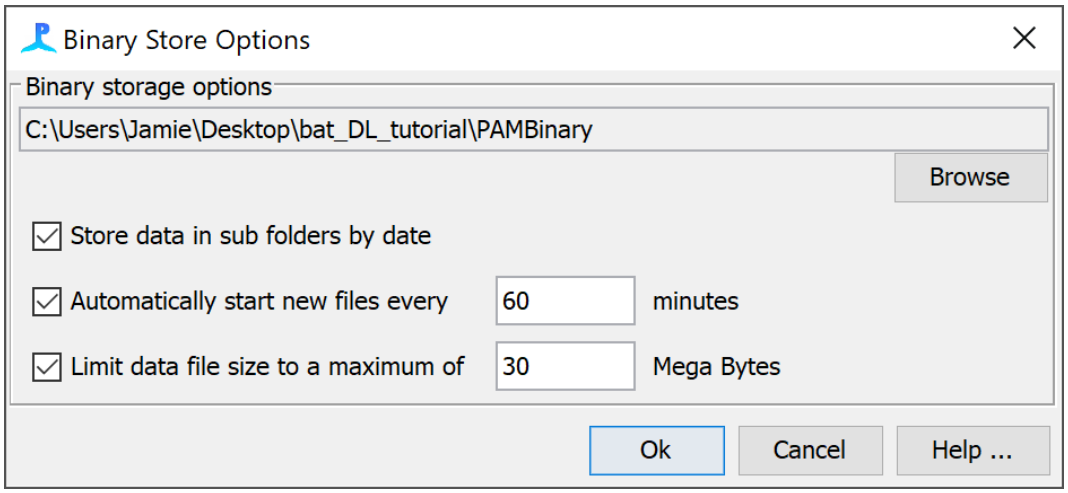


Figure 10B. The binary storage dialog.

The first time a database is opened in viewer a datagram option will appear. This allows the user to set the time bin over which data is summarised so that long-term dataset can be visualised. Set the ***Datagram Bin Size*** to 10s.

PAMGuard will open and look very similar to normal mode except for an extra tab ***Data Map***. The data map shows the user a summary of the different processed data streams and allows users to navigate through datasets in PAMGuard. Here we have only analysed an hour of data but this dataset could be months or even years long and the data map allows us to quickly visualise the different data streams (here click detections and an LTSA) in PAMGuard at large temporal scales. Use the slider in the ***Scales*** section at the top right of the data map to change the temporal scale shown. The data map also provides the user the ability to load up different sections of the data into memory. Right click on the click detection datagram and selected ***Center Data Here***. Now go to the Click Detector display and the data loaded will be from the location you selected on the data map.

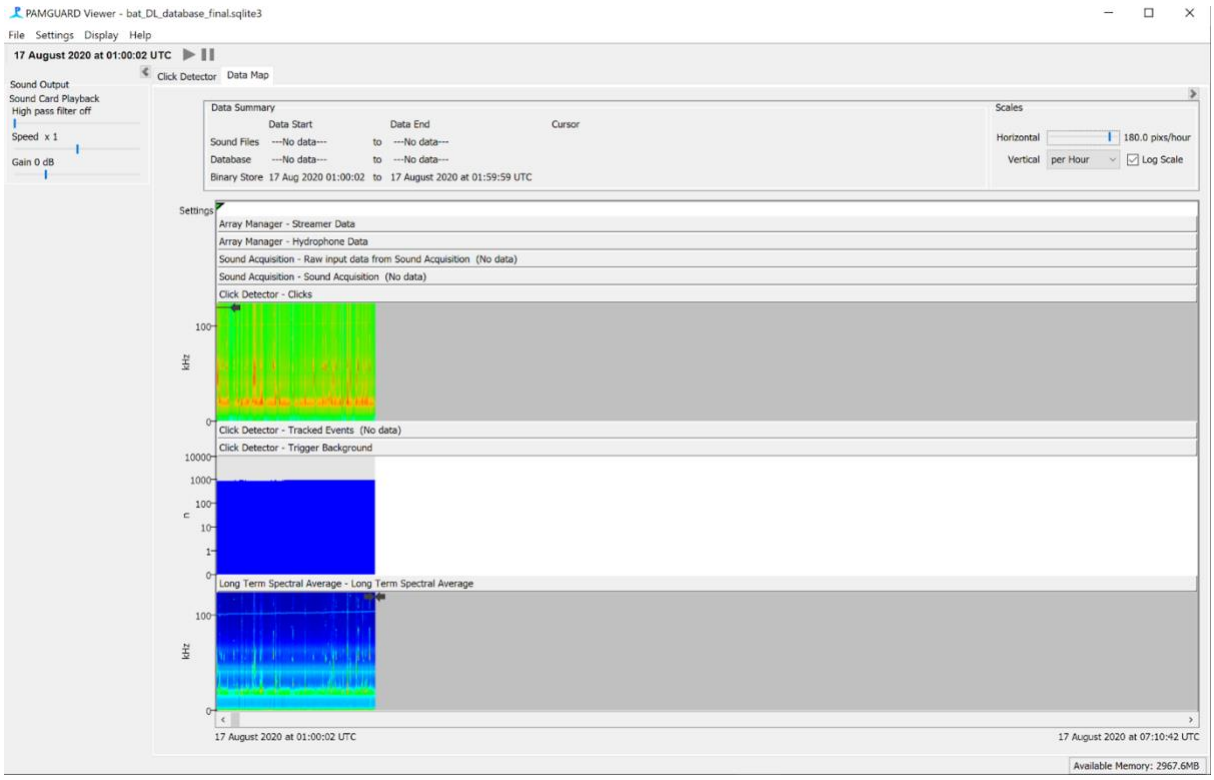
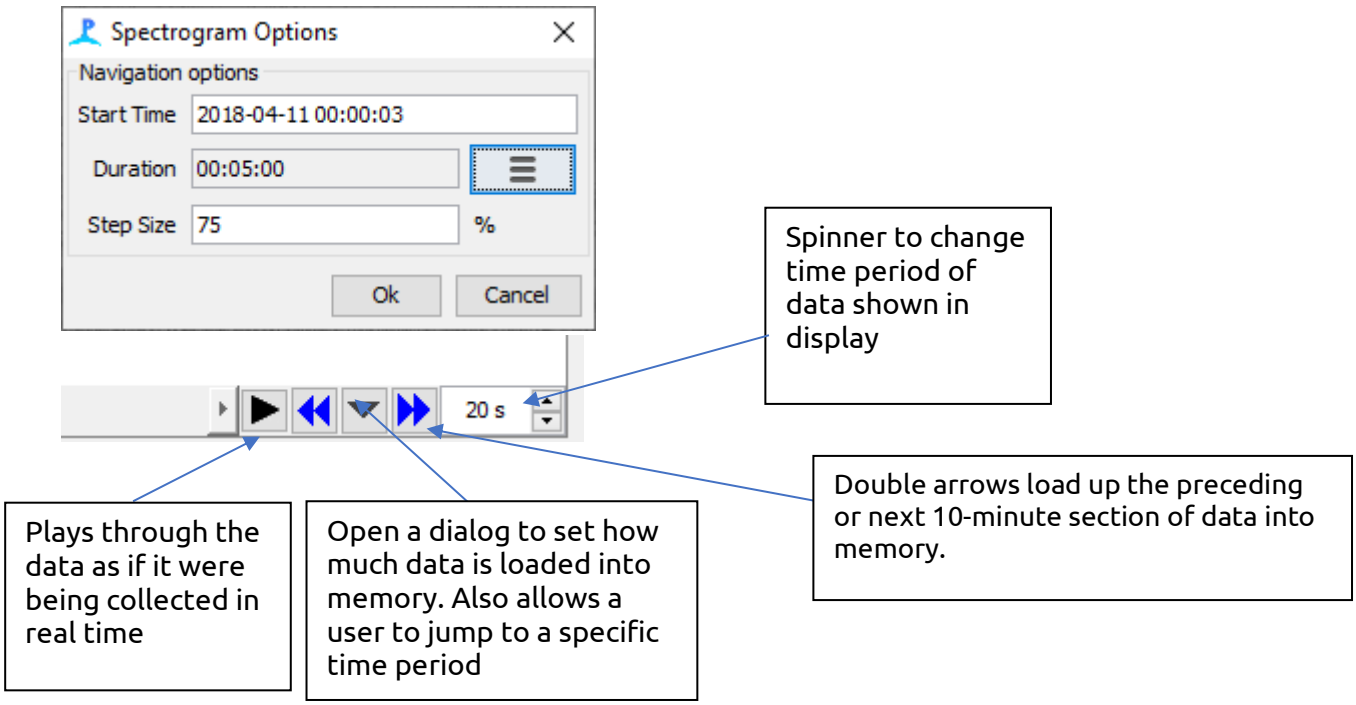


Figure 11B the PAMGuard data map shows users the processed data in PAMGuard at large temporal scales. Right clicking on a data stream allows the user to load up a section of data.

It is also possible navigate through the dataset directly form the click detector display using the navigation tools on the bottom right corner of the main display.



Navigate through the dataset using the datagram and viewer mode controls. Check out the detections in the click detector. You should notice that many of the detections are not bat echolocation calls – just noise and there are also some social calls.

Once you are comfortable navigating through the dataset, the next stage is to classify the call detections to identify the false positives and classify the bat species present. Add the deep learning module ***File->Add Modules -> Classifiers -> Deep Learning Classifier***. Open the deep learning settings dialog ***Settings -> Raw Deep Learning Classifier settings …*** In the ***Raw Sound Data*** drop down menu select ***Clicks*** so that the module knows to analyse detected calls instead of raw sound data from the Sound Acquisition module. Next import the model - we will be using an ***AnimalSpot*** deep learning model which greatly simplifies our setup because it contains all the metadata PAMGuard requires for setting up the deep learning process. Select ***AnimalSpot*** in the ***Deep Learning Model*** selection box. Next select the import model button and navigate to *Bat**.pk* deep learning file in the tutorial directory. The model will take up to a minute to load. Once the model has loaded, select Use ***default segment length*** – this will change the ***Window length*** to the default for the model and change the ***Hop length*** to half the window length.

Select which channels the classifier runs on. If channel grouping is enabled, the detector only operates on the first channel, but when data are saved due to a detection, all channels are saved.

Indicates the type of deep learning framework used. Here we are using an AnimalSpot model i.e., one trained in the AnimalSpot framework

The threshold for binary classification. There are two ways in which binary classification works depending on the **Raw Sound Data** source. **Continuous raw data.** If binary threshold is reached then the raw data is saved as a new detection **Detections (e.g. Clicks, Clips).** The binary classification is added to the metadata of the detection along with all prediction values for the different classes.

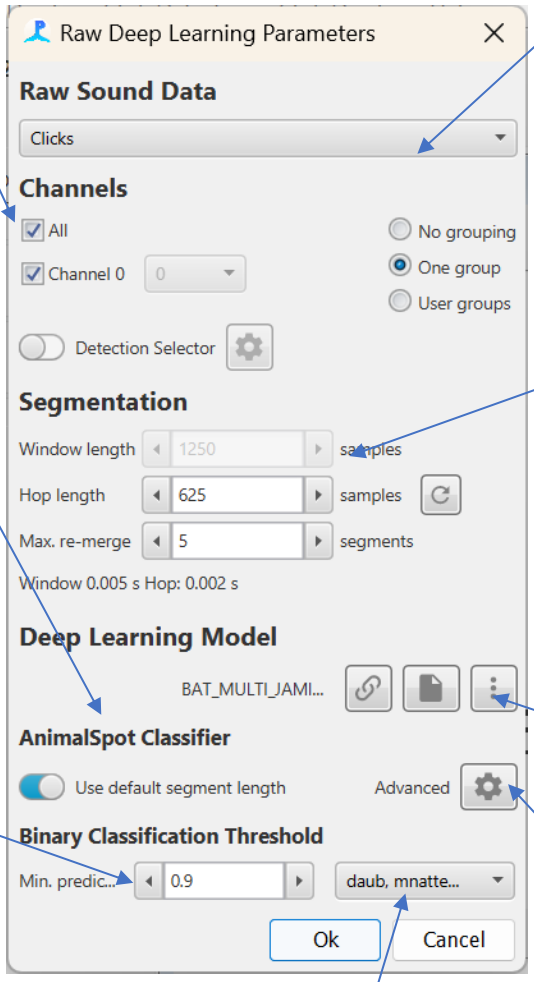


Figure 12B. The raw deep learning settings pane

The source of data. This can be raw sound data or any detector which outputs raw data e.g., the click detector and clip generator.

Window length is the segment size of raw data in samples that the classifier will send to the deep learning model.
Hop length is analogous to a spectrogram hop and is the time in samples the window moves before sending another segment to the deep learning model.

Import the deep learning models from file or link

Opens the advanced settings pane

Select which classes the binary classifier should analyse. In this exercise this is a list of different bat species.

The loaded model will have automatically populated the drop-down menu containing the different species classes in the ***Binary Classification Threshold*** settings. We wish to classify a call if the threshold for any species passes 0.7 so set the **Min. prediction** threshold to 0.7. The class names also contain a class called noise – obviously we do not wish to classify a detection if it's predicted to be noise so deselect noise in the drop-down menu.

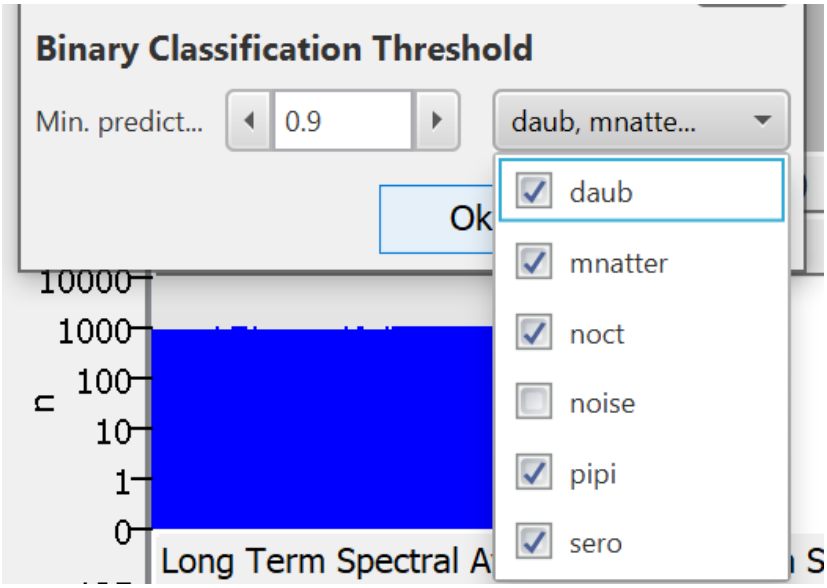


Figure 13B. The drop down meu contains a list of the different species classes and which classes are used for binary threshold.

The great advantage of using an AnimalSpot model (as opposed to a Generic Model) is that it holds the metadata which allows PAMGuard to setup the deep learning model properly i.e., it tells PAMGuard exactly what type of image the model requires and the transforms required to generate that image from raw sound data. You can view this by clicking the ***Advanced*** button. This will show the advanced settings which includes a list of transforms and a preview of the input image for a bat. It is best to leave these settings as they are – any changes may stop the model from working so this should be left to experienced users only

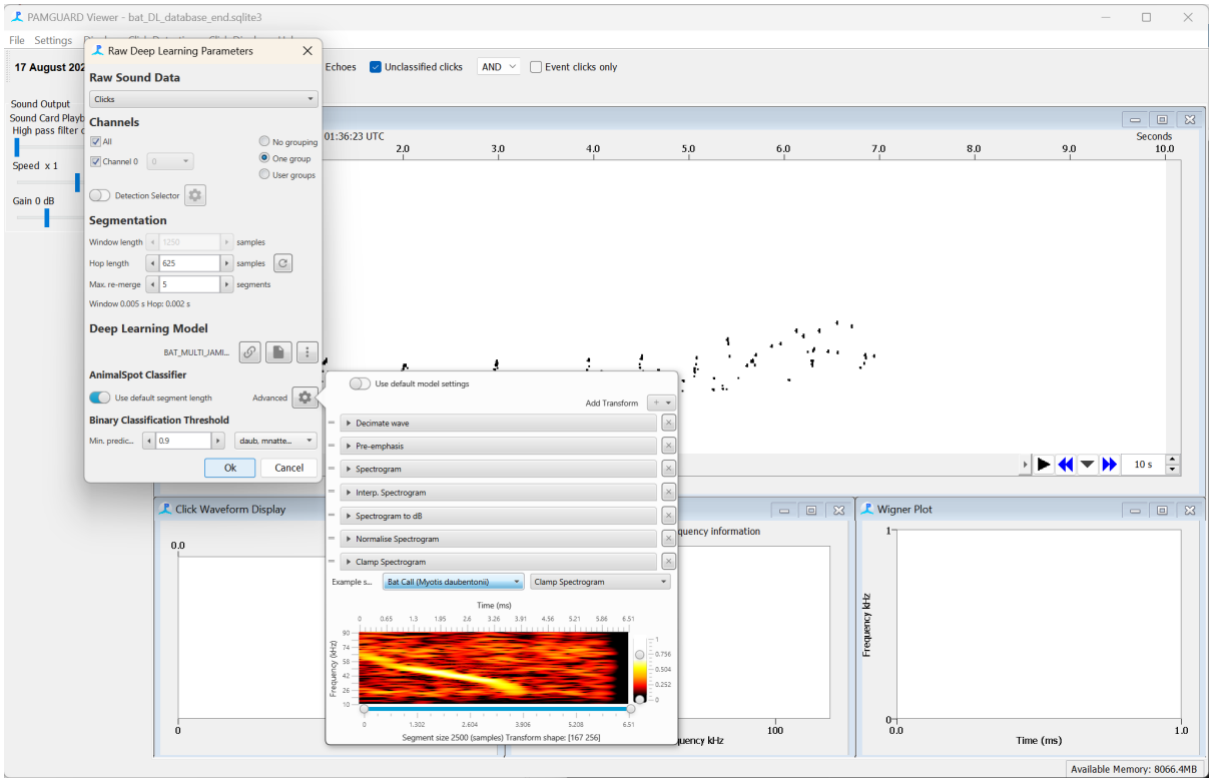


Figure 14B. the advanced settings pane shows a list of transforms that are used to convert a raw segment of sound into an image for the deep learning model. The transforms can be manually rearranged, removed, added and settings changed using this pane, however, this may break the deep learning model so use with caution.

Click ***Ok*** to close the deep learning settings. We now need to run the classifier on the click detections. Go to ***Settings-> Raw Deep Learning Classifier -> Reclassify detections….*** This will bring up PAMGuard's generic batch processing dialog. The ***Data*** drop down menu allows users to select which data is going to be analysed. For now, keep this as ***Loaded data*** i.e. only the call detections loaded into memory will be analysed. To analyse the entire dataset then ***All data*** can be selected. Or to analyse between two specific time ***Select Data*** can be used. Select the ***Deep Learning Classification*** check box and click the cog button. This will bring up the deep learning settings pane shown in the previous steps – we have already set everything up so there is no need to change anything. Click the ***Start*** button to begin processing. The ***File*** progress bar will show the status of processing. This may take a few minutes depending on the processor speed of your computer (or whether you have an Nvidea graphics card)

Note: *If no processing occurs then a common mistake here is that Sound Acquisition is selected instead of Clicks in the Raw Sound Data section of the deep learning settings.*

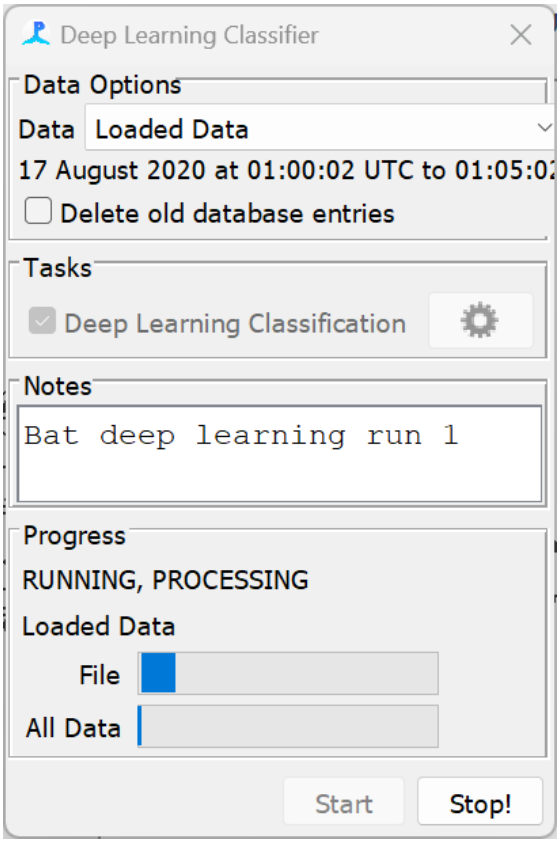


Figure 15B. The batch processing dialog. This allows users to select which data is processed.

When processing detections (as opposed to raw sound data) the deep learning module adds results to the original detection's metadata instead of creating a new detection. To view the results go to the ***Click Detector*** display, select a detection and hover the mouse over it for a few seconds and some text showing the click metadata will appear. The metadata contains a list of the predictions for each species – there is more than one prediction because the clicks are split into overlapping segments based on the ***Window Length*** and ***Hop Length*** parameters in the deep learning settings and each segment is passed to the deep learning model for classification. Thus, there can be multiple segments per call detections leading to multiple predictions.

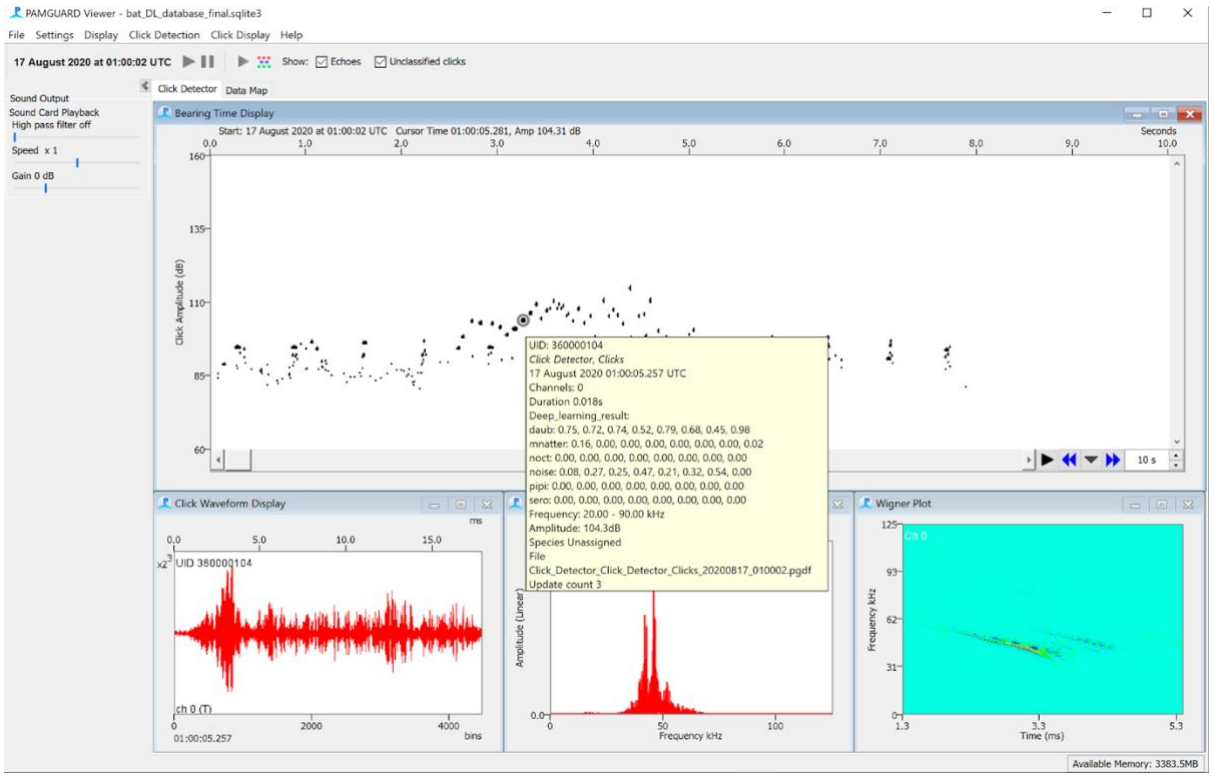


Figure 16B One way to view deep learning classification results is to click on a detection in the click display, hover the mouse for a few seconds and detection's metadata will pop up. The predictions for each class are added to the metadata of each call detection.

Visualising bat data in PAMGuard

The click detector display in PAMGuard is now a relatively old (but much loved and still supported) display which only has a restricted number of options in how to display data. However, PAMGuard has a more modern display for visualising different data streams which is more suitable for displaying bat data.

Create a new user display tab ***Add Modules -> Displays -> User Display.*** Click on the User Display tab and in the top toolbar select ***User Display -> New Time base data display fx.***

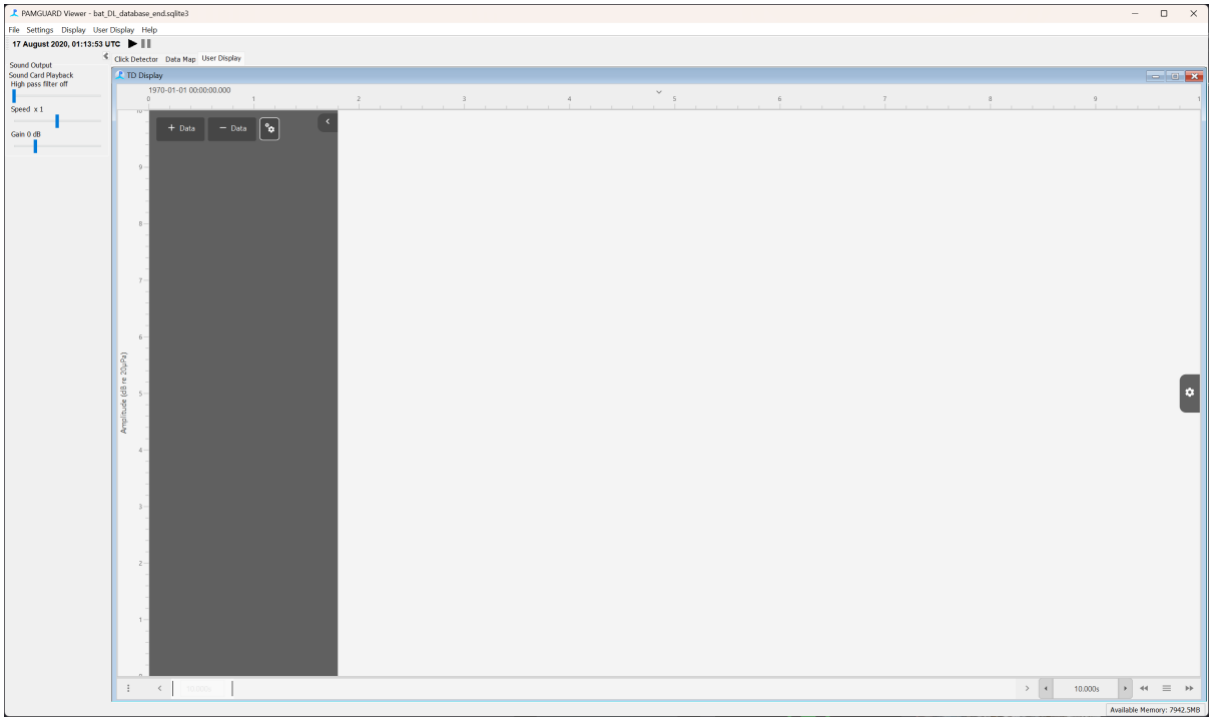


Figure 17B. The time base display can be sued to visualise different data stream in PAMGuard. When it is first added it is blank - the user must add the data stream they wish to view.

The time display can plot almost any data stream which has a time dimensions e.g., detections such as clicks, spectrogram, noise etc. We need to add the data stream we are interested in before the display shows anything. Add the click data stream by selecting ***+ Add -> Click Detector, Clicks.*** The display needs a little nudge to load new data. Click on the "three bar" icon on the bottom right of the display and select the "Duration" field to be ten minutes in the navigation dialog that pops up.

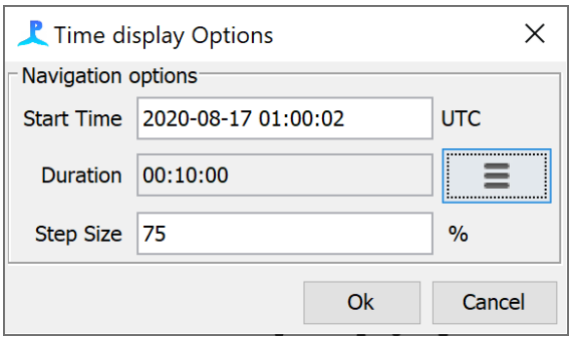


Figure 18B. The navigation dialog for the time base display.

The display will now look very similar to the click display, hover over a click and you will see the same metadata text. Note that the scroll bar at the bottom contains the datagram for the section of data that is loaded. For example, because we have loaded call detections, the scroll bar shows datagram for calls. This allows users to quickly browse to interesting section of data in the display.

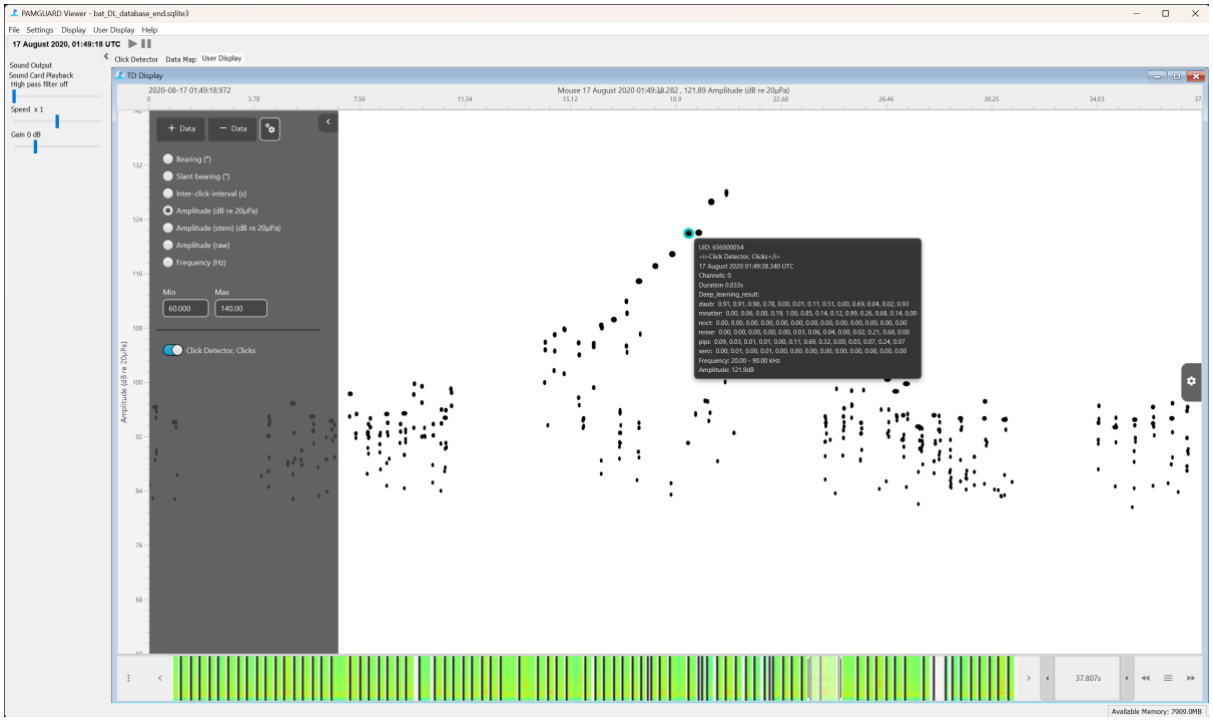


Figure 19B. The time base display after the click data stream has been added. This looks similar to the click display, however, notice that there are many more options for the y-axis and the scroll bar contains a datagram of the call detections loaded into memory.

The ***Y-axis*** menu allows users to plot the data in a number of different ways. Try out the different views, such as ***Amplitude (raw)*** to view waveforms and ***Frequency*** to view spectra. Set the ***Y-axis*** to ***Frequency*** and open the plot settings using the cog button on the top right of the display. In the slide out menu select ***Show Spectrogram***, change the colour scheme to white/grey and set the ***FFT Length*** to 512 and ***Hop Length*** to 32. Set the time axis to show around 1 second (by typing in the scroll bar text box) and you should be able to see the individual spectra of bat calls.

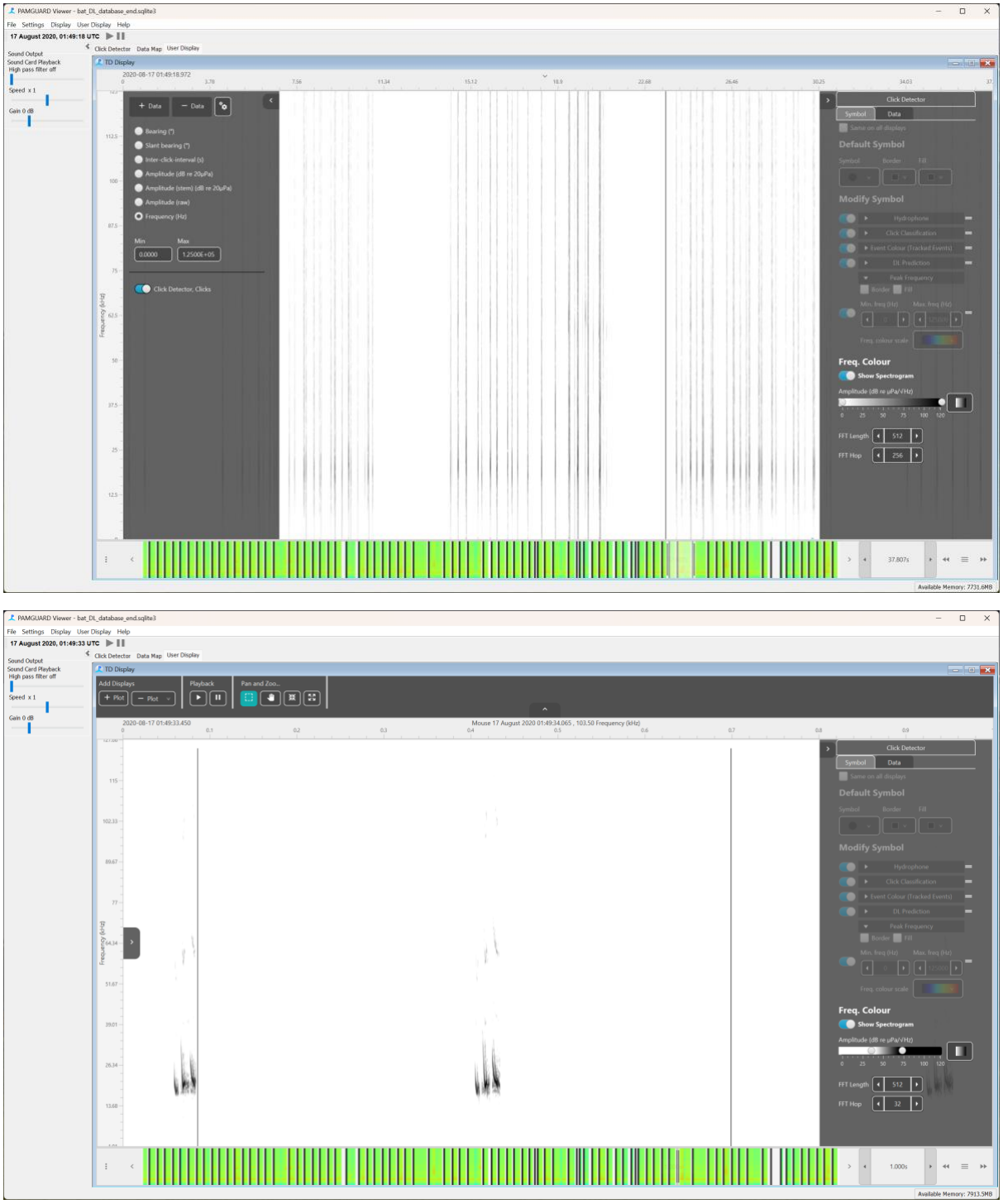


Figure 20B. Viewing the spectrograms of call detections in the time base display. Top shows the zoomed out spectra (not very useful) and the bottom shows more zoomed in spectra where you can see the individual bat calls.

The background colour of the plot can be changed by selecting the cogs button in the ***Display Data*** portion of the left menu pane. This can be useful when visualising some colour schemes for the spectrogram.

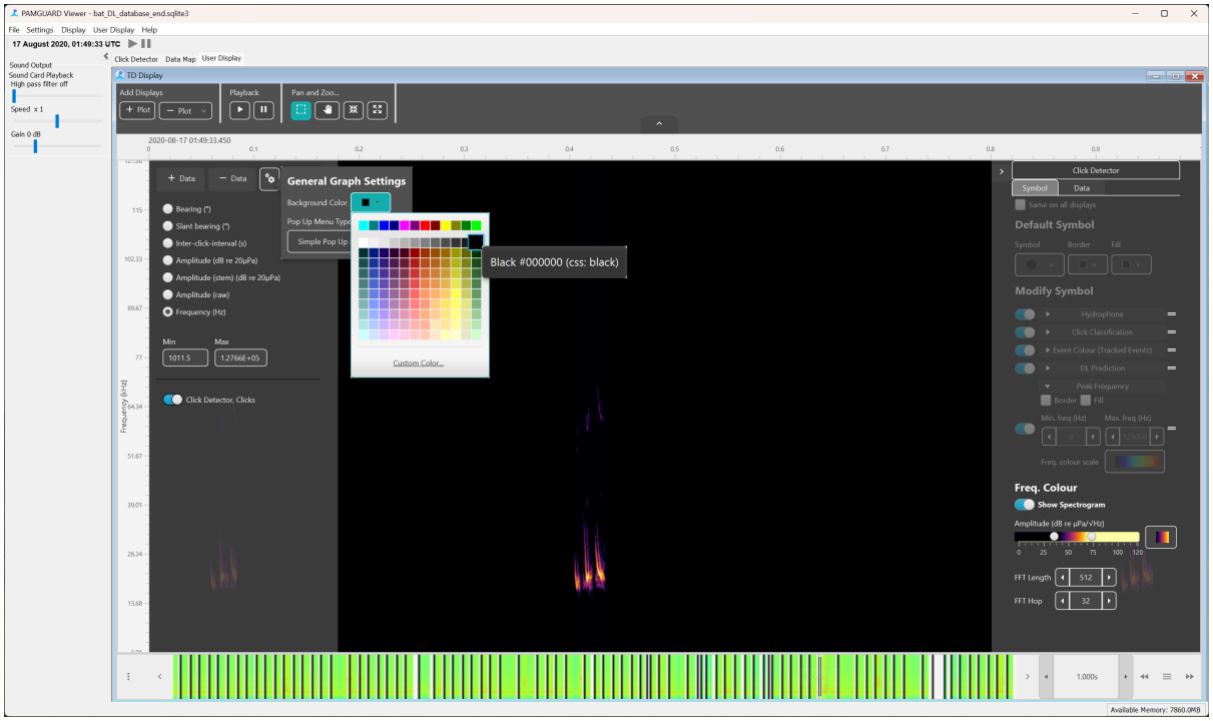


Figure 21B. Changing the background colour of the display.

The Time Base display also implements a powerful symbol manager for colouring detections by specific properties. For example, switch the waveform view and open the right-hand side settings menu again. Deselect ***Show Spectrogram*** and enable the ***Peak Frequency*** symbols under the ***Modify Symbol*** section of the settings pane. Expand ***Peak Frequency*** settings, tick ***Border*** and ***Fill*** to show we want to colour both the symbol border and the fill and select the HSV colour scheme. The click waveforms are now coloured by the peak frequency of each detected click. Change the frequency limits of the colour scheme and the colours of the waveforms will also change.

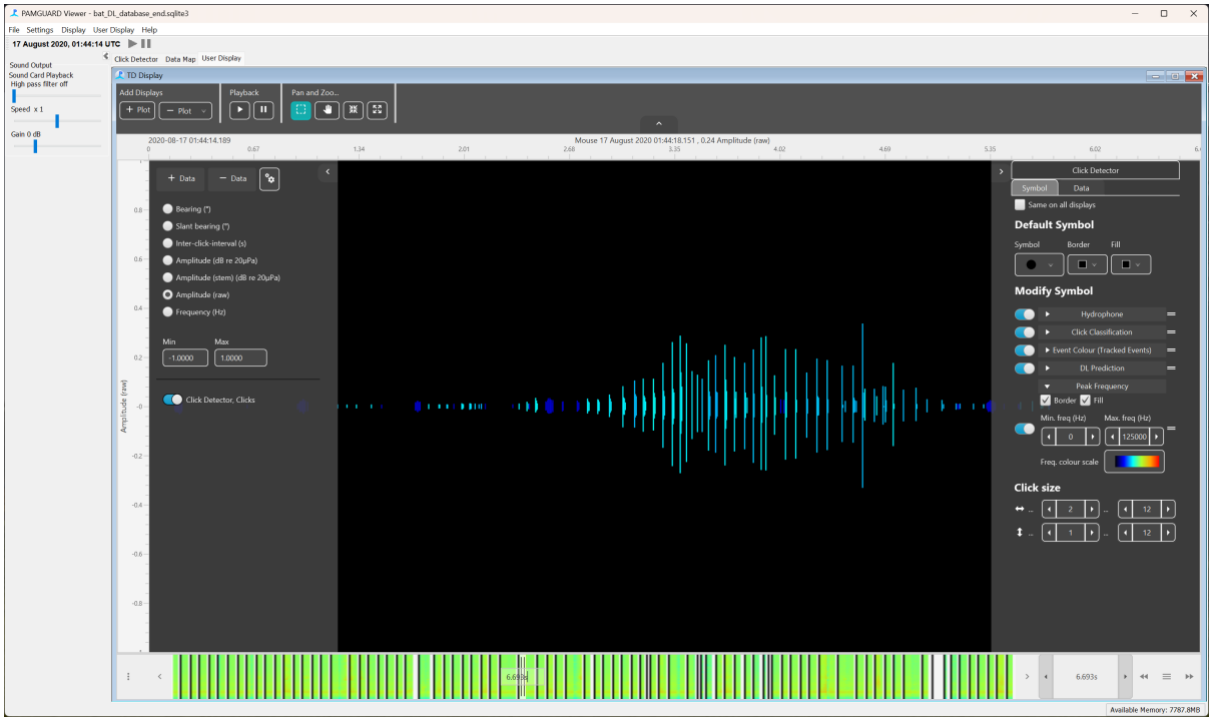


Figure 22B. Waveforms of detected calls coloured by peak frequency.

We can also colour clicks by their deep learning prediction. Select ***DL Prediction*** and expand the settings. Select both the ***Border*** and ***Fill*** boxes and the colour scheme. The call detections can only be coloured by one class of prediction at a time which can be selected in the ***Select Class*** menu.

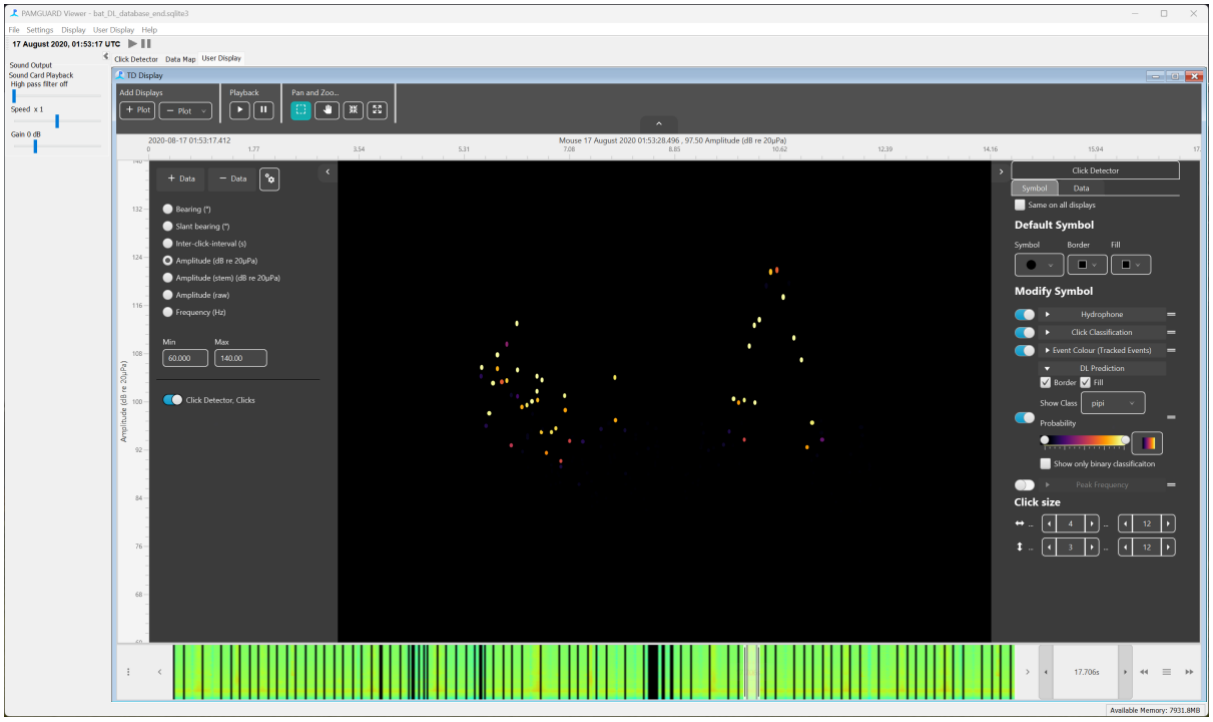


Figure 23B. Clicks plotted with Amplitude (dB) and coloured by the probability that a detection is a call from a Daubenton's bat (daub).

The time base display also allows users to more closely inspect individual data units but this feature is experimental so is not enabled by default. In the left sliding pane select the cog button under ***Display Data***. Select ***Adv. Pop up Menu*** in the ***Pop Up Menu Type*** box. Now when you right click on a click you will be presented with a plot showing the waveform of the detection. You can change the plot to show spectra, wigner or spectrogram data by opening the left hiding pane (top left arrow). The detection can be exported directly to MATLAB, R or .wav file using the ***Export to*** buttons. The exported clips are saved to clipboard and in the user folder.

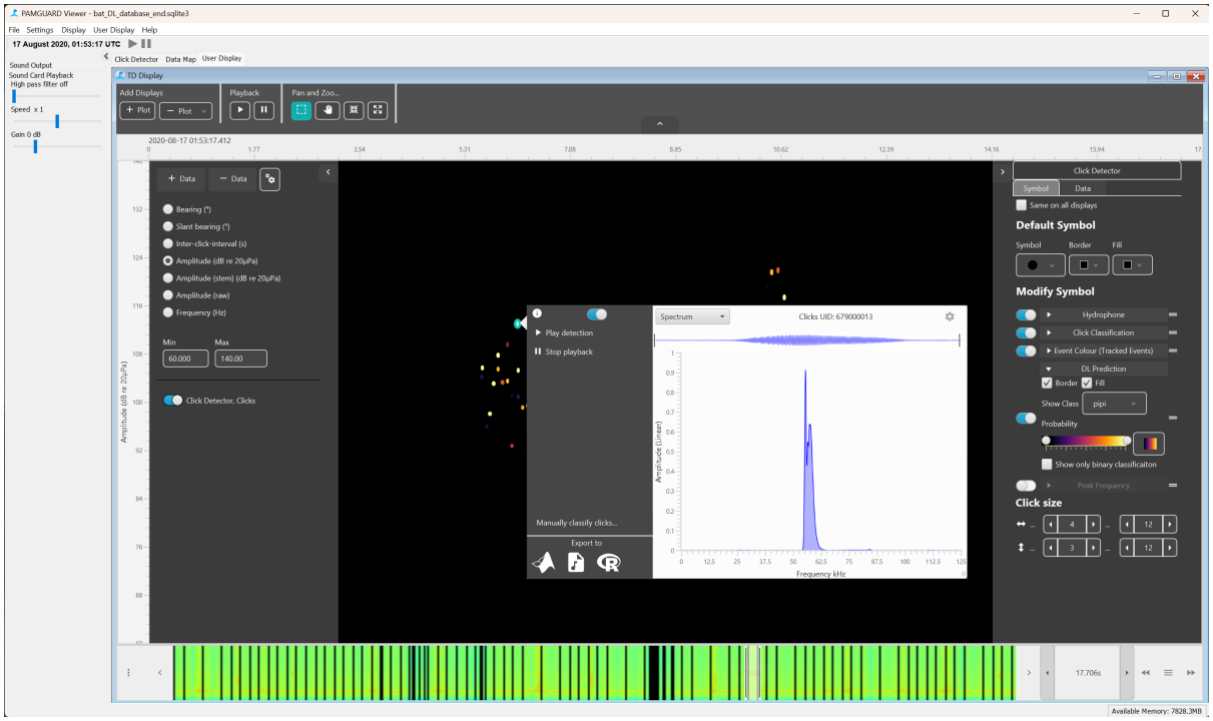


Figure 24B. the detection preview pane showing the spectra of a selected bat call. The call can be exported to MATLAB, R or as a raw wave file using this menu.

Multiple detections can be selected by drawing a box in the time base display. Right clicking on the box will bring up the same detection preview menu but with arrow buttons on the top right allowing users to iterate through the selected call detections. The ***Export to*** buttons will now export all the selected detections.

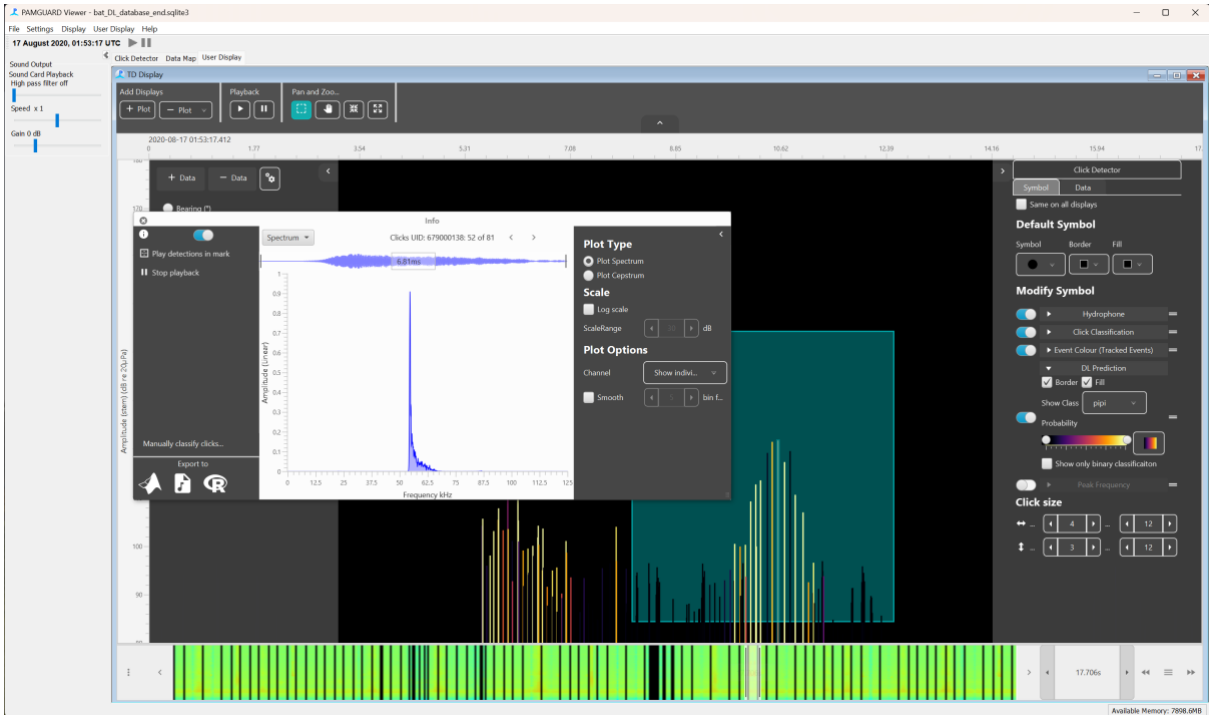


Figure 25B. Multiple calls can be selected.

Play around with the time base display. There are many more features not described here.

Importing bat call detections into MATLAB

Note that we will be using the MATLAB library for this however R has an identical PAMGuard library with the same functions. It can be found at https://github.com/TaikiSan21/PamBinaries

As mentioned in the previous sections, the deep learning classifier does not create new detections unless the input is raw sound data. The classification results from the deep learning classifier are stored in the metadata of the original call detections- in this case these are the detections generated by Click detector module. Although detections can be directly exported to MATLAB using the adv pop up menu this is not practical in many cases e.g., if we wanted to load the whole dataset into MATLAB. Instead, we can use the PAMGuard MATLAB library.

The PAMGuard MATLAB library found in online at **www.pamguard.org/48_MATLABRcode.html.** Download the library and add it to your MATLAB path. It is easy to load specific data streams into MATLAB using the loadPamguardBinaryFile.m and loadPamguardBinaryFolder.m functions. Here we will load in a single binary file of clicks and plot the prediction values calculated by the deep learning classifier.

Go the binary file folder created in the previous section and copy the path to the first file (the file should be called Click_Detector_Click_Detector_Clicks_20200817_010002.pgdf). Loading the file into MATLAB is straightforward - replace the file variable below with the path to your file and run the script.

```
file = '/Click_Detector_Click_Detector_Clicks_20200817_010002.pgdf';
clicks = loadPamguardBinaryFile(file);
```

`clicks` is a structure of all the click detections within the binary file. Note that the `loadPamguardBinaryFile` function works for all type of binary file – so for example, if you selected an LTSA binary file the functions would load a structure of LTSA data.

1x227 struct with 17 fields

Fields	Duration	freqLimits	date	triggerMap	type	flags	angleErrors	duration	nChan	wave	annotations
10	517	[20000,900...	7.3802e...	1	0	0	[]	517	1	517x1 do...	1x1 struct
11	889	[20000,900...	7.3802e...	1	0	0	[]	889	1	889x1 do...	1x1 struct
12	529	[20000,900...	7.3802e...	1	0	0	[]	529	1	529x1 do...	1x1 struct
13	1784	[20000,900...	7.3802e...	1	0	0	[]	1784	1	1784x1 d...	1x1 struct

You can browse the metadata of the click detections in MATLAB's variable explorer. The deep learning data is contained in the annotations field. Annotations in PAMGuard are extra metadata that can be added to a detection (such a click) – different modules tag detections with different annotations. Here we only have one annotation form the deep learning classifier. Open the annotation structure and it contains a list of structures containing predictions, classID, whether the segment passed binary classification and type (which can be ignored for now)

Fields	predictions	classID	isbinary	type
1	[0.5382,5.09...	[0,1,2,3,4...	0	1
2	[0.2279,3.97...	[0,1,2,3,4...	0	1
3	[0.9718,0.02...	[0,1,2,3,4...	1	1

Using the loaded click structure, it is pretty easy to plot the loaded click detections and predictions together.

```
sR = 250000; % the sample rate of the wave data
classindex = 1; % the class to plot.
windowLen = 1250; % the window length used to segment the data in samples
windowHop = 625;  % the hop size used for define segment overla in samples.

clkampdB = zeros(length(clicks),2); % pre allocate for speed
pltpredicitons=[];
for i=1:length(clicks)

    %calcualte the clcik amplitude
    maxclk = max(clicks(i).wave(:,1));
    minclk = min(clicks(i).wave(:,1));

    %get the date number for the clicks
    clkampdB(i,1) = clicks(i).date;

    %the peak to peak amplitude in dB
    clkampdB(i,2) = 20*log10(maxclk-minclk);

    %extract al predicitons to create a single prediciton time series.
    if (~isempty(clicks(i).annotations))

        for j=1:length(clicks(i).annotations)

clkpredictions(j,:)=clicks(i).annotations(1).dlclassification(j).prediction
s;
        end

        % convert the prediciton index into a datenumber for each
        % prediciton.
        classpredicitons(:,1)       =       ((0:length(clkpredictions(:,1))-
1)*windowHop/sR)/60/60/24 +  clicks(i).date;
        %extract the predicitons form the class we are interested in.
        classpredicitons(:,2) = clkpredictions(:,classindex);

        %merge it all together
        pltpredicitons = [pltpredicitons ; classpredicitons];

    end

end
```

```
%plot the data
subplot(2,1,1)
scatter(clkampdB(:,1), clkampdB(:,2), 'filled')
datetick('x', 'KeepLimits')
ylabel('Amplitude (dB)')

subplot(2,1,2)
plot(pltpredicitons(:,1), pltpredicitons(:,2))
xlabel('Time')
ylabel('Prediction')
datetick('x', 'KeepLimits')
```

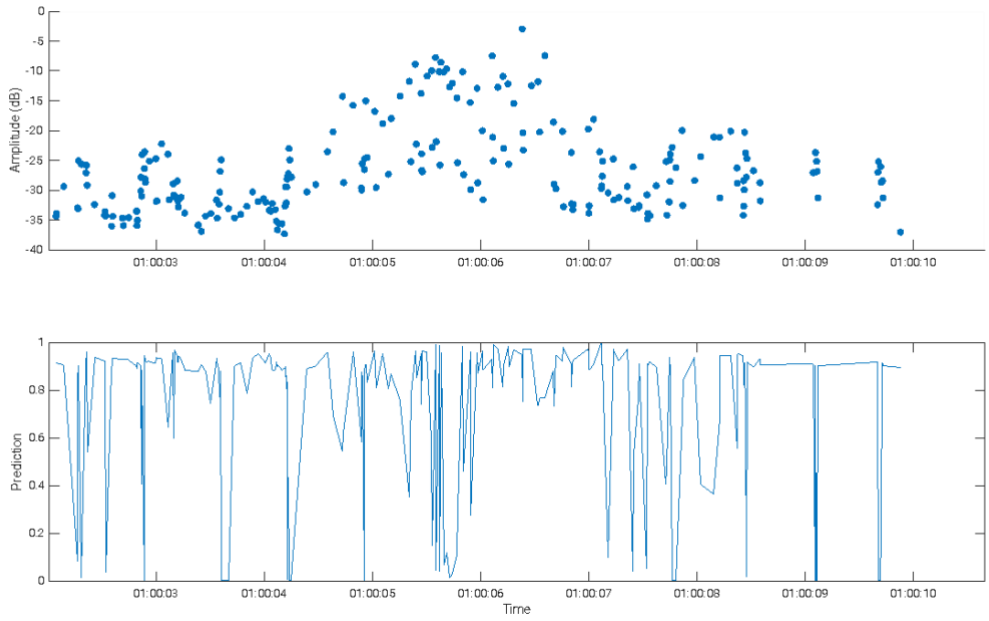


Figure 42. Plotting click amplitudes and deep learning predictions in MATLAB

MATLAB/R challenge – can you colour the scatter plot by the highest prediction for a particular class? Hint – use a similar structure to the previous code ad check out the help files for scatter plot.

Acknowledgements

Thanks to Christian Bergler for developing and sharing the model. You can find out more about this research from;
Brinkløv, S. M. M., Macaulay, J., Bergler, C., Tougaard, J., Beedholm, K., Elmeros, M., & Madsen, P. T. (2023). Open-source workflow approaches to passive acoustic monitoring of bats. Methods in Ecology and Evolution, 14, 1747–1763. https://doi.org/10.1111/2041-210X.14131

Localising Gibbons Using Deep Learning

ARBIMON

The model used in this tutorial was developed and shared by Arbimon. Arbimon is a free ecoacoustic analysis platform empowering scientists and conservationists with an efficient way to upload, store, and analyze mass amounts of ecoacoustic data to derive insights that help better protect biodiversity around the world.

Learn more at arbimon.org

Introduction

One of the advantages of integrating deep learning into PAMGuard is the ability to integrate deep learning classifiers into acoustic workflows. In other words, users can take advantage of both the accuracy and ubiquity of deep learning classifiers and the multitude of acoustic analysis tools that already exist in PAMGuard. This exercise demonstrates this idea by integrating deep learning into a localisation workflow for Gibbons. Compact 4 channel recorders were placed in a rainforest and recorded Gibbon calls. The microphones were arranged in such a way that it is possible to determine angle to the Gibbons which allows researchers to apply spatially explicit capture recapture methods to determine population size. This exercise will demonstrate how to set up the deep learning classifier and downstream localisation module to calculate angles and then extract those data for downstream analysis.

Gibbon call detection

To begin, open a blank PAMGuard configuration file. You will be greeted by the usual PAMGuard blank canvas – first we will add the sound acquisition and deep learning modules to run the Gibbon deep learning classifier.
We are working in air so select ***File->Sound Medium->Air*** then add the Sound Acquisition module by selecting ***File->Add modules -> Sound Processing -> Sound Acquisition.*** Now open the settings dialog by clicking ***Settings -> Sound Acquisition…***. Select ***Audiofile folder or multiple files*** from the ***Data Source Type*** selection menu. Click on ***Select Folder or Files*** and browse to the file that contains Gibbon calls *(../gibbon_tutorial/wav/20240205_230252.wav).* The sample rate should show up as 32000Hz with 4 channels. Everything is now set up, so click ***Ok***. You will then be shown a prompt asking to add more microphones to the array (PAMGuard has detected the 4 channel files and by default only two microphones are present in the array). Click OK to open the array manager (or ***Settings -> Microphone Array…***) to bring up the array dialog. Import the array by selecting ***Import…*** and select the file *../gibbon_tutorial/Cibbon_Array/paf*.

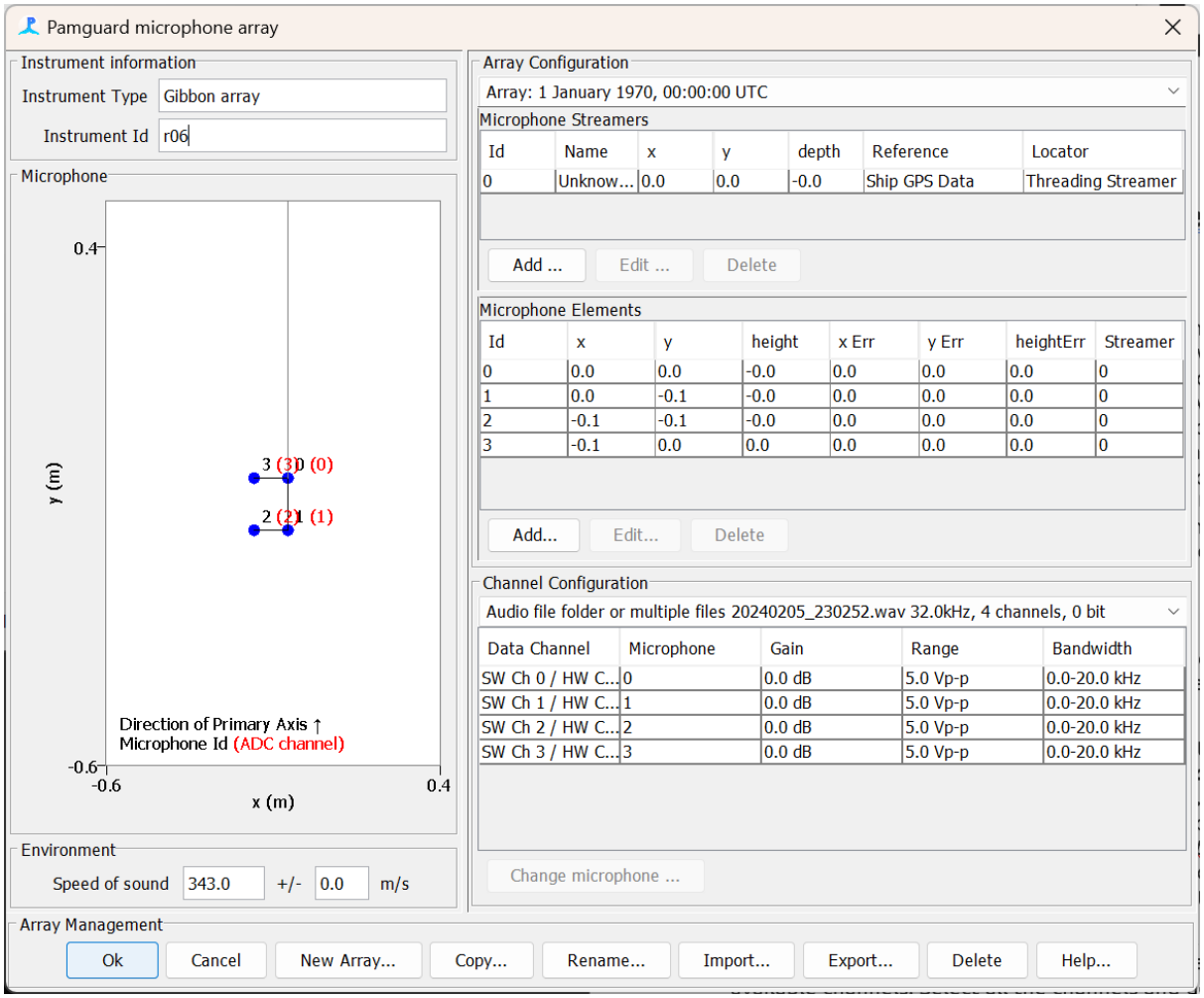


Figure 1C. The microphone array manager is used to define the dimensions of the microphone and speed of sound.

Next add the raw deep learning module ***File->Add modules -> Classifiers -> Deep Learning Classifier***. Open the deep learning module settings. You should see that there are four available channels. Select all the channels and define one group. This means that the deep learning module will only classify data from the first channel in the group but will save data from all the channel for downstream localisation. Next import the gibbon classifier. Select the folder button and navigate the gibbon_model.zip file. This contains the deep learning model and metadata for setting the model up in PAMGuard. Once the model has loaded you should see new option appear – select the ***Use default segment length*** toggle to make sure we will be inputting the correct segment size. Also make sure that that all the output classes are selected under ***Binary Classification Threshold*** and reduce the ***Min prediction threshold*** to 0.7. This means that any time the prediction for any class goes above 0.7 PAMGuard will save a chunk of raw sound data. Click OK to exit the module and save the settings

Note that you may see a warning if you select all classes for binary classification – this is because many models contain a noise class i.e. if a species is not detected then the prediction value for "noise" is high – if all classes are selected then you would continually be detecting something which would result in all data being saved as detections. In this case the Gibbon classifier does not have a noise class - each class defines a probability of a particular type of song – if no Gibbon is detected all outputs are close to zero.

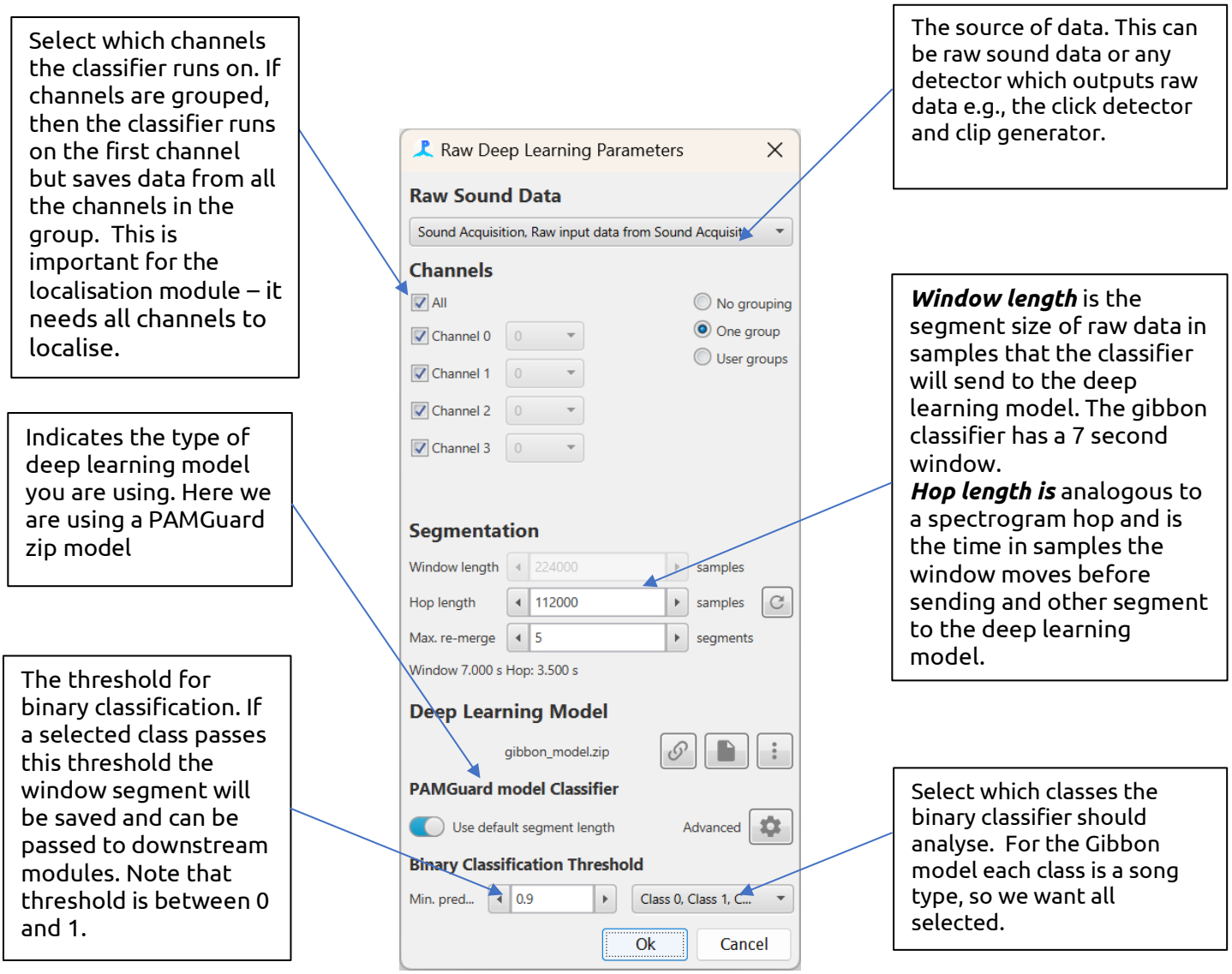


Figure 2C. The deep learning module settings once the Gibbon classifier has been loaded.

The deep learning module is now set up. Next, we need to localise the Gibbon calls. The microphones are spaced close together so they will not be able to determine range but can determine bearing. The best module to use is therefore the bearing localiser. Add the bearing localiser module ***File->Add modules -> Localisers -> Bearing localiser and open*** **it's settings** ***Settings -> Bearing Localiser***. In this instance we are going to use a basic beam forming approach to determine bearing – this offers a good balance between accuracy and computational power. First, we only want to calculate bearings for deep learning detection so in the ***Source*** Tab select ***Deep Learning Classifier, DLClassifier Data***. In the ***Channel Groups*** tab make sure that all channels are selected and in the ***Algorithms*** Tab select ***Basic Frequency Domain Beamformer*** - then open the beam former settings by selecting the *Settings* cog icon. In the BeamOGram settings we will define the angles the beam former will check. We have a planar array which can provide us with full 360° horizontal angles and vertical angles between 0° and 90° which have an up-down ambiguity. i.e. a vertical angle of say 30° can either be pointing +30° or -30° but there is no way to tell form the data. In the ***BeamOGram*** settings therefore set the ***primary angle*** range to -180° and 180° with a 10° step and set the ***secondary angle*** to -90° and 0° with a 10° step. Note that PAMGuard interpolates the results to provide a more accurate angle so the 10° step does not necessarily mean 10° accuracy. Finally set the ***frequency*** range to between 800 and 2500 Hz as this is where Gibbon calls are primarily located, and we don't want other sounds messing up our bearings.

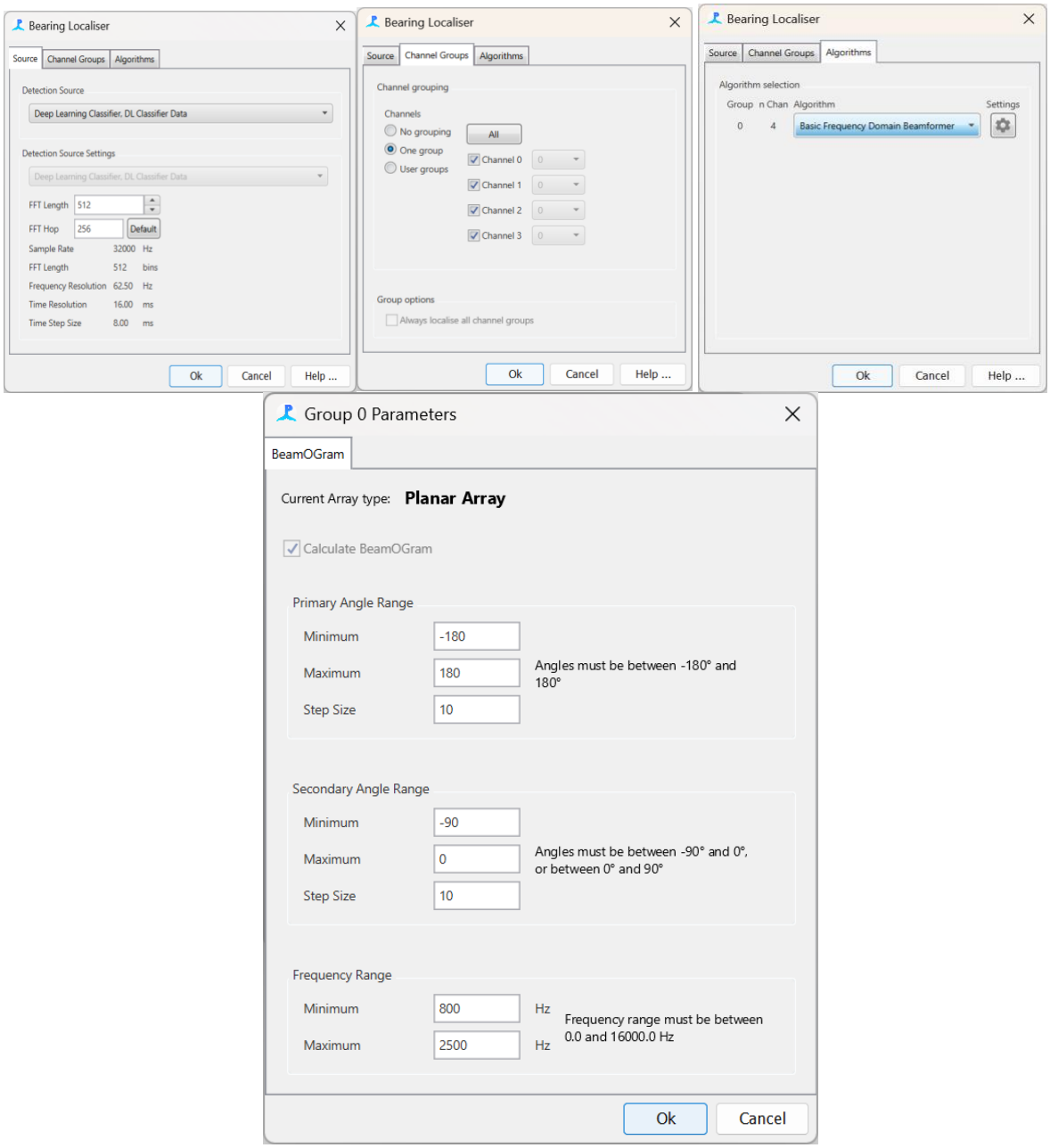


Figure 3C. Settings for the bearing localiser to use some basic beam forming for bearing calculations.

Click OK and close the bearing localiser settings.
Now we have our acoustic workflow but not data management or displays.

To save the data add both the database and binary file storage modules. Go to **File -> Add Modules -> Utilities -> Binary Storage** and then **File -> Add Modules -> Utilities -> Database.** Now we need to tell PAMGuard the location on the computer to save the data. Go to **File -> Binary Store** -> **Storage options…** and press the **Browse** button. Use the folder selection dialog to select or create a new folder. Next go to **File -> Database -> Database Selection…** and in the settings dialog click **Browse…** and create a new SQLite database.

Next, we will add displays. Add the FFT module which will allow us to view some spectrograms. Go to ***File-> Add Modules -> Sound Processing -> FFT (Spectrogram) Engine***

and then open the FFT settings **Settings -> *FFT (Spectrogram) Engine.*** Make sure all the channels are selected and close the settings by clicking ***Ok.***

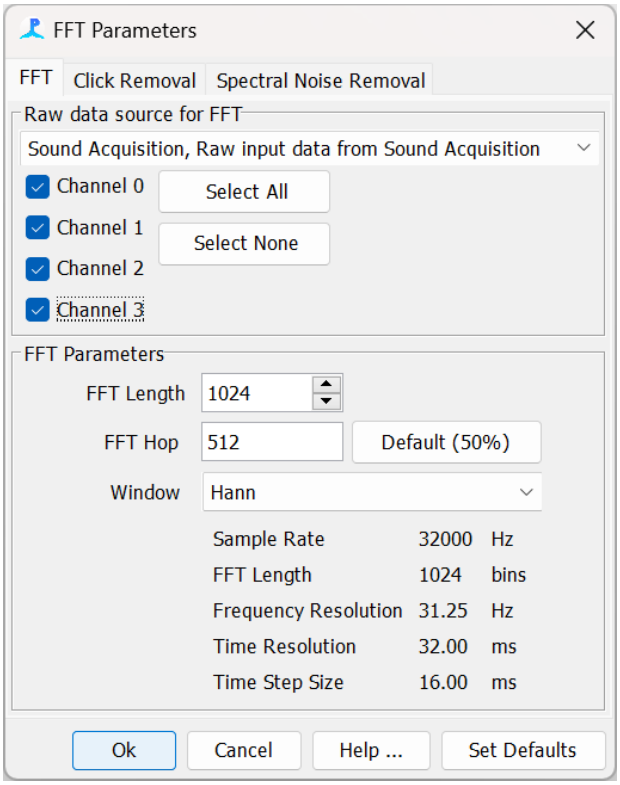


Figure 4C. Settings for the FFT module. The FFT module is used to generate spectrograms to allow use to visualise data.

At this point is a good idea to check the data model ***File-> Show Data Model…*** The data model shows the acoustic workflow – you should be able to see the Sound Acquisition connected to the deep learning module which is then connected to the bearing localiser with the FFT module separate.

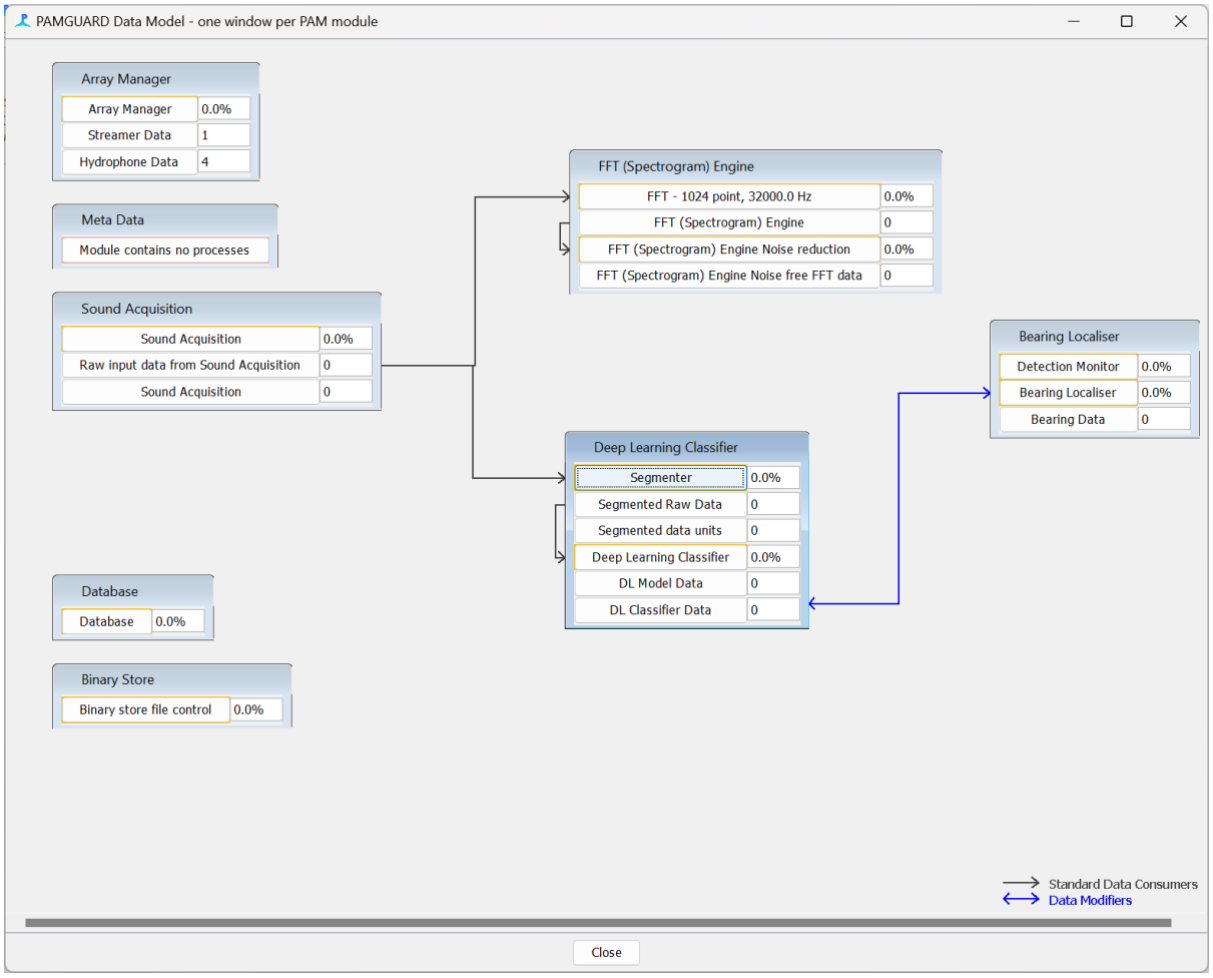


Figure 5C. The data model shows the deep learning module connected to the bearing localiser.

Now we are ready to add displays. Create a new user display tab ***Add Modules -> Displays -> User Display.*** Click on the User Display tab and in the top toolbar select ***User Display -> New Time base data display fx.*** Also add a new bearing display ***User Display-> New Bearing Localiser Display***. Arrange the windows so the time display take about 2/3 of the space.

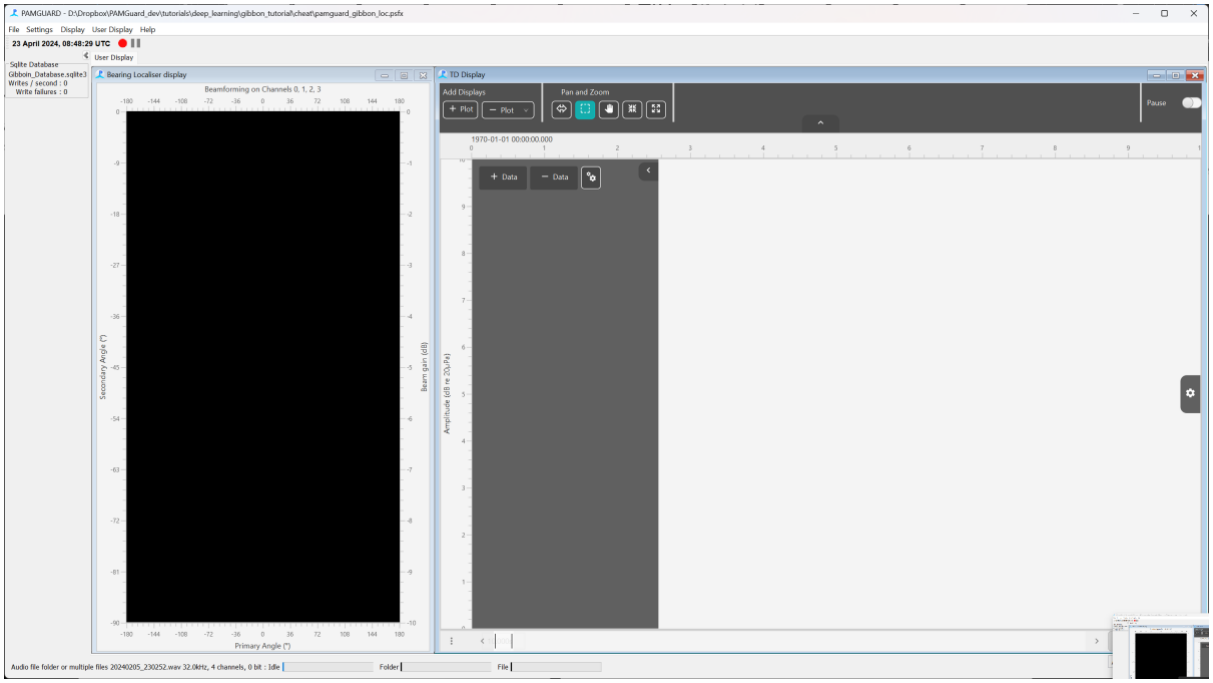


Figure 6C. The time base (right) and bearing display (left) after they have initially been added to the data model.

The Time Base Display allows users to visualise and overlay any data stream in PAMGuard with a time dimension e.g. spectrograms, click detections, whistle detections, deep learning detections, etc. It also allows users to change the y-axis so that data can be visualised in a variety of ways. For example, click detections can be viewed as FFT spectra with a frequency axis or scatter points on an amplitude bearing or bearing axis. Here we will be viewing spectrogram data (which only has one y-axis: frequency) but we need to add it the Time Base Display first. Click the ***+Add*** button and select ***FFT (Spectrogram) Engine***. The display will now have a frequency y-axis ready to display spectrogram data. We also want to show classified deep learning detections on this display (i.e. when the prediction values go above the set threshold and a detection is saved). Click the ***+Data*** button and select ***Deep Learning, Deep Learning Detections.*** The default colour for detections is a black translucent box which doesn't look great on a spectrogram - select the cog on the right of the display to open settings and click expand both setting panes. Set the **Border** and **Fill** colours to be cyan. It's also a good idea to reduce the frequency and amplitude limits on the spectrogram using the associated sliders.
Next add a new plot in the time display by pressing the + button in the top Add Displays menu (note this can be hidden – press the down arrow in the middle of the top axis to show the menu). This will create a new plot underneath the spectrogram. In the new plot on the left settings pane, select the ***+ Add*** button and add the ***Prediction probability, Deep Learning Classifier*** data stream. Now go the top spectrogram plot and press ***+ Add*** button and select ***Deep learning detection, Deep Learning Classifier***
The bearing display has nothing to set up. It shows a heat map of bearings for horizontal and vertical angles. Red means the horizontal/vertical angle combination is likely and blue mean unlikely. The most probably horizontal/vertical localised angle is shown by a white cross.

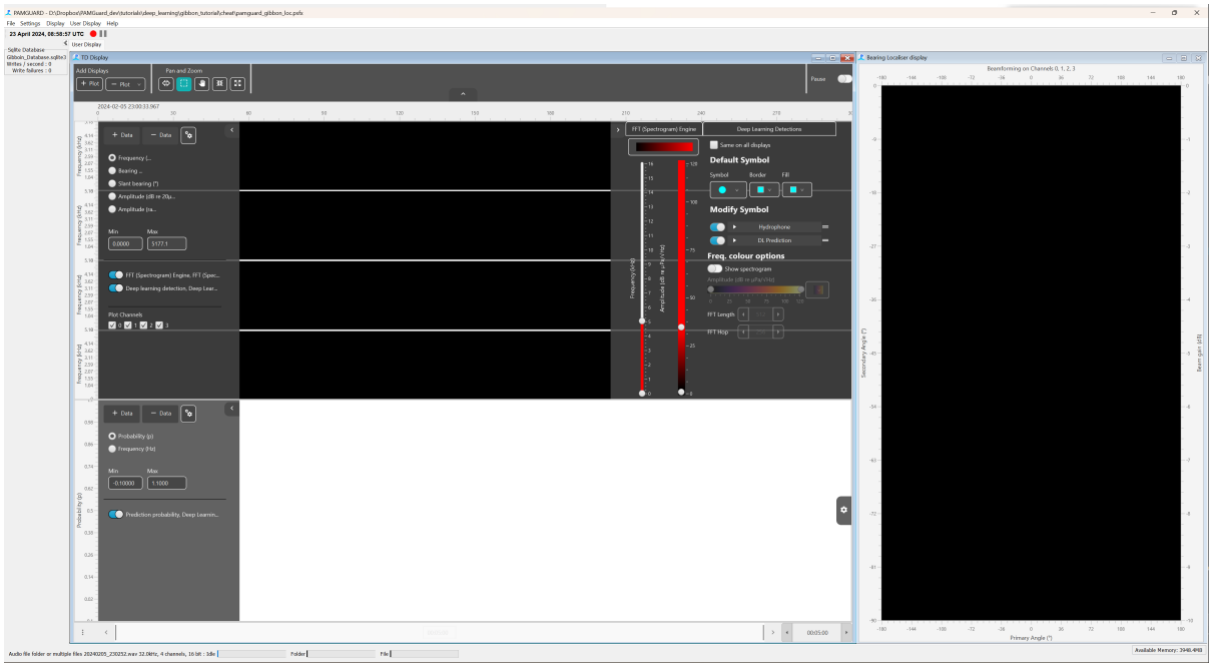


Figure 7C. The time base display settings for the spectrogram and deep learning detections. Change the colour of the detections to cyan to make the easier to see against a red spectrogram – or experiment with your own spectrogram and detection colour combinations!

The last thing to do is to add a Sound output Module. This allow us to listen to the sound but also to control the speed of data analysis which can be useful. Add the Sound Output module by selected ***File->Add modules -> Sound Processing -> Sound Output.*** Next open the settings dialog by selecting ***Settings->Sound Output…*** Select two channels to listen to (most computers have stereo output) and make sure the output sample rate is below 48kHz. Click ***Ok*** to close the settings.

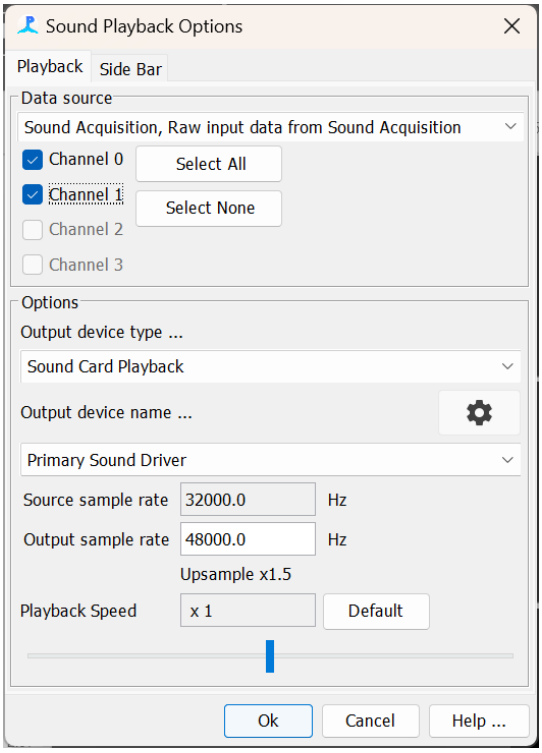


Figure 8C. The Sound Output module settings.

We are enow ready to run everything but first SAVE THE CONFIGUATION. Click the big red button and PAMGuard will start processing. Use the Speed slider on the left of the screen to control processing speed and watch as PAMGuard detects Gibbins and calculates bearings!

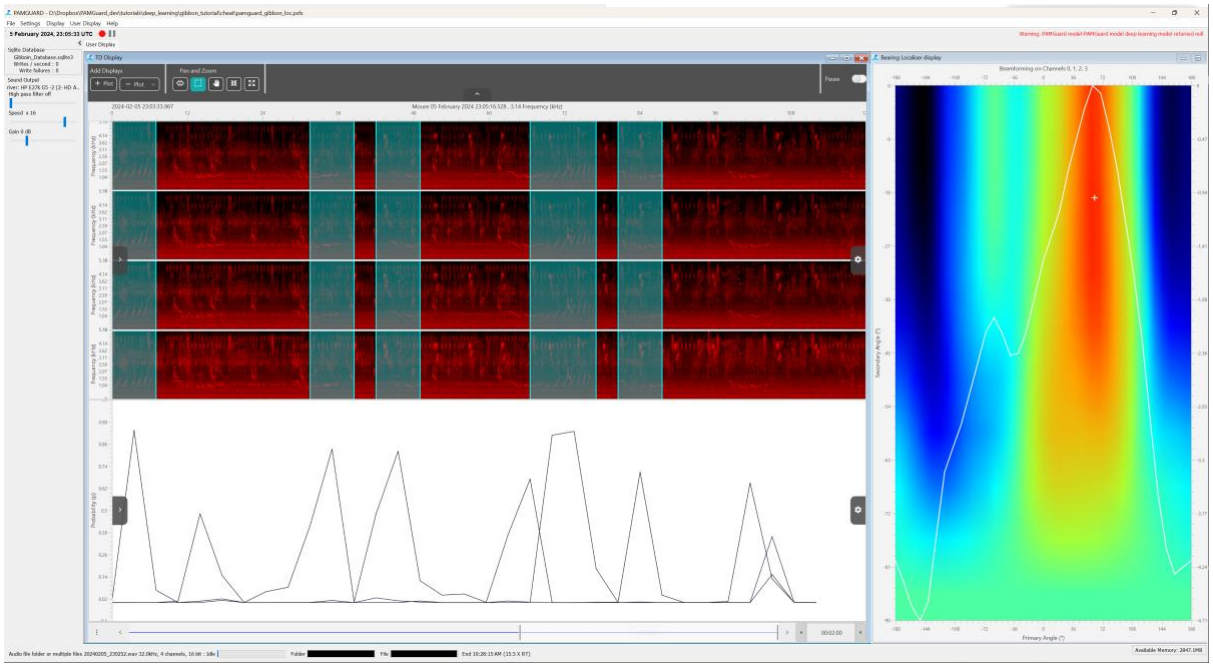


Figure 9C. PAMGuard as it processes the data. The spectrogram for all four channels is show with overlaid detections on the top left. A time series of prediction values from the deep learning model is show on the bottom left and bearing localisations on the right. Note that the bearing localisation pane shows the likelihood of that a horizontal/vertical angle combination are correct based on the received data. The most likely horizontal/vertical angle combination is indicated by a white cross.

Opening the data in PAMGuard viewer model

We will now open PAMGuard viewer mode. Open PAMGuard viewer mode and select the database we created earlier in the exercise. PAMGuard will open to the data map that shows us an overview if the processed data. We have only processed a very small audio file here and so the data map, which is used to visualise long term datasets, only shows a sliver of data. Select this section by right clicking and selecting ***Centre Data Here*** and go to the time display. You should see the detections and time series of prediction similar to when we processed the data.

Note that sometime PAMGuard resets the display to show load zero minutes. If this occurs select the menu icon on the bottom right of the display and how much data should be loaded e.g. 15 minutes.

We also have to set PAMGuard up to use raw files for the displays, allowing us to visualise the spectrograms. Select Sound Acquisition settings ***Settings-> Sound Acquisition*** and, in the ***Offline Files*** Tab, select the folder the Gibbon wav file is in.

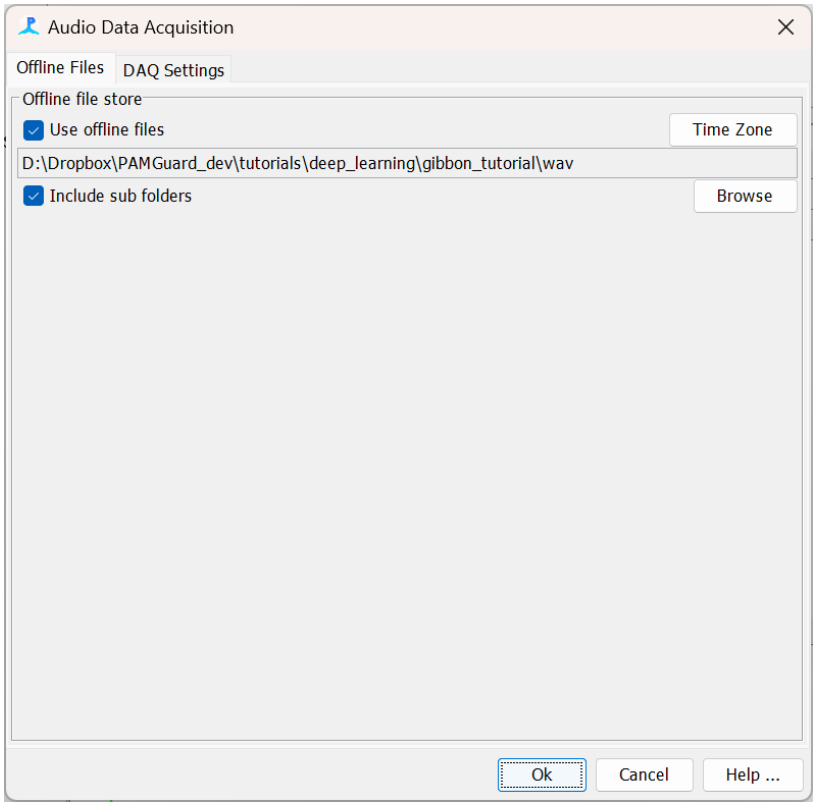


Figure 10C. The sound acquisition module in viewer mode has options to allow the user to select the location of raw data.

The spectrogram should now appear with detection from the deep learning module overlaid as translucent boxes.

The time base display allows closer inspection of detection data – this feature is still experimental and so is not enabled by default. To enable it, select the left settings pane (top left arrow button), click the cog button just under ***Display Data*** and select ***Adv. Pop up menu*** in the ***Pop up menu type*** selection menu. Right click on one of the deep learning detection and select the information icon on the top left of the pop up menu. This brings up the metadata for each detection and in that you will see a field called ***Bearing: Basic BF-Angles*** which shows the horizontal and vertical angles respectively.
This small dataset has been provided as a proof of concept to learn how to set PAMGuard up. If this were a larger dataset then the data map would be used to rapidly navigate to interesting sections of data, showing the user where the deep learning classifier had detected large numbers of detections.

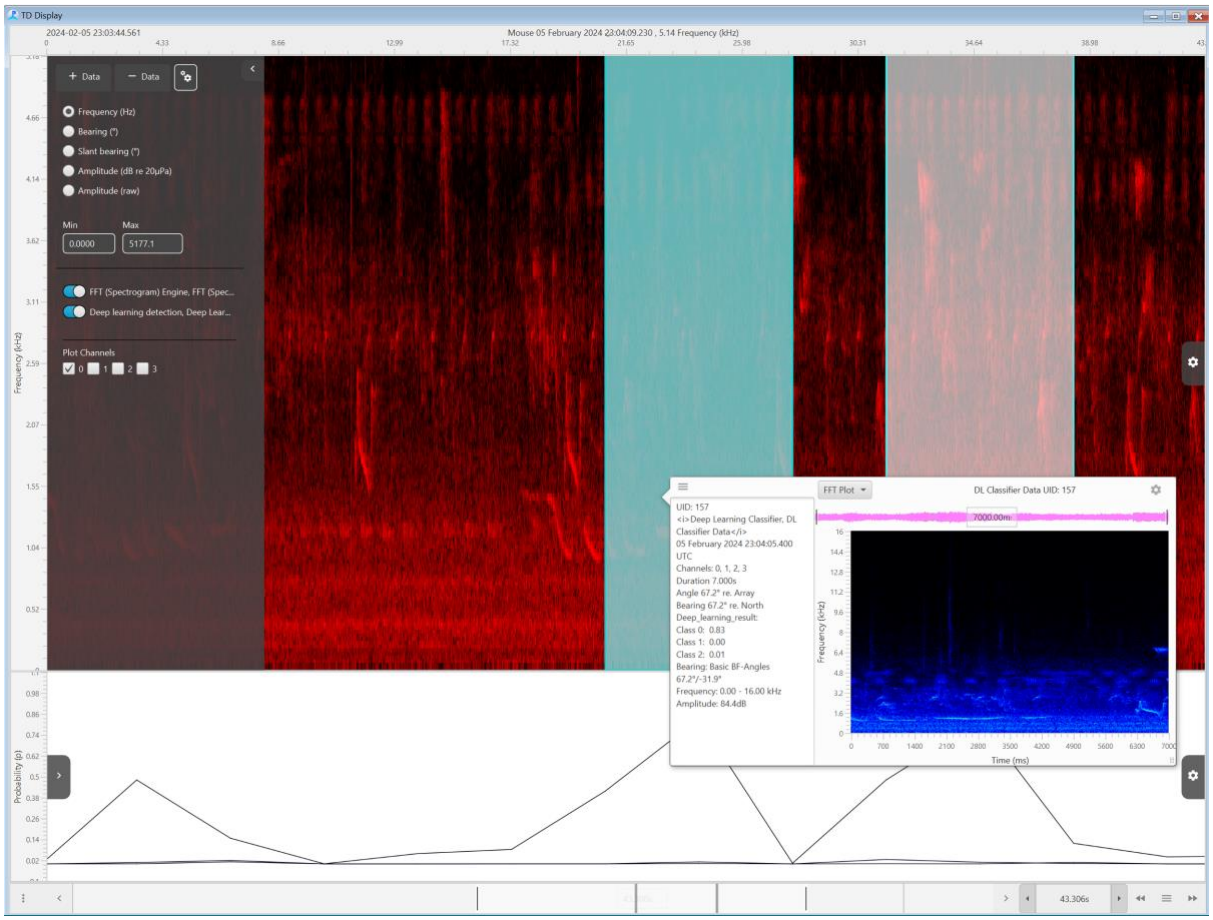


Figure 11C. The time base display allows users to scroll through detections and time series of predications from the deep learning module.

Importing the results into MATLAB

Now that the data are processed, we may want to use the results for some sort of further analysis. PAMGuard has an associated R and MATLAB library that can import data from binary files and databases. Here we will import data in MATLAB and plot a time series of prediction values alongside the bearings of detections. Note this exercise requires the PAMGuard-MATLAB library – if you do not have it download from https://github.com/PAMGuard/PAMGuardMatlab and add it to your MATLAB path. Loading data is simple. Define a variable which is the path to your binary file and then use the function loadPamguardBinaryFolder to load all data within the folder.

```
%path to the folder where pAMGuard binary files are saved
binaryFolder =
'/Users/au671271/Library/CloudStorage/Dropbox/PAMGuard_dev/tutorials/deep_l
earning/gibbon_tutorial/cheat/PAMBinary';

%load up a time series of predictions data from PAMGuard.
predictions = loadPamguardBinaryFolder(binaryFolder,
'Deep_Learning_Classifier_Deep_Learning_Classifier_DL_Model_Data*.pgdf');

%load up the detections.
detections = loadPamguardBinaryFolder(binaryFolder,
'Deep_Learning_Classifier_Deep_Learning_Classifier_DL_Detection*.pgdf');
```

This will load two structure, predictions which is a time series of all predictions and detections which contains raw data from data which passed the defined prediction threshold set in PAMGuard. Both structures contain the default fields for detecitons in PAMGuard and additional field added by the deep leanring module. The important fields to notice are 1) the *predicitons* field for the predictions values which is a list of output values for each class and *wave* and *annotations.bearing* which contains the raw wave data and loclaised bearings for each detection respectively.

Fields	millis	channelMap	UID	startSample	sampleDuration	freqLimits	millisDuration	date	type	isbinary	predictions	annotations
1	1.7072e...	15	6000001	108801	224000	[0,16000]	7000	7.3929e...	0	0	[5.5165e-04,...	[]
2	1.7072e...	15	6000002	220801	224000	[0,16000]	7000	7.3929e...	0	0	[0.1453,4.56...	[]
3	1.7072e...	15	6000003	332801	224000	[0,16000]	7000	7.3929e...	0	0	[0.0589,0.00...	[]
4	1.7072e...	15	6000004	444801	224000	[0,16000]	7000	7.3929e...	0	0	[4.6617e-04,...	[]
5	1.7072e...	15	6000005	556801	224000	[0,16000]	7000	7.3929e...	0	0	[5.6813e-05,...	[]
6	1.7072e...	15	6000006	668801	224000	[0,16000]	7000	7.3929e...	0	0	[1.2514e-04,...	[]

Fields	millis	channelMap	UID	startSample	sampleDuration	freqLimits	millisDuration	date	nChan	nSamps	scale	wave	annotations
1	.7072e...	15	148	1340801	224000	[0,16000]	7000	7.3929e...	4	224000	2.0264e-...	*224000x...*	*1x1 struct*
2	.7072e...	15	148	1340801	224000	[0,16000]	7000	7.3929e...	4	224000	2.0264e-...	*224000x...*	*1x1 struct*
3	.7072e...	15	157	2348801	224000	[0,16000]	7000	7.3929e...	4	224000	1.7061e-...	*224000x...*	*1x1 struct*
4	.7072e...	15	157	2348801	224000	[0,16000]	7000	7.3929e...	4	224000	1.7061e-...	*224000x...*	*1x1 struct*

Next, we will plot the prediction data. First, we extract the prediction values from the predictions structure and put them into a matrix which can then be plotted using the plot function.

```
%Convert the prediction struct into an easier to plot matrix
predictionsm = zeros(length(predictions), nclass+1); % pre allocate array
for speed
for i=1:length(predictionsm)
    predictionsm(i,1) = predictions(i).date;
    %add prediction values to the rest of the matrix
    for j=1:nclass
    predictionsm(i,j+1) = predictions(i).predictions(j); % class one
    end
end
```

Now we can plot the data using

```
hold on
for i=1:nclass
plot(predictionsm(:,1), predictionsm(:,i+1), 'LineWidth', 1.5)
end
xlabel('Time')
ylabel('Prediction')
datetick('x', 'KeepLimits');
legend({' Nomascus leucogenys Common Song', ...
    'Nomascus leucogenys Alternative Song 2', ...
    'Nomascus leucogenys Alternative Song'})
set(gca, 'FontSize', 14)
hold off
```

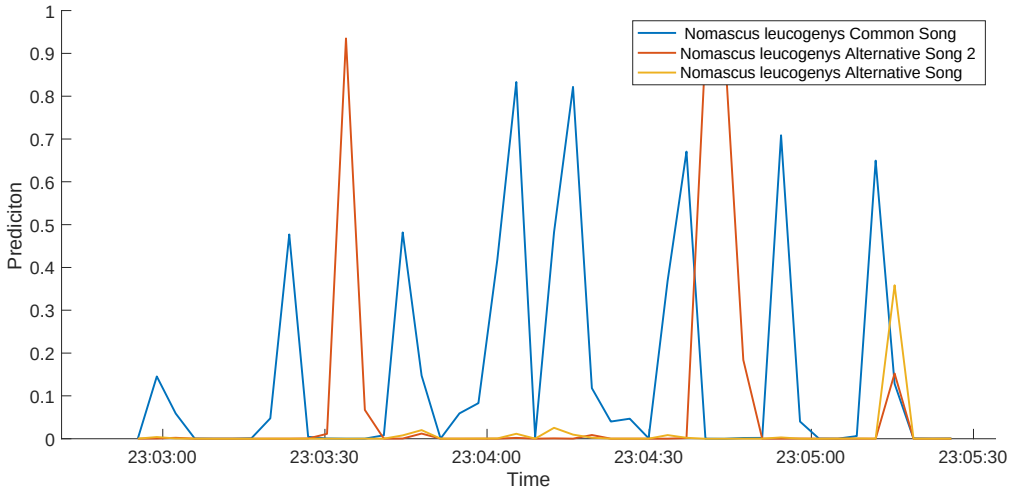


Figure 12C. Plot of a time series of predictions for each song type created in MATLAB.

We may also want to plot the bearing angles of the detections. In a similar way we extract the data from the structure into an easier-to-work-with matrix.

```
%get bearings - matrix of date, horizontal angle, vertical angle
for i=1:length(detections)
    detectionm(i,1) = detections(i).date;
    detectionm(i,[2,3]) = detections(i).annotations.bearing.angles;
end
```

Finally, we plot the bearings as a scatter plot alongside the predictions.

```
hold on
scatter(detectionm(:,1),rad2deg(detectionm(:,2)), 'filled');
scatter(detectionm(:,1),rad2deg(detectionm(:,3)), 'filled');
hold off
ylabel('Angle (degrees)')
legend({' Horizontal angle (degrees)', 'Vertical Angle (degrees)'},...
    'Location', 'southeast')
datetick('x', 'KeepLimits');
ylim([-180, 180])
set(gca, 'FontSize', 14)
```

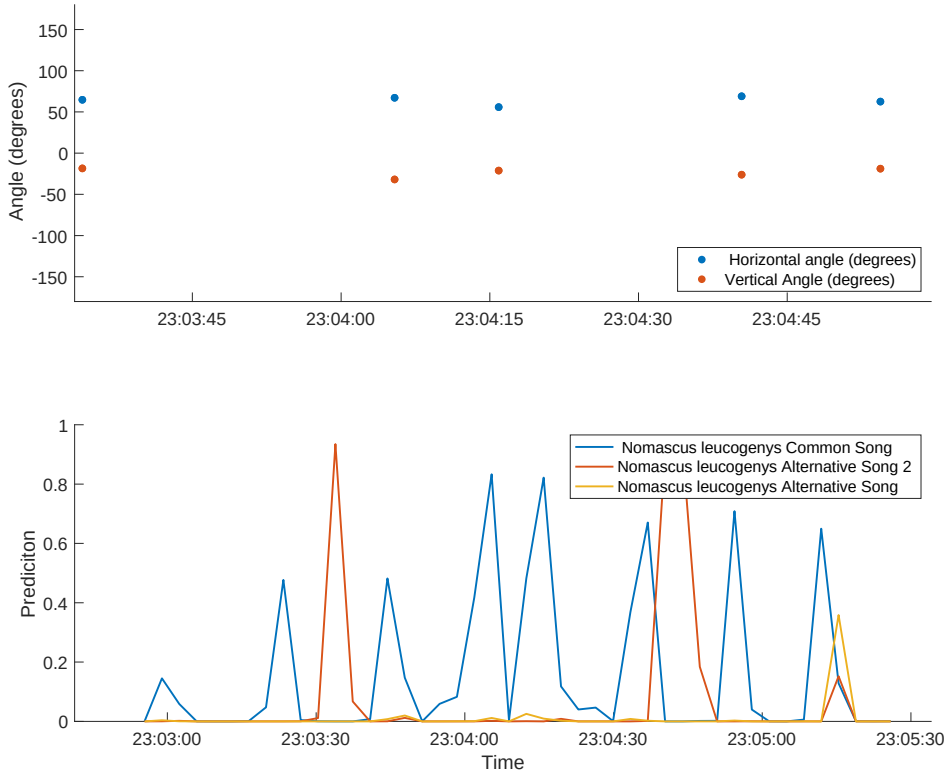


Figure 13C. Plot of bearings to each detection (i.e. when prediction values exceed a threshold) alongside a time series of predictions for each song type plotted using MATLAB. The bearing values are roughly in the same direction which makes sense!

The full script is.

```
clear % start off with a clean workspace

%% Import deep learning detections
%path to the folder where pAMGuard binary files are saved
binaryFolder =
'/Users/au671271/Library/CloudStorage/Dropbox/PAMGuard_dev/tutorials/deep_l
earning/gibbon_tutorial/cheat/PAMBinary';
nclass = 3; %define how many output class there are for predicitons


%load up a time series of predictions data from PAMGuard.
predictions = loadPamguardBinaryFolder(binaryFolder,
'Deep_Learning_Classifier_Deep_Learning_Classifier_DL_Model_Data*.pgdf');

%load up the detections.
detections = loadPamguardBinaryFolder(binaryFolder,
'Deep_Learning_Classifier_Deep_Learning_Classifier_DL_Detection*.pgdf');

%% Plot the predicitons and detections

%Convert the prediciton struc into an easier to plot matrix
predictionsm = zeros(length(predictions), nclass+1); % pre allocate array
for speed
for i=1:length(predictions)
```

```
    predictionsm(i,1) = predictions(i).date;
    %add prediction values to the rest of the matrix
    for j=1:nclass
    predictionsm(i,j+1) = predictions(i).predictions(j); % class one
    end
end

%get bearings - matrix of date, horizontal angle, vertical angle
for i=1:length(detections)
    detectionm(i,1) = detections(i).date;
    detectionm(i,[2,3]) = detections(i).annotations.bearing.angles;
end

tiledlayout(2,1)
nexttile
hold on
scatter(detectionm(:,1),rad2deg(detectionm(:,2)), 'filled');
scatter(detectionm(:,1),rad2deg(detectionm(:,3)), 'filled');
hold off
ylabel('Angle (degrees)')
legend({' Horizontal angle (degrees)', 'Vertical Angle (degrees)'},...
    'Location', 'southeast')
datetick('x', 'KeepLimits');
ylim([-180, 180])
set(gca, 'FontSize', 14)

nexttile
hold on
%plot a line for each class
for i=1:nclass
plot(predictionsm(:,1), predictionsm(:,i+1), 'LineWidth', 1.5)
end
%add axis albels
xlabel('Time')
ylabel('Prediciton')
datetick('x', 'KeepLimits');
legend({' Nomascus leucogenys Common Song', ...
    'Nomascus leucogenys Alternative Song 2', ...
    'Nomascus leucogenys Alternative Song'})
set(gca, 'FontSize', 14)
hold off
```

Importing the results into R

R also has a package to import PAMGuard binary data. Make sure you have downloaded the *PamBinaries* package.

-TODO

Acknowledgements

Marie Roche and her group for sharing the right whale classifier

Christian Bergler for sharing his bat classifier.

Elizabeth Ferguson at www.oceanscienceanalytics.com for being a first beta tester.

Arbimon (www.arbimon.org) for sharing their Gibbon model.