Echofilter Documentation

Release 1.1.1

Scott C. Lowe

Nov 18, 2022

CONTENTS:

CHAPTER

ONE

USAGE GUIDE

Echofilter is an application for segmenting an echogram. It takes as its input an *Echoview* .EV file, and produces as its output several lines and regions:

- *turbulence* (*entrained air*) line
- *bottom (seafloor) line*
- *surface line*
- *nearfield line*
- *passive data* regions
- *bad data regions for entirely removed periods of time, in the form of boxes covering the entire vertical depth
- *bad data regions for localised anomalies, in the form of polygonal contour patches

Echofilter uses a *machine learning model* to complete this task. The machine learning model was trained on *upfacing stationary* and *downfacing mobile* data provided by Fundy Ocean Research Centre for Energy (FORCE.).

Disclaimer

- The *model* is only confirmed to work reliably with *upfacing* data recorded at the same location and with the same instrumentation as the data it was trained on. It is expected to work well on a wider range of data, but this has not been confirmed. Even on data similar to the *training data*, the *model* is not perfect and it is recommended that a human analyst manually inspects the results it generates to confirm they are correct.
- * *Bad data regions* are particularly challenging for the *model* to generate. Consequently, the bad data region outputs are not reliable and should be considered experimental. By default, these outputs are disabled.
- Integration with *Echoview* was tested for Echoview 10 and 11.

1.1 Installation

1.1.1 Installing as an executable file

Echofilter is distributed as an *executable binary file* for Windows. All dependencies are packaged as part of the distribution.

1. Download the zip file containing the echofilter executable as follows:
    a. Go to the releases tab of the echofilter repository.
    b. Select the release to download. It is recommended to use the latest version, with the highest release number.
    c. Click on the file named echofilter-executable-M.N.P.zip, where M.N.P is replaced with the version number, to download it. For example: echofilter-executable-1.1.1.zip

       Alternatively, the zipped executables can be downloaded from a mirror on GDrive.
2. Unzip the zip file, and put the directory contained within it wherever you like on your Windows machine. It is recommended to put it as an "echofilter" directory within your Programs folder, or similar. (You may need the WinZip or 7z application to unzip the .zip file.)
3. In File Explorer,
    a. navigate to the echofilter directory you unzipped. This directory contains a file named *echofilter.exe*.
    b. left click on the echofilter directory containing the *echofilter.exe* file
    c. Shift+Right click on the echofilter directory
    d. select "Copy as path"
    e. paste the path into a text editor of your choice (e.g. Notepad)
4. Find and open the Command Prompt application (your Windows machine comes with this pre-installed). That application is also called cmd.exe. It will open a window containing a terminal within which there is a command prompt where you can type to enter commands.
5. Within the Command Prompt window (the terminal window):
    a. type: `"cd "` (without quote marks, with a trailing space) and then right click and select paste in order to paste the full path to the echofilter directory, which you copied to the clipboard in step 3d.
    b. press enter to run this command, which will change the current working directory of the terminal to the echofilter directory.
    c. type: `echofilter --version`
    d. press enter to run this command
    e. you will see the version number of echofilter printed in the terminal window
    f. type: `echofilter --help`
    g. press enter to run this command
    h. you will see the help for echofilter printed in the terminal window
6. (Optional) So that you can just run *echofilter* without having to change directory (using the `cd` command) to the directory containing *echofilter.exe*, or use the full path to *echofilter.exe*, every time you want to use it, it is useful to add echofilter to the PATH environment variable. This step is entirely optional and for your convenience only. The PATH environment variable tells the terminal where it should look for executable commands.
    a. Instructions for how to do this depend on your version of Windows and can be found here: https://www.computerhope.com/issues/ch000549.htm.

   b. An environment variable named PATH (case-insensitive) should already exist.

   c. If this is a string, you need to edit the string and prepend the path from 3e, plus a semicolon. For example, change the current value of `C:\Program Files;C:\Winnt;C:\Winnt\System32` into `C:\Program Files\echofilter;C:\Program Files;C:\Winnt;C:\Winnt\System32`

   d. If this is a list of strings (without semicolons), add your path from 3e (e.g. `C:\Program Files\echofilter`) to the list

7. You can now run *echofilter* on some files, by using the echofilter command in the terminal. *Example commands* are shown below.

1.2 Command line interface primer

In this section, we provide some pointers for users new to using the command prompt.

1.2.1 Spaces in file names

Running commands on files with spaces in their file names is problematic. This is because spaces are used to separate arguments from each other, so for instance:

```
command-name some path with spaces
```

is actually running the command `command-name` with four arguments: `some`, `path`, `with`, and `spaces`.

You can run commands on paths containing spaces by encapsulating the path in quotes (either single, `'`, or double `"` quotes), so it becomes a single string. For instance:

```
command-name "some path with spaces"
```

In the long run, you may find it easier to change your directory structure to not include any spaces in any of the names of directories used for the data.

1.2.2 Trailing backslash

The backslash (\) character is an escape character, used to give alternative meanings to symbols with special meanings. For example, the quote characters `"` and `'` indicate the start or end of a string but can be escaped to obtain a literal quote character.

On Windows, \ is also used to denote directories. This overloads the \ symbol with multiple meanings. For this reason, you should not include a trailing \ when specifying directory inputs. Otherwise, if you provide the path in quotes, an input of `"some\path\"` will not be registered correctly, and will include a literal `"` character, with the end of the string implicitly indicated by the end of the input. Instead, you should use `"some\path"`.

Alternatively, you could escape the backslash character to ensure it is a literal backslash with `"some\path\\"`, or use a forward slash with `"some/path/"` since *echofilter* also understands forward slashes as a directory separator.

1.2.3 Argument types

Commands at the command prompt can take arguments. There are a couple of types of arguments:

- mandatory, positional arguments
- optional arguments
  - shorthand arguments which start with a single hyphen (`-v`)
  - longhand arguments which start with two hyphens (`--verbose`)

For *echofilter*, the only positional argument is the path to the file(s) or directory(ies) to process.

Arguments take differing numbers of parameters. For *echofilter* the positional argument (files to process) must have at least one entry and can contain as many as you like.

Arguments which take zero parameters are sometimes called flags, such as the flag `--skip-existing`

Shorthand arguments can be given together, such as `-vvfsn`, which is the same as all of `--verbose --verbose --force --skip --dry-run`.

In the help documentation, arguments which require at least one value to be supplied have text in capitals after the argument, such as `--suffix-var SUFFIX_VAR`. Arguments which have synonyms are listed together in one entry, such as `--skip-existing`, `--skip`, `-s`; and `--output-dir OUTPUT_DIR`, `-o OUTPUT_DIR`. Arguments where a variable is optional have it shown in square brackets, such as `--cache-csv [CSV_DIR]`. Arguments which accept a variable number of values are shown such as `--extension SEARCH_EXTENSION [SEARCH_EXTENSION ...]`. Arguments whose value can only take one of a set number of options are shown in curly brackets, such as `--facing {downward,upward,auto}`.

1.2.4 Breaking up long lines

To increase readability, long lines for commands at the command prompt (or in scripts) can be broken up into multiple lines by using a continuation character. Writing the continuation character at the very end of a line indicates that the new line character which follows it should be ignored, and both lines should be treated together as if they were one line.

On Linux, the line continuation character is \ (backslash).

```
cp "path/to/source/file_with_a_very_very_long_filename" \
    "path/to/destination/location/"
```

On Windows, the line continuation character depends on the command prompt being used.

In the Windows command prompt (cmd.exe) application, which is used to run Windows batch (.bat) files, the line continuation character is ^ (caret).

```
copy "path\to\source\file_with_a_very_very_long_filename" ^
    "path\to\destination\location\"
```

In the Windows command prompt, when you are separating out arguments you must make sure you include at least one space at the start of the second line. There must be spaces between arguments for them to be registered as distinct arguments, and for some reason only having a space before the ^ on the preceding line does not work.

In the Windows PowerShell application, the line continuation character is ` (backtick).

```
copy "path\to\source\file_with_a_very_very_long_filename" `
    "path\to\destination\location\"
```

Please note that, in all cases, the line continuation character must be the very final character on the line. If there is whitespace after the continuation character, that will stop the line continuation character from actually merging the lines together. In that case, the two lines will be executed as separate commands (which may result in an error, or if not will not result in the expected behaviour).

1.3 Quick Start

Note that it is recommended to close *Echoview* before running *echofilter* so that *echofilter* can run its own Echoview instance in the background. After *echofilter* has started processing the files, you can open Echoview again for your own use without interrupting *echofilter*.

1.3.1 Recommended first time usage

The first time you use *echofilter*, you should run it in simulation mode (by supplying the `--dry-run` argument) beforehand so you can see what it will do:

```
echofilter some/path/to/directory_or_file --dry-run
```

The path you supply to *echofilter* can be an absolute path, or a relative path. If it is a relative path, it should be relative to the current working directory of the command prompt.

1.3.2 Example commands

Review echofilter's documentation help within the terminal:

```
echofilter --help
```

Specifying a single file to process, using an absolute path:

```
echofilter "C:\Users\Bob\Desktop\MinasPassage\2020\20200801_SiteA.EV"
```

Specifying a single file to process, using a path relative to the current directory of the command prompt:

```
echofilter "MinasPassage\2020\20200801_SiteA.EV"
```

Simulating processing of a single file, using a relative path:

```
echofilter "MinasPassage\2020\20200801_SiteA.EV" --dry-run
```

Specifying a directory of *upfacing stationary* data to process, and excluding the bottom line from the output:

```
echofilter "C:\Users\Bob\OneDrive\Desktop\MinasPassage\2020" --no-bottom-line
```

Specifying a directory of *downfacing mobile* data to process, and excluding the surface line from the output:

```
echofilter "C:\Users\Bob\Documents\MobileSurveyData\Survey11" --no-surface-line
```

Processing the same directory after some files were added to it, skipping files already processed:

```
echofilter "C:\Users\Bob\Documents\MobileSurveyData\Survey11" --no-surface --skip
```

Processing the same directory after some files were added to it, overwriting files already processed:

```
echofilter "C:\Users\Bob\Documents\MobileSurveyData\Survey11" --no-surface --force
```

Ignoring all *bad data regions* (default), using `^` to break up the long command into multiple lines for Windows cmd:

```
echofilter "path/to/file_or_directory" ^
    --minimum-removed-length -1 ^
    --minimum-patch-area -1
```

Including *bad data regions* in the *EVR* output:

```
echofilter "path/to/file_or_directory" ^
    --minimum-removed-length 10 ^
    --minimum-patch-area 25
```

Keep line predictions during *passive* periods (default is to linearly interpolate lines during passive data collection):

```
echofilter "path/to/file_or_directory" --lines-during-passive predict
```

Specifying file and variable suffix, and line colours and thickness:

```
echofilter "path/to/file_or_directory" ^
    --suffix "_echofilter-model" ^
    --color-surface "green" --thickness-surface 4 ^
    --color-nearfield "red" --thickness-nearfield 3
```

Processing a file with more output messages displayed in the terminal:

```
echofilter "path/to/file_or_directory" --verbose
```

Processing a file and sending the output to a log file instead of the terminal:

```
echofilter "path/to/file_or_directory" -v > path/to/log_file.txt 2>&1
```

1.3.3 Config file

You may find that you are setting some parameters every time you call echofilter, to consistently tweak the input or output processing settings in the same way. If this is the case, you can save these arguments to a configuration file, and pass the configuration file to echofilter instead.

For example, if you have a file named `"echofilter_params.cfg"` with the following contents:

Listing 1: echofilter_params.cfg

```
--suffix "_echofilter-model"
--color-surface "green"
--thickness-surface 4
--color-nearfield "red"
--thickness-nearfield 3
```

then you can call echofilter with this configuration file as follows:

```
echofilter "file_or_dir" --config "path/to/echofilter_params.cfg"
```

and it will use the parameters specified in your config file. The format of the parameters is the same as they would be on the command prompt, except in the config file each parameter must be on its own line.

The parameters in the config file also can be added to, or even overridden, at the command prompt. For example:

```
echofilter "file_or_dir" --config "path/to/echofilter_params.cfg" --suffix "_test"
```

will use the `--suffix "_test"` argument from the command prompt instead of the value set in the file `"echofilter_params.cfg"`, but will still use the other parameters as per the config file.

If you have several different workflows or protocols which you need to use, you can create multiple config files corresponding to each of these workflows and choose which one to use with the `--config` argument.

Common configuration options which you want to always be enabled can be set in a special default config file in your home directory named `".echofilter"`. The path to your homedirectory, and hence to the default config file, depends on your operating system. On Windows it is typically `"C:\Users\USERNAME\.echofilter"`, whilst on Linux it is typically `"/home/USERNAME/.echofilter"`, where `"USERNAME"` is replaced with your username. If it exists, the the default config file is always loaded everytime you run echofilter.

If a config file is manually provided with the `--config` argument, any parameters set in the manually provided config file override those in the default config file ("~/.echofilter).

With the default verbosity settings, at the start of the inference routine echofilter outputs the set of parameters it is using, and the source for each of these parameters (command line, manual config file, default config file, or program defaults).

You can read more about the syntax for the configuration files here.

1.3.4 Argument documentation

Echofilter has a large number of customisation options. The complete list of argument options available to the user can be seen in the *CLI Reference*, or by consulting the help for *echofilter*. The help documentation is output to the terminal when you run the command `echofilter --help`.

1.3.5 Actions

The main *echofilter* action is to perform *inference* on a file or collection of files. However, certain arguments trigger different actions.

help

Show *echofilter* documentation and all possible arguments.

```
echofilter --help
```

version

Show program's version number.

```
echofilter --version
```

list checkpoints

Show the available model checkpoints and exit.

```
echofilter --list-checkpoints
```

list colours

List the available (main) colour options for lines. The palette can be viewed at https://matplotlib.org/gallery/color/named_colors.html

```
echofilter --list-colors
```

List all available colour options (very long list) including the XKCD colour palette of 954 colours, which can be viewed at https://xkcd.com/color/rgb/

```
echofilter --list-colors full
```

1.4 Inference operations

In this section, we describe the *inference* process, its outputs and inputs. Inference is the process of generating predictions from the *model*, and is the principal functionality of *echofilter*.

1.4.1 Processing overview

This is an overview of how files are processed in the *inference* pipeline.

First, the setup:

- If a directory input was given, determine list of files to process.
- Download the model *checkpoint*, if necessary.
- Load the *model* from the *checkpoint* into memory.
- If any file to process is an *EV file*, open *Echoview*.
- If it was not already open, hide the Echoview window.

After the *model* is loaded from its checkpoint, each file is processed in turn. The processing time for an individual file scales linearly with the number of *pings* in the file (twice as many pings = twice as long to process).

Each file is processed in the following steps:

- If the input is an *EV file*, export the *Sv* data to *CSV* format.
  - By default, the *Sv* data is taken from `"Fileset1: Sv pings T1"`.
  - Unless `--cache-csv` is provided, the *CSV file* is output to a temporary file, which is deleted after the *CSV file* is imported.
- Import the *Sv* data from the *CSV file*. (If the input was a *CSV file*, this is the input; if the input was an *EV file* this is the *CSV file* generated from the *EV file* in the preceding step.)
- Rescale the height of the *Sv* input to have the number of pixels expected by the *model*.
- Automatically determine whether the *echosounder* recording is *upfacing* or *downfacing*, based on the order of the Depths data in the *CSV file*.
  - If the orientation was manually specified, issue a warning if it does not match the detected orientation.
  - Reflect the data in the Depth dimension if it is *upfacing*, so that the shallowest *samples* always occur first, and deepest last.
- Normalise the distribution of the *Sv* intensities to match that expected by the *model*.
- Split the input data into segments
  - Detect temporal discontinuities between *pings*.
  - Split the input *Sv* data into segments such that each segment contains contiguous *pings*.
- Pass the each segment of the input through the *model* to generate output probabilities.
- Crop the depth dimension down to zoom in on the most salient data.
  - If *upfacing*, crop the top off the echogram to show only 2m above the shallowest estimated *surface line* depth.
  - If *downfacing*, crop the bottom off the echogram only 2m below the deepest estimated *bottom line* depth.

  - If more than 35% of the echogram's height (threshold value set with `--autocrop-threshold`) was cropped away, pass the cropped *Sv* data through the *model* to get better predictions based on the zoomed in data.

- Line boundary probabilities are converted into output depths.
  - The boundary probabilities at each pixel are integrated to make a cumulative probability distribution across depth, $p(\text{depth} > \text{boundary location})$.
  - The output boundary depth is estimated as the depth at which the cumulative probability distribution first exceeds 50%.
- Bottom, surface, and turbulence lines are output to *EVL* files.
  - Note: there is no EVL file for the *nearfield line* since it is at a constant depth as provided by the user and not generated by the *model*.
- Regions are generated:
  - Regions are collated if there is a small gap between consecutive *passive data* or *bad data regions*.
  - Regions which are too small (fewer than 10 pings for rectangles) are dropped.
  - All regions are written to a single *EVR* file.
- If the input was an *EV file*, the lines and regions are imported into the *EV file*, and a *nearfield line* is added.

1.4.2 Simulating processing

To see which files will be processed by a command and what the output will be, run *echofilter* with the `--dry-run` argument.

1.4.3 Input

Echofilter can process two types of file as its input: .EV files and .CSV files. The *EV file* input is more user-friendly, but requires the Windows operating system, and a fully operational *Echoview* application (i.e. with an Echoview dongle). The *CSV file* format can be processed without Echoview, but must be generated in advance from the .EV file on a system with Echoview. The *CSV files* must contain raw *Sv* data (without thresholding or masking) and in the format produced by exporting *Sv* data from Echoview. These raw *CSV files* can be exported using the utility *ev2csv*, which is provided as a separate executable in the *echofilter* package.

If the input path is a directory, all files in the directory are processed. By default, all subdirectories are recursively processed; this behaviour can be disabled with the `--no-recursive-dir-search` argument. All files in the directory (and subdirectories) with an appropriate file extension will be processed. By default, files with a .CSV or .EV file extension (case insensitive) which will be processed. The file extensions to include can be set with the `--extension` argument.

Multiple input files or directories can also be specified (each separated by a space).

By default, when processing an *EV file*, the *Sv* data is taken from the `"Fileset1: Sv pings T1"` variable. This can be changed with the `--variable-name` argument.

1.4.4 Loading model

The *model* used to process the data is loaded from a *checkpoint* file. The executable *echofilter.exe* comes with its default model checkpoint bundled as part of the release. Aside from this, the first time a particular model is used, the checkpoint file will be downloaded over the internet. The checkpoint file will be cached on your system and will not need to be downloaded again unless you clear your cache.

Multiple models are available to select from. These can be shown by running the command `echofilter --list-checkpoints`. The default model will be highlighted in the output. In general, it is recommended to use the default checkpoint. See *Model checkpoints* below for more details.

When running *echofilter* for *inference*, the checkpoint can be specified with the `--checkpoint` argument.

If you wish to use a custom model which is not built in to *echofilter*, specify a path to the checkpoint file using the `--checkpoint` argument.

1.4.5 Output

Output files

For each input file, *echofilter* produces the following output files:

<input>.bottom.evl An Echoview line file containing the depth of the *bottom line*.

<input>.regions.evr An Echoview region file containing spatiotemporal definitions of *passive* recording rectangle regions, *bad data* full-vertical depth rectangle regions, and *bad data* anomaly polygonal (contour) regions.

<input>.surface.evl An Echoview line file containing the depth of the *surface line*.

<input>.turbulence.evl An Echoview line file containing the depth of the *turbulence line*.

where <input> is the path to an input file, stripped of its file extension. There is no *EVL* file for the *nearfield line*, since it is a virtual line of fixed depth added to the *EV file* during the *Importing outputs into EV file* step.

By default, the output files are located in the same directory as the file being processed. The output directory can be changed with the `--output-dir` argument, and a user-defined suffix can be added to the output file names using the `--suffix` argument.

If the output files already exist, by default *echofilter* will stop running and raise an error. If you want to overwrite output files which already exist, supply the `--overwrite-files` argument. If you want to skip inputs whose output files all already exist, supply the `--skip` argument. Note: if both `--skip` and `--overwrite-files` are supplied, inputs whose outputs all exist will be skipped and those inputs for which only some of the outputs exist will have existing outputs overwritten.

Specific outputs can be dropped by supplying the corresponding argument `--no-bottom-line`, `--no-surface-line`, or `--no-turbulence-line` respectively. To drop particular types of region entirely from the *EVR* output, use `--minimum-passive-length -1`, `--minimum-removed-length -1`, or `--minimum-patch-area -1` respectively. By default, *bad data* regions (rectangles and contours) are not included in the *EVR* file. To include these, set `--minimum-removed-length` and `--minimum-patch-area` to non-negative values.

The lines written to the *EVL* files are the raw output from the model and do not include any offset.

Importing outputs into EV file

If the input file is an Echoview *EV file*, by default *echofilter* will import the output files into the *EV file* and save the *EV file* (overwriting the original *EV file*). The behaviour can be disabled by supplying the `--no-ev-import` argument.

All lines will be imported twice: once at the original depth and a second time with an offset included. This offset ensures the exclusion of data biased by the acoustic deadzone, and provides a margin of safety at the bottom depth of the *entrained air*. The offset moves the *surface* and *turbulence* lines downwards (deeper), and the *bottom line* upwards (shallower). The default offset is 1m for all three lines, and can be set using the `--offset` argument. A different offset can be used for each line by providing the `--offset-bottom`, `--offset-surface`, and `--offset-turbulence` arguments.

The names of the objects imported into the *EV file* have the suffix `"_echofilter"` appended to them, to indicate the source of the line/region. However, if the `--suffix` argument was provided, that suffix is used instead. A custom suffix for the variable names within the EV file can be specified using the `--suffix-var` argument.

If the variable name to be used for a line is already in use, the default behaviour is to append the current datetime to the new variable name. To instead overwrite existing line variables, supply the `--overwrite-ev-lines` argument. Note that existing regions will not be overwritten (only lines).

By default, a *nearfield line* is also added to the *EV file* at a fixed range of 1.7m from the *transducer* position. The *nearfield distance* can be changed as appropriate for the *echosounder* in use by setting the `--nearfield` parameter.

The colour and thickness of the lines can be customised using the `--color-surface`, `--thickness-surface` (etc) arguments. See `echofilter --list-colors` to see the list of supported colour names.

1.5 Pre-trained models

The currently available model checkpoints can be seen by running the command:

```
echofilter --list-checkpoints
```

All current checkpoints were trained on data acquired by FORCE.

1.5.1 Training Datasets

Stationary

data collection bottom-mounted *stationary*, autonomous

orientation uplooking

echosounder 120 kHz Simrad WBAT

locations

- FORCE tidal power demonstration site, Minas Passage
  - 45°21'47.34"N 64°25'38.94"W
  - December 2017 through November 2018
- SMEC, Grand Passage
  - 44°15'49.80"N 66°20'12.60"W
  - December 2019 through January 2020

organization FORCE

Mobile

data collection vessel-based 24-hour transect surveys

orientation downlooking

echosounder 120 kHz Simrad EK80

locations

- FORCE tidal power demonstration site, Minas Passage
  - 45°21'57.58"N 64°25'50.97"W
  - May 2016 through October 2018

organization FORCE

1.5.2 Model checkpoints

The architecture used for all current models is a U-Net with a backbone of 6 EfficientNet blocks in each direction (encoding and decoding). There are horizontal skip connections between compression and expansion blocks at the same spatial scale and a latent space of 32 channels throughout the network. The depth dimension of the input is halved (doubled) after each block, whilst the time dimension is halved (doubled) every other block.

Details for notable model checkpoints are provided below.

conditional_mobile-stationary2_effunet6x2-1_lc32_v2.2

- Trained on both *upfacing stationary* and *downfacing mobile* data.
- Jaccard Index of **96.84%** on *downfacing mobile* and **94.51%** on *upfacing stationary validation* data.
- Default model checkpoint.

conditional_mobile-stationary2_effunet6x2-1_lc32_v2.1

- Trained on both *upfacing stationary* and *downfacing mobile* data.
- Jaccard Index of 96.8% on *downfacing mobile* and 94.4% on *upfacing stationary validation* data.

conditional_mobile-stationary2_effunet6x2-1_lc32_v2.0

- Trained on both *upfacing stationary* and *downfacing mobile* data.
- Jaccard Index of 96.62% on *downfacing mobile* and 94.29% on *upfacing stationary validation* data.
- *Sample* outputs on *upfacing stationary* data were thoroughly verified via manual inspection by trained analysts.

stationary2_effunet6x2-1_lc32_v2.1

- Trained on *upfacing stationary* data only.
- Jaccard Index of 94.4% on *upfacing stationary validation* data.

stationary2_effunet6x2-1_lc32_v2.0

- Trained on *upfacing stationary* data only.
- Jaccard Index of 94.41% on *upfacing stationary validation* data.
- *Sample* outputs thoroughly were thoroughly verified via manual inspection by trained analysts.

mobile_effunet6x2-1_lc32_v1.0

- Trained on *downfacing mobile* data only.

1.6 Citing Echofilter

For technical details about how the Echofilter model was trained, and our findings about its empirical results, please consult our companion paper:

> SC Lowe, LP McGarry, J Douglas, J Newport, S Oore, C Whidden, DJ Hasselman (2022). Echofilter: A Deep Learning Segmention Model Improves the Automation, Standardization, and Timeliness for Post-Processing Echosounder Data in Tidal Energy Streams. *Front. Mar. Sci.*, **9**, 1–21. doi: 10.3389/fmars.2022.867857.

If you use Echofilter for your research, we would be grateful if you could cite this paper in any resulting publications.

For your convenience, we provide a copy of this citation in bibtex format.

You can browse papers which utilise Echofilter here.

1.7 Issues

1.7.1 Known issues

There is a memory leak somewhere in *echofilter*. Consequently, its memory usage will slowly rise while it is in use. When processing a very large number of files, you may eventually run out of memory. In this case, you must close the Command Window (to release the memory). You can then restart *echofilter* from where it was up to, or run the same command with the `--skip` argument, to process the rest of the files.

1.7.2 Troubleshooting

- If you run out of memory after processing a single file, consider closing other programs to free up some memory. If this does not help, report the issue.
- If you run out of memory when part way through processing a large number of files, restart the process by running the same command with the `--skip` argument. See the known issues section above.
- If you have a problem using a *checkpoint* for the first time:
  - check your internet connection
  - check that you have at least 100MB of hard-drive space available to download the new checkpoint
  - if you have an error saying the checkpoint was not recognised, check the spelling of the checkpoint name.
- If you receive error messages about writing or loading *CSV files* automatically generated from *EV files*, check that sufficient hard-drive space is available.
- If you experience problems with operations which occur inside *Echoview*, please re-run the code but manually open Echoview before running *echofilter*. This will leave the Echoview window open and you will be able to read the error message within Echoview.

1.7.3 Reporting an issue

If you experience a problem with *echofilter*, please report it by creating a new issue on our repository if possible, or otherwise by emailing scottclowe@gmail.com.

Please include:

- Which version of echofilter which you are using. This is found by running the command `echofilter --version`.
- The operating system you are using. On Windows 10, system information information can be found by going to `Start > Settings > System > About`. Instructions for other Windows versions can be found here.
- If you are using Echoview integration, your Echoview version number (which can be found by going to `Help > About` in Echoview), and whether you have and are using an Echoview HASP USB dongle.
- What you expected to happen.
- What actually happened.
- All steps/details necessary to reproduce the issue.
- Any error messages which were produced.

1.8 Glossary

Active data Data collected while the *echosounder* is emitting sonar pulses ("*pings*") at regular intervals. This is the normal operating mode for data in this project.

Algorithm A finite sequence of well-defined, unambiguous, computer-implementable operations.

Bad data regions Regions of data which must be excluded from analysis in their entirety. Bad data regions identified by *echofilter* come in two forms: rectangular regions covering the full depth-extend of the echogram for a period of time, and polygonal or contour regions encompassing a localised area.

Bottom line A line separating the seafloor from the *water column*.

Checkpoint A checkpoint file defines the weights for a particular *neural network model*.

Conditional model A *model* which outputs conditional probabilities. In the context of an *echofilter* model, the conditional probabilities are $p(x|\text{upfacing})$ and $p(x|\text{downfacing})$, where x is any of the *model* output types; conditional models are necessarily hybrid models.

CSV A comma-separated values file. The *Sv* data can be exported into this format by *Echoview*.

Dataset A collection of data *samples*. In this project, the datasets are *Sv* recordings from multiple surveys.

Downfacing The orientation of an *echosounder* when it is located at the surface and records from the *water column* below it.

Echofilter A software package for defining the placement of the boundary lines and regions required to post-process *echosounder* data. The topic of this usage guide.

echofilter.exe The compiled *echofilter* program which can be run on a Windows machine.

Echogram The two-dimensional representation of a temporal series of *echosounder*-collected data. Time is along the x-axis, and depth along the y-axis. A common way of plotting *echosounder* recordings.

Echosounder An electronic system that includes a computer, transceiver, and *transducer*. The system emits sonar *pings* and records the intensity of the reflected echos at some fixed sampling rate.

Echoview A Windows software application (Echoview Software Pty Ltd, Tasmania, Australia) for hydroacoustic data post-processing.

Entrained air Bubbles of air which have been submerged into the ocean by waves or by the strong *turbulence* commonly found in tidal energy channels.

EV file An *Echoview* file bundling *Sv* data together with associated lines and regions produced by processing.

EVL The *Echoview* line file format.

EVR The *Echoview* region file format.

Inference The procedure of using a *model* to generate output predictions based on a particular input.

Hybrid model A *model* which has been trained on both *downfacing* and *upfacing* data.

Machine learning (ML) The process by which an *algorithm* builds a mathematical model based on *sample* data ("*training data*"), in order to make predictions or decisions without being explicitly programmed to do so. A subset of the field of Artificial Intelligence.

Mobile A mobile *echosounder* is one which is moving (relative to the ocean floor) during its period of operation.

Model A mathematical model of a particular type of data. In our context, the model understands an echogram-like input *sample* of *Sv* data (which is its input) and outputs a probability distribution for where it predicts the *turbulence* (*entrained air*) boundary, *bottom boundary*, and *surface boundary* to be located, and the probability of *passive* periods and *bad data*.

Nearfield The region of space too close to the *echosounder* to collect viable data.

Nearfield distance The maximum distance which is too close to the *echosounder* to be viable for data collection.

Nearfield line A line placed at the *nearfield distance*.

Neural network An artificial neural network contains layers of interconnected neurons with weights between them. The weights are learned through a *machine learning* process. After *training*, the network is a *model* mapping inputs to outputs.

Passive data Data collected while the *echosounder* is silent. Since the sonar pulses are not being generated, only ambient sounds are collected. This package is designed for analysing *active data*, and hence *passive data* is marked for removal.

Ping An *echosounder* sonar pulse event.

Sample (model input) A single echogram-like matrix of *Sv* values.

Sample (ping) A single datapoint recorded at a certain temporal latency in response to a particular *ping*.

Stationary A stationary *echosounder* is at a fixed location (relative to the ocean floor) during its period of operation.

Surface line Separates atmosphere and water at the ocean surface.

Sv The volume backscattering strength.

Test set Data which was used to evaluate the ability of the *model* to generalise to novel, unseen data.

Training The process by which a *model* is iteratively improved.

Training data Data which was used to train the *model(s)*.

Training set A subset (partition) of the *dataset* which was used to train the *model*.

Transducer An underwater electronic device that converts electrical energy to sound pressure energy. The emitted sound pulse is called a "*ping*". The device converts the returning sound pressure energy to electrical energy, which is then recorded.

Turbulence In contrast to laminar flow, fluid motion in turbulent regions are characterized by chaotic fluctuations in flow speed and direction. Air is often entrained into the *water column* in regions of strong turbulence.

Turbulence line A line demarcating the depth of the end-boundary of air entrained into the *water column* by *turbulence* at the sea surface.

Upfacing The orientation of an *echosounder* when it is located at the seabed and records from the *water column* above it.

Validation set Data which was used during the *training* process to evaluate the ability of the *model* to generalise to novel, unseen data.

Water column The body of water between seafloor and ocean surface.

CHAPTER

TWO

CLI REFERENCE

These pages describe the various arguments for the command line interface of the *echofilter* program, which performs the inference process of generating entrained-air, seafloor, and surface lines for an input Echoview EV or CSV file.

Additionally, we provide documentation for the *ev2csv* utility program, which can be used to convert EV files to raw CSV files, the training script *echofilter-train*, and the script *echofilter-generate-shards* which converts raw data to the format to use for the training process.

2.1 echofilter

Remove echosounder noise by identifying the ocean floor and entrained air at the ocean surface.

```
usage: echofilter [-h] [--version] [--list-checkpoints]
                  [--list-colors [{css4,full,xkcd}]] [-c CONFIG_FILE]
                  [--source-dir SOURCE_DIR] [--recursive-dir-search]
                  [--no-recursive-dir-search]
                  [--extension SEARCH_EXTENSION [SEARCH_EXTENSION ...]]
                  [--skip-existing] [--skip-incompatible]
                  [--continue-on-error] [--output-dir OUTPUT_DIR] [--dry-run]
                  [--overwrite-files] [--overwrite-ev-lines] [--force]
                  [--no-ev-import] [--no-turbulence-line] [--no-bottom-line]
                  [--no-surface-line] [--no-nearfield-line]
                  [--suffix-file SUFFIX_FILE] [--suffix-var SUFFIX_VAR]
                  [--color-turbulence COLOR_TURBULENCE]
                  [--color-turbulence-offset COLOR_TURBULENCE_OFFSET]
                  [--color-bottom COLOR_BOTTOM]
                  [--color-bottom-offset COLOR_BOTTOM_OFFSET]
                  [--color-surface COLOR_SURFACE]
                  [--color-surface-offset COLOR_SURFACE_OFFSET]
                  [--color-nearfield COLOR_NEARFIELD]
                  [--thickness-turbulence THICKNESS_TURBULENCE]
                  [--thickness-turbulence-offset THICKNESS_TURBULENCE_OFFSET]
                  [--thickness-bottom THICKNESS_BOTTOM]
                  [--thickness-bottom-offset THICKNESS_BOTTOM_OFFSET]
                  [--thickness-surface THICKNESS_SURFACE]
                  [--thickness-surface-offset THICKNESS_SURFACE_OFFSET]
                  [--thickness-nearfield THICKNESS_NEARFIELD]
                  [--cache-dir CACHE_DIR] [--cache-csv [CSV_DIR]]
                  [--suffix-csv SUFFIX_CSV] [--keep-ext]
```

(continues on next page)

(continued from previous page)

```
                  [--line-status LINE_STATUS] [--offset OFFSET]
                  [--offset-turbulence OFFSET_TURBULENCE]
                  [--offset-bottom OFFSET_BOTTOM]
                  [--offset-surface OFFSET_SURFACE] [--nearfield NEARFIELD]
                  [--cutoff-at-nearfield | --no-cutoff-at-nearfield]
                  [--lines-during-passive {interpolate-time,interpolate-index,predict,
↪redact,undefined}]
                  [--collate-passive-length COLLATE_PASSIVE_LENGTH]
                  [--collate-removed-length COLLATE_REMOVED_LENGTH]
                  [--minimum-passive-length MINIMUM_PASSIVE_LENGTH]
                  [--minimum-removed-length MINIMUM_REMOVED_LENGTH]
                  [--minimum-patch-area MINIMUM_PATCH_AREA]
                  [--patch-mode PATCH_MODE] [--variable-name VARIABLE_NAME]
                  [--keep-exclusions]
                  [--row-len-selector {init,min,max,median,mode}]
                  [--facing {downward,upward,auto}]
                  [--training-standardization]
                  [--prenorm-nan-value PRENORM_NAN_VALUE]
                  [--postnorm-nan-value POSTNORM_NAN_VALUE]
                  [--crop-min-depth CROP_MIN_DEPTH]
                  [--crop-max-depth CROP_MAX_DEPTH]
                  [--autocrop-threshold AUTOCROP_THRESHOLD]
                  [--image-height IMAGE_HEIGHT] [--checkpoint CHECKPOINT]
                  [--unconditioned]
                  [--logit-smoothing-sigma SIGMA [SIGMA ...]]
                  [--device DEVICE]
                  [--hide-echoview | --show-echoview | --always-hide-echoview]
                  [--minimize-echoview] [--verbose] [--quiet]
                 FILE_OR_DIRECTORY [FILE_OR_DIRECTORY ...]
```

2.1.1 Actions

These arguments specify special actions to perform. The main action of this program is supressed if any of these are given.

--version, -V Show program's version number and exit.

--list-checkpoints Show the available model checkpoints and exit.

--list-colors, --list-colours Possible choices: css4, full, xkcd

Show the available line color names and exit. The available color palette can be viewed at https://matplotlib.org/gallery/color/named_colors.html. The XKCD color palette is also available, but is not shown in the output by default due to its size. To show the just main palette, run as `--list-colors` without argument, or `--list-colors css4`. To show the full palette, run as `--list-colors full`.

2.1.2 Configuration

-c, --config Path to a configuration file. The settings in the configuration file will override the default values described in the rest of the help documentation, but will themselves be overridden by any arguments provided at the command prompt. Config file syntax allows: key=value, flag=true, stuff=[a,b,c] (for details, see syntax at https://goo.gl/R74nmi).

2.1.3 Positional arguments

FILE_OR_DIRECTORY File(s)/directory(ies) to process. Inputs can be absolute paths or relative paths to either files or directories. Paths can be given relative to the current directory, or optionally be relative to the SOURCE_DIR argument specified with `--source-dir`. For each directory given, the directory will be searched recursively for files bearing an extension specified by SEARCH_EXTENSION (see the `--extension` argument for details). Multiple files and directories can be specified, separated by spaces. This is a required argument. At least one input file or directory must be given, unless one of the arguments listed above under "Actions" is given. In order to process the directory given by SOURCE_DIR, specify "." for this argument, such as:

```
echofilter . --source-dir SOURCE_DIR
```

2.1.4 Input file arguments

Optional parameters specifying which files will processed.

--source-dir, -d Path to source directory which contains the files and folders specified by the paths argument. Default: `"."` (the current directory).

--recursive-dir-search, -r For any directories provided in the FILE_OR_DIRECTORY input, all subdirectories will also be recursively walked through to find files to process. This is the default behaviour.

--no-recursive-dir-search, -R For any directories provided in the FILE_OR_DIRECTORY input, only files within the specified directory will be included in the files to process. Subfolders within the directory will not be included.

--extension, -x File extension(s) to process. This argument is used when the FILE_OR_DIRECTORY is a directory; files within the directory (and all its recursive subdirectories) are filtered against this list of extensions to identify which files to process. Default: `['csv']`. (Note that the default SEARCH_EXTENSION value is OS-specific.)

--skip-existing, --skip, -s Skip processing files for which all outputs already exist

--skip-incompatible Skip over incompatible input CSV files, without raising an error. Default behaviour is to stop if an input CSV file can not be processed. This argument is useful if you are processing a directory which contains a mixture of CSV files - some are Sv data exported from EV files and others are not.

--continue-on-error Continue running on remaining files if one file hits an error.

2.1.5 Destination file arguments

Optional parameters specifying where output files will be located.

--output-dir, -o Path to output directory. If empty (default), each output is placed in the same directory as its input file. If OUTPUT_DIR is specified, the full output path for each file contains the subtree of the input file relative to the base directory given by SOURCE_DIR.

--dry-run, -n Perform a trial run, with no changes made. Text printed to the command prompt indicates which files would be processed, but work is only simulated and not performed.

--overwrite-files Overwrite existing files without warning. Default behaviour is to stop processing if an output file already exists.

--overwrite-ev-lines Overwrite existing lines within the Echoview file without warning. Default behaviour is to append the current datetime to the name of the line in the event of a collision.

--force, -f Short-hand equivalent to supplying both `--overwrite-files` and `--overwrite-ev-lines`.

--no-ev-import Do not import lines and regions back into any EV file inputs. Default behaviour is to import lines and regions and then save the file, overwriting the original EV file.

--no-turbulence-line Do not output an evl file for the turbulence line, and do not import a turbulence line into the EV file.

--no-bottom-line Do not output an evl file for the bottom line, and do not import a bottom line into the EV file.

--no-surface-line Do not output an evl file for the surface line, and do not import a surface line into the EV file.

--no-nearfield-line Do not add a nearfield line to the EV file.

--suffix-file, --suffix Suffix to append to output artifacts evl and evr files, between the name of the file and the extension. If SUFFIX_FILE begins with an alphanumeric character, "-" is prepended to it to act as a delimiter. The default behavior is to not append a suffix.

--suffix-var Suffix to append to line and region names when imported back into EV file. If SUFFIX_VAR begins with an alphanumeric character, "-" is prepended to it to act as a delimiter. The default behaviour is to match SUFFIX_FILE if it is set, and use `"_echofilter"` otherwise.

--color-turbulence Color to use for the turbulence line when it is imported into Echoview. This can either be the name of a supported color (see `--list-colors` for options), or a a hexadecimal string, or a string representation of an RGB color to supply directly to Echoview (such as `"(0,255,0)"`). Default: `"orangered"`.

--color-turbulence-offset Color to use for the offset turbulence line when it is imported into Echoview. If unset, this will be the same as COLOR_TURBULENCE.

--color-bottom Color to use for the bottom line when it is imported into Echoview. This can either be the name of a supported color (see `--list-colors` for options), or a a hexadecimal string, or a string representation of

an RGB color to supply directly to Echoview (such as `"(0,255,0)"`). Default: `"orangered"`.

--color-bottom-offset Color to use for the offset bottom line when it is imported into Echoview. If unset, this will be the same as COLOR_BOTTOM.

--color-surface Color to use for the surface line when it is imported into Echoview. This can either be the name of a supported color (see `--list-colors` for options), or a a hexadecimal string, or a string representation of an RGB color to supply directly to Echoview (such as `"(0,255,0)"`). Default: `"green"`.

--color-surface-offset Color to use for the offset surface line when it is imported into Echoview. If unset, this will be the same as COLOR_SURFACE.

--color-nearfield Color to use for the nearfield line when it is created in Echoview. This can either be the name of a supported color (see `--list-colors` for options), or a a hexadecimal string, or a string representation of an RGB color to supply directly to Echoview (such as `"(0,255,0)"`). Default: `"mediumseagreen"`.

--thickness-turbulence Thicknesses with which the turbulence line will be displayed in Echoview. Default: `2`.

--thickness-turbulence-offset Thicknesses with which the offset turbulence line will be displayed in Echoview. If unset, this will be the same as THICKNESS_TURBULENCE.

--thickness-bottom Thicknesses with which the bottom line will be displayed in Echoview. Default: `2`.

--thickness-bottom-offset Thicknesses with which the offset bottom line will be displayed in Echoview. If unset, this will be the same as THICKNESS_BOTTOM.

--thickness-surface Thicknesses with which the surface line will be displayed in Echoview. Default: `1`.

--thickness-surface-offset Thicknesses with which the offset surface line will be displayed in Echoview. If unset, this will be the same as THICKNESS_SURFACE.

--thickness-nearfield Thicknesses with which the nearfield line will be displayed in Echoview. Default: `1`.

--cache-dir Path to checkpoint cache directory. Default: `"/home/docs/.cache/echofilter"`.

--cache-csv Path to directory where CSV files generated from EV inputs should be cached. If this argument is supplied with an empty string, exported CSV files will be saved in the same directory as each input EV file. The default behaviour is discard any CSV files generated by this program once it has finished running.

--suffix-csv Suffix to append to the file names of cached CSV files which are exported from EV files. The suffix is inserted between the input file name and the new file extension, ".csv". If SUFFIX_CSV begins with an alphanumeric character, a delimiter is prepended. The delimiter is "-", or "." if `--keep-ext` is given. The default behavior is to not append a suffix.

--keep-ext If provided, the output file names (evl, evr, csv) maintain the input file extension before their suffix (including a new file extension). Default behaviour is to strip the input file name extension before constructing the output paths.

2.1.6 Output configuration arguments

Optional parameters specifying the properties of the output.

--line-status Status value for all the lines which are generated. Options are:

0: none, 1: unverified, 2: bad, 3: good

Default: `3`.

--offset Offset for turbulence, bottom, and surface lines, in metres. This will shift turbulence and surface lines downwards and the bottom line upwards by the same distance of OFFSET. Default: `1.0`.

--offset-turbulence Offset for the turbulence line, in metres. This shifts the turbulence line downards by some distance OFFSET_TURBULENCE. If this is set, it overwrites the value provided by `--offset`.

--offset-bottom Offset for the bottom line, in metres. This shifts the bottom line upwards by some distance OFFSET_BOTTOM. If this is set, it overwrites the value provided by `--offset`.

--offset-surface Offset for the surface line, in metres. This shifts the surface line downards by some distance OFFSET_SURFACE. If this is set, it overwrites the value provided by `--offset`.

--nearfield Nearfield distance, in metres. Default: `1.7`. If the echogram is downward facing, the nearfield cutoff will be NEARFIELD meters below the shallowest depth recorded in the input data. If the echogram is upward facing, the nearfield cutoff will be NEARFIELD meters above the deepest depth recorded in the input data. When processing an EV file, by default a nearfield line will be added at the nearfield cutoff depth. To prevent this behaviour, use the `--no-nearfield-line` argument.

--cutoff-at-nearfield Enable cut-off at the nearfield distance for both the turbulence line (on downfacing data) as well as the bottom line (on upfacing data). Default behavior is to only clip the bottom line.

--no-cutoff-at-nearfield Disable cut-off at the nearfield distance for both the turbulence line (on downfacing data) and the bottom line (on upfacing data). Default behavior is to clip the bottom line but not the turbulence line.

--lines-during-passive Possible choices: interpolate-time, interpolate-index, predict, redact, undefined

Method used to handle line depths during collection periods determined to be passive recording instead of active recording. Options are:

interpolate-time: depths are linearly interpolated from active recording periods, using the time at which recordings where made.

interpolate-index: depths are linearly interpolated from active recording periods, using the index of the recording.

predict: the model's prediction for the lines during passive data collection will be kept; the nature of the prediction depends on how the model was trained.

redact: no depths are provided during periods determined to be passive data collection.

undefined: depths are replaced with the placeholder value used by Echoview to denote undefined values, which is -10000.99.

Default: `"interpolate-time"`.

--collate-passive-length Maximum interval, in ping indices, between detected passive regions which will removed to merge consecutive passive regions together into a single, collated, region. Default: `10`.

--collate-removed-length Maximum interval, in ping indices, between detected blocks (vertical rectangles) marked for removal which will also be removed to merge consecutive removed blocks together into a single, collated, region. Default: `10`.

--minimum-passive-length Minimum length, in ping indices, which a detected passive region must have to be included in the output. Set to -1 to omit all detected passive regions from the output. Default: `10`.

--minimum-removed-length Minimum length, in ping indices, which a detected removal block (vertical rectangle) must have to be included in the output. Set to -1 to omit all detected removal blocks from the output (default). When enabling this feature, the recommended minimum length is 10.

--minimum-patch-area Minimum area, in pixels, which a detected removal patch (contour/polygon) region must have to be included in the output. Set to -1 to omit all detected patches from the output (default). When enabling this feature, the recommended minimum area is 25.

--patch-mode Type of mask patches to use. Must be supported by the model checkpoint used. Should be one of:

merged: Target patches for training were determined after merging as much as possible into the turbulence and bottom lines.

original: Target patches for training were determined using original lines, before expanding the turbulence and bottom lines.

ntob: Target patches for training were determined using the original bottom line and the merged turbulence line.

Default: "merged" is used if downfacing; "ntob" if upfacing.

2.1.7 Input processing arguments

Optional parameters specifying how data will be loaded from the input files and transformed before it given to the model.

--variable-name, --vn Name of the Echoview acoustic variable to load from EV files. Default: `"Fileset1: Sv pings T1"`.

--keep-exclusions, --keep-thresholds Export CSV with all thresholds, exclusion regions, and bad data exclusions set as per the EV file. Default behavior is to ignore these settings and export the underlying raw data.

--row-len-selector Possible choices: init, min, max, median, mode

How to handle inputs with differing number of depth samples across time. This method is used to select the "master" number of depth samples and minimum and maximum depth. The Sv values for all timepoints are interpolated onto this range of depths in order to create an input which is sampled in a rectangular manner. Default: `"mode"`, the modal number of depths is used, and the modal depth range is select amongst time samples which bear this number of depths.

--facing Possible choices: downward, upward, auto

Orientation of echosounder. If this is "auto" (default), the orientation is automatically determined from the ordering of the depths field in the input (increasing depth values = "downward"; diminishing depths = "upward").

--training-standardization If this is given, Sv intensities are scaled using the values used when the model was trained before being given to the model for inference. The default behaviour is to derive the standardization values from the Sv statistics of the input instead.

--prenorm-nan-value If set, NaN values in the imported CSV data will be replaced with this Sv intensity value.

--postnorm-nan-value If set, NaN values in the imported CSV data will be replaced with this Sv intensity value after the input distribution has been standardized to have zero mean and unit variance.

--crop-min-depth Shallowest depth, in metres, to analyse. Data will be truncated at this depth, with shallower data removed before the Sv input is shown to the model. Default behaviour is not to truncate.

--crop-max-depth Deepest depth, in metres, to analyse. Data will be truncated at this depth, with deeper data removed before the Sv input is shown to the model. Default behaviour is not to truncate.

--autocrop-threshold, --autozoom-threshold The inference routine will re-run the model with a zoomed in version of the data, if the fraction of the depth which it deems irrelevant exceeds the AUTO_CROP_THRESHOLD. The extent of the depth which is deemed relevant is from the shallowest point on the surface line to the deepest point on the bottom line. The data will only be zoomed in and re-analysed at most once. To always run the model through once (never auto zoomed), set to 1. To always run the model through exactly twice (always one round of auto-zoom), set to 0. Default: `0.35`.

--image-height, --height Height to which the Sv image will be rescaled, in pixels, before being given to the model. The default behaviour is to use the same height as was used when the model was trained.

2.1.8 Model arguments

Optional parameters specifying which model checkpoint will be used and how it is run.

--checkpoint Name of checkpoint to load, or path to a checkpoint file. Default: `"conditional_mobile-stationary2_effunet6x2-1_lc32_v2.2"`.

--unconditioned, --force-unconditioned If this flag is present and a conditional model is loaded, it will be run for its unconditioned output. This means the model is output is not conditioned on the orientation of the echosounder. By default, conditional models are used for their conditional output.

--logit-smoothing-sigma Standard deviation of Gaussian smoothing kernel applied to the logits provided as the model's output. The smoothing regularises the output to make it smoother. Multiple values can be given to use different kernel sizes for each dimension, in which case the first value is for the timestamp dimension and the second value is for the depth dimension. If a single value is given, the kernel is symmetric. Values are relative to the pixel space returned by the UNet model. Disabled by default.

--device Device to use for running the model for inference. Default: use first GPU if available, otherwise use the CPU. Note: echofilter.exe is complied without GPU support and can only run on the CPU. To use the GPU you must use the source version.

2.1.9 Echoview window management

Optional parameters specifying how to interact with any Echoview windows which are used during this process.

--hide-echoview Hide any Echoview window spawned by this program. If it must use an Echoview instance which was already running, that window is not hidden. This is the default behaviour.

--show-echoview Don't hide an Echoview window created to run this code. (Disables the default behaviour which is equivalent to `--hide-echoview`.)

--always-hide-echoview, --always-hide Hide the Echoview window while this code runs, even if this process is utilising an Echoview window which was already open.

--minimize-echoview Minimize any Echoview window used to runs this code while it runs. The window will be restored once the program is finished. If this argument is supplied, `--show-echoview` is implied unless `--hide-echoview` is also given.

2.1.10 Verbosity arguments

Optional parameters controlling how verbose the program should be while it is running.

--verbose, -v Increase the level of verbosity of the program. This can be specified multiple times, each will increase the amount of detail printed to the terminal. The default verbosity level is 2.

--quiet, -q Decrease the level of verbosity of the program. This can be specified multiple times, each will reduce the amount of detail printed to the terminal.

2.2 ev2csv

Echoview to raw CSV exporter

```
usage: ev2csv [-h] [--version] [--source-dir SOURCE_DIR]
              [--recursive-dir-search] [--no-recursive-dir-search]
              [--skip-existing] [--keep-exclusions] [--output-dir OUTPUT_DIR]
              [--dry-run] [--force] [--keep-ext] [--output-suffix SUFFIX]
              [--variable-name VARIABLE_NAME]
              [--hide-echoview | --show-echoview | --always-hide-echoview]
              [--minimize-echoview] [--verbose] [--quiet]
              FILE_OR_DIRECTORY [FILE_OR_DIRECTORY ...]
```

2.2.1 Actions

These arguments specify special actions to perform. The main action of this program is supressed if any of these are given.

--version, -V Show program's version number and exit.

2.2.2 Positional arguments

FILE_OR_DIRECTORY File(s)/directory(ies) to process. Inputs can be absolute paths or relative paths to either files or directories. Paths can be given relative to the current directory, or optionally be relative to the SOURCE_DIR argument specified with `--source-dir`. For each directory given, the directory will be searched recursively for files bearing an extension specified by SEARCH_EXTENSION (see the `--extension` argument for details). Multiple files and directories can be specified, separated by spaces. This is a required argument. At least one input file or directory must be given. In order to process the directory given by SOURCE_DIR, specify "." for this argument, such as:

```
ev2csv . --source-dir SOURCE_DIR
```

2.2.3 Input file arguments

Optional parameters specifying which files will processed.

--source-dir, -d Path to source directory which contains the files and folders specified by the paths argument. Default: `"."` (the current directory).

--recursive-dir-search For any directories provided in the FILE_OR_DIRECTORY input, all subdirectories will also be recursively walked through to find files to process. This is the default behaviour.

--no-recursive-dir-search For any directories provided in the FILE_OR_DIRECTORY input, only files within the specified directory will be included in the files to process. Subfolders within the directory will not be included.

--skip-existing, --skip Skip processing files for which all outputs already exist

2.2.4 Processing arguments

Optional parameters specifying how to process files.

--keep-exclusions, --keep-thresholds Export CSV with all thresholds, exclusion regions, and bad data exclusions set as per the EV file. Default behavior is to ignore these settings and export the underlying raw data.

2.2.5 Destination file arguments

Optional parameters specifying where output files will be located.

--output-dir, -o Path to output directory. If empty (default), each output is placed in the same directory as its input file. If OUTPUT_DIR is specified, the full output path for each file all contains the subtree of the input file relative to the base directory given by SOURCE_DIR.

--dry-run, -n Perform a trial run, with no changes made. Text printed to the command prompt indicates which files would be processed, but work is only simulated and not performed.

--force, -f Overwrite existing files without warning. Default behaviour is to stop processing if an output file already exists.

--keep-ext If provided, the output file names (evl, evr, csv) maintain the input file extension before their suffix (including a new file extension). Default behaviour is to strip the input file name extension before constructing the output paths.

--output-suffix, --suffix Output filename suffix. Default is `"_Sv_raw.csv"`, or `".Sv_raw.csv"` if the `--keep_ext` argument is supplied. if `--keep-exclusions` is given, the `"_raw"` component is dropped.

2.2.6 Input processing arguments

Optional parameters specifying how data will be loaded from the input files and transformed before it given to the model.

--variable-name, --vn Name of the Echoview acoustic variable to load from EV files. Default: `"Fileset1: Sv pings T1"`.

2.2.7 Echoview window management

Optional parameters specifying how to interact with any Echoview windows which are used during this process.

--hide-echoview Hide any Echoview window spawned by this program. If it must use an Echoview instance which was already running, that window is not hidden. This is the default behaviour.

--show-echoview Don't hide an Echoview window created to run this code. (Disables the default behaviour which is equivalent to `--hide-echoview`.)

--always-hide-echoview, --always-hide Hide the Echoview window while this code runs, even if this process is utilising an Echoview window which was already open.

--minimize-echoview Minimize any Echoview window used to runs this code while it runs. The window will be restored once the program is finished. If this argument is supplied, `--show-echoview` is implied unless `--hide-echoview` is also given.

2.2.8 Verbosity arguments

Optional parameters controlling how verbose the program should be while it is running.

--verbose, -v Increase the level of verbosity of the program. This can be specified multiple times, each will increase the amount of detail printed to the terminal. The default verbosity level is 1.

--quiet, -q Decrease the level of verbosity of the program. This can be specified multiple times, each will reduce the amount of detail printed to the terminal.

2.3 echofilter-train

Echofilter model training

```
usage: echofilter-train [-h] [--version] [--data-dir DIR]
                        [--dataset DATASET_NAME]
                        [--train-partition TRAIN_PARTITION]
                        [--val-partition VAL_PARTITION]
                        [--shape SAMPLE_SHAPE SAMPLE_SHAPE]
                        [--crop-depth CROP_DEPTH] [--resume PATH]
                        [--cold-restart] [--warm-restart] [--log LOG_NAME]
                        [--log-append LOG_NAME_APPEND] [--conditional]
```

(continues on next page)

(continued from previous page)

```
                        [--nblock N_BLOCK] [--latent-channels LATENT_CHANNELS]
                        [--expansion-factor EXPANSION_FACTOR]
                        [--expand-only-on-down]
                        [--blocks-per-downsample BLOCKS_PER_DOWNSAMPLE [BLOCKS_PER_
↪DOWNSAMPLE ...]]
                        [--blocks-before-first-downsample BLOCKS_BEFORE_FIRST_DOWNSAMPLE␣
↪[BLOCKS_BEFORE_FIRST_DOWNSAMPLE ...]]
                        [--only-skip-connection-on-downsample]
                        [--deepest-inner DEEPEST_INNER]
                        [--intrablock-expansion INTRABLOCK_EXPANSION]
                        [--se-reduction SE_REDUCTION]
                        [--downsampling-modes DOWNSAMPLING_MODES [DOWNSAMPLING_MODES ...
↪]]
                        [--upsampling-modes UPSAMPLING_MODES [UPSAMPLING_MODES ...]]
                        [--fused-conv] [--no-residual] [--actfn ACTFN]
                        [--kernel KERNEL_SIZE] [--device DEVICE] [--multigpu]
                        [--no-amp] [--amp-opt AMP_OPT] [-j N] [-p PRINT_FREQ]
                        [-b BATCH_SIZE] [--no-stratify] [--epochs N_EPOCH]
                        [--seed SEED] [--optim OPTIMIZER]
                        [--schedule SCHEDULE] [--lr LR] [--momentum MOMENTUM]
                        [--base-momentum BASE_MOMENTUM] [--wd WEIGHT_DECAY]
                        [--warmup-pct WARMUP_PCT]
                        [--warmdown-pct WARMDOWN_PCT]
                        [--anneal-strategy ANNEAL_STRATEGY]
                        [--overall-loss-weight OVERALL_LOSS_WEIGHT]
```

2.3.1 Actions

These arguments specify special actions to perform. The main action of this program is supressed if any of these are given.

--version, -V Show program's version number and exit.

2.3.2 Data parameters

--data-dir path to root data directory

--dataset which dataset to use

--train-partition which partition to train on (default depends on dataset)

--val-partition which partition to validate on (default depends on dataset)

--shape input shape [W, H] (default: `(128, 512)`)

--crop-depth depth, in metres, at which data should be truncated (default: `None`)

--resume path to latest checkpoint (default: ````)

--cold-restart when resuming from a checkpoint, use this only for initial weights

--warm-restart when resuming from a checkpoint, use the existing weights and optimizer state but start a new LR schedule

--log output directory name (default: DATE_TIME)

--log-append string to append to output directory name (default: HOSTNAME)

2.3.3 Model parameters

--conditional train a model conditioned on the direction the sounder is facing (in addition to an unconditional model)

--nblock, --num-blocks number of blocks down and up in the UNet (default: `6`)

--latent-channels number of initial/final latent channels to use in the model (default: `32`)

--expansion-factor expansion for number of channels as model becomes deeper (default: `1.0`, constant number of channels)

--expand-only-on-down only expand channels on dowsampling blocks

--blocks-per-downsample for each dim (time, depth), number of blocks between downsample steps (default: `(2, 1)`)

--blocks-before-first-downsample for each dim (time, depth), number of blocks before first downsample step (default: `(2, 1)`)

--only-skip-connection-on-downsample only include skip connections when downsampling

--deepest-inner layer to include at the deepest point of the UNet (default: "horizontal_block"). Set to "identity" to disable.

--intrablock-expansion expansion within inverse residual blocks (default: `6.0`)

--se-reduction, --se reduction within squeeze-and-excite blocks (default: `4.0`)

--downsampling-modes for each downsampling step, the method to use (default: `"max"`)

--upsampling-modes for each upsampling step, the method to use (default: `"bilinear"`)

--fused-conv use fused instead of depthwise separable convolutions

--no-residual don't use residual blocks

--actfn activation function to use

--kernel convolution kernel size (default: `5`)

2.3.4 Training parameters

--device device to use (default: `"cuda"`, using first gpu)

--multigpu train on multiple GPUs

--no-amp use fp32 instead of mixed precision (default: use mixed precision on gpu)

--amp-opt optimizer level for apex automatic mixed precision (default: `"O1"`)

-j, --workers number of data loading workers (default: `8`)

-p, --print-freq print frequency (default: `50`)

-b, --batch-size mini-batch size (default: `16`)

--no-stratify disable stratified sampling; use fully random sampling instead

--epochs number of total epochs to run (default: `20`)

--seed seed for initializing training.

2.3.5 Optimizer parameters

--optim, --optimiser, --optimizer optimizer name (default: `"rangerva"`)

--schedule LR schedule (default: `"constant"`)

--lr, --learning-rate initial learning rate (default: `0.1`)

--momentum momentum (default: `0.9`)

--base-momentum base momentum; only used for OneCycle schedule (default: same as momentum)

--wd, --weight-decay weight decay (default: `1e-05`)

--warmup-pct fraction of training to spend warming up LR; only used for OneCycle MesaOneCycle schedules (default: `0.2`)

--warmdown-pct fraction of training before warming down LR; only used for MesaOneCycle schedule (default: `0.7`)

--anneal-strategy annealing strategy; only used for OneCycle schedule (default: `"cos"`)

--overall-loss-weight weighting for overall loss term (default: `0.0`)

2.4 echofilter-generate-shards

Generate dataset shards

```
usage: echofilter-generate-shards [-h] [--version] [--root ROOT_DATA_DIR]
                                  [--partitioning-version PARTITIONING_VERSION]
                                  [--max-depth MAX_DEPTH]
                                  [--shard-len SHARD_LEN] [--ncores NCORES]
                                  [--verbose]
                                  partition dataset
```

2.4.1 Positional Arguments

partition partition to shard

dataset dataset to shard

2.4.2 Named Arguments

--version, -V show program's version number and exit

--root root data directory

Default: "/data/dsforce/surveyExports"

--partitioning-version partitioning version

Default: "firstpass"

--max-depth maximum depth to include in sharded data

--shard-len	number of samples in each shard Default: 128
--ncores	number of cores to use (default: all). Set to 1 to disable multiprocessing.
--verbose, -v	increase verbosity Default: 0

CHAPTER

THREE

CHANGELOG

All notable changes to echofilter will be documented here.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

Categories for changes are: Added, Changed, Deprecated, Removed, Fixed, Security.

3.1 Version 1.1.1

Release date: 2022-11-16. Full commit changelog.

3.1.1 Fixed

Inference

- EVL final value pad was for a timestamp in between the preceding two, not extending forward in time by half a timepoint. (#300)

Metadata

- Declare `python_requires<3.11` requirement. (#302)
- Declare `torch<1.12.0` requirement. (#302)

3.2 Version 1.1.0

Release date: 2022-11-12. Full commit changelog.

3.2.1 Changed

Inference

- Disable logit smoothing by default. The previous behaviour can be restored by setting `--logit-smoothing-sigma=1` at the CLI. (#293)

3.2.2 Fixed

Inference

- Fix bug where joined segments of data would have their first ping dropped. (#272)

Training

- Make the number of channels in the first block respect the `initial_channels` argument. (#271)

Miscellaneous

- Fix unseen internal bugs, including in `generate_shards`. (#283)

3.2.3 Added

Inference

- Add support for using a config file to provide arguments to the CLI. (#294)
- Add `--continue-on-error` argument to inference routine, which will capture an error when processing an individual file and continue running the rest. (#245)
- Break up large files into more manageable chunks of at most 1280 pings, to reduce out-of-memory errors. (#245)
- Reduce GPU memory consumption during inference by moving outputs to CPU memory sooner. (#245)
- Fill in missing values in the input file through 2d linear interpolation. (#246)
- Pad Sv data in timestamp dimension during inference to ensure the data is fully within the network's effective receptive field. (#277)
- Add `--prenorm-nan-value` and `--postnorm-nan-value` options to control what value NaN values in the input are mapped to. (#274)
- Add support for providing a single path as a string to the run_inference API. (Note that the CLI already supported this and so is unchanged). (#288)
- Add more verbosity messages. (#276, #278, #292)

ev2csv

- Add `--keep-thresholds` option which allow for exporting Sv data with thresholds and exclusions enabled (set as they currently are in the EV file). The default behaviour is still to export raw Sv data (disabling all thresholds). The default file name for the CSV file depends on whether the export is of raw or thresholded data. (#275)
- Add `--keep-ext` argument to ev2csv, which allows the existing extension on the input path to be kept preceding the new file extension. (#242)

Tests

- Add tests which check that inference commands run, whether checking their outputs. (#289)

Internal

- Add EVR reader `echofilter.raw.loader.evr_reader`. (#280)

3.3 Version 1.0.3

Release date: 2022-11-15. Full commit changelog.

This minor patch fix addresses package metadata.

3.3.1 Fixed

Metadata

- Declare `python_requires>=3.6,<3.11` requirement. (#264, #302)
- Declare `torch<1.12.0` requirement. (#302)

3.4 Version 1.0.2

Release date: 2022-11-06. Full commit changelog.

This minor patch fix addresses github dependencies so the package can be pushed to PyPI.

3.4.1 Changed

Requirements

- Change `torch_lr_finder` train requirement from a specific github commit ref to >=0.2.0. (#260)
- Remove ranger from train requirements. (#261)

Training

- Default optimizer changed from `"rangerva"` to `"adam"`. If you have manually installed ranger you can still use the `"rangerva"` optimizer if you specify it. (#261)

3.5 Version 1.0.1

Release date: 2022-11-06. Full commit changelog.

This patch fix addresses requirement inconsistencies and documentation building. This release is provided under the AGPLv3 license.

3.5.1 Changed

Requirements

- Add a vendorized copy of functions from torchutils and remove it from the requirements. (#249)

3.5.2 Fixed

Release

- Added checkpoints.yaml file to package_data. (#255)
- Added appdirs package, required for caching model checkpoints. (#240)
- Support for pytorch>=1.11 by dropping import of `torch._six.container_abcs`. (#250)

3.6 Version 1.0.0

Release date: 2020-10-18. Full commit changelog.

This is the first major release of echofilter.

3.6.1 Added

Inference

- Add support for loading checkpoints shipped as part of the package. (#228)
- More detailed error messages when unable to download or load a model i.e. due to a problem with the Internet connection, a 404 error, or because the hard disk is out of space. (#228)

Documentation

- Add Usage Guide source and sphinx documentation PDF generation routines (#232, #233, #234, #235)

3.7 Version 1.0.0rc3

Release date: 2020-09-23. Full commit changelog.

This is the third release candidate for the forthcoming v1.0.0 major release.

3.7.1 Fixed

Inference

- Include extension in temporary EVL file, fixing issue importing it into Echoview. (#224)

3.8 Version 1.0.0rc2

Release date: 2020-09-23. Full commit changelog.

This is the second release candidate for the forthcoming v1.0.0 major release.

3.8.1 Fixed

Inference

- Fix reference to `echofilter.raw.loader.evl_loader` when loading EVL files into Echoview. (#222)

3.9 Version 1.0.0rc1

Release date: 2020-09-23. Full commit changelog.

This is a release candidate for the forthcoming v1.0.0 major release.

3.9.1 Changed

Inference

- Import lines into Echoview twice, once with and once without offset. (#218)
- EVL outputs now indicate raw depths, before any offset or clipping is applied. (#218)
- Change default `--lines-during-passive` value from `"predict"` to `"interpolate-time"`. (#216)
- Disable all bad data region outputs by default. (#217)
- Change default nearfield cut-off behaviour to only clip the bottom line (upfacing data) and not the turbulence line (downfacing data). (#219)

Training

- Reduce minimum distance by which surface line must be above turbulence line from 0.25m to 0m. (#212)
- Reduce minimum distance by which bottom line must be above surface line from 0.5m to 0.02m. (#212)

3.9.2 Fixed

Inference

- Change nearfield line for downfacing recordings to be nearfield distance below the shallowest recording depth, not at a depth equal to the nearfield distance. (#214)

3.9.3 Added

Inference

- Add new checkpoints: v2.0, v2.1 for stationary model; v2.0, v2.1, v2.2 for conditional hybrid model. (#213)
- Add notes to lines imported into Echoview. (#215)
- Add arguments controlling color and thickness of offset lines (`--color-surface-offset`, etc). (#218)
- Add argument `--cutoff-at-nearfield` which re-enables clipping of the turbulence line at nearfield depth with downfacing data. (#219)

3.10 Version 1.0.0b4

Release date: 2020-07-05. Full commit changelog.

This is a beta pre-release of v1.0.0.

3.10.1 Changed

Inference

- Arguments relating to top are renamed to turbulence, and "top" outputs are renamed "turbulence". (#190)
- Change default checkpoint from `conditional_mobile-stationary2_effunet6x2-1_lc32_v1.0` to `conditional_mobile-stationary2_effunet6x2-1_lc32_v2.0`. (#208)
- Status value in EVL outputs extends to final sample (as per specification, not observed EVL files). (#201)
- Rename `--nearfield-cutoff` argument to `--nearfield`, add `--no-cutoff-at-nearfield` argument to control whether the turbulence/bottom line can extend closer to the echosounder that the nearfield line. (#203)
- Improved UI help and verbosity messages. (#187, #188, #203, #204, #207)

Training

- Use 0m as target for surface line for downfacing, not the top of the echogram. (#191)
- Don't include periods where the surface line is below the bottom line in the training loss. (#191)
- Bottom line target during nearfield is now the bottom of the echogram, not 0.5m above the bottom. (#191)
- Normalise training samples separately, based on their own Sv intensity distribution after augmentation. (#192)
- Record echofilter version number in checkpoint file. (#193)
- Change "optimal" depth zoom augmentation, used for validation, to cover a slightly wider depth range past the deepest bottom and shallowest surface line. (#194)
- Don't record fraction of image which is active during training. (#206)

Miscellaneous

- Rename top->turbulence, bot->bottom surf->surface, throughout all code. (#190)
- Convert undefined value -10000.99 to NaN when loading lines from EVL files. (#191)
- Include surface line in transect plots. (#191)
- Move argparser and colour styling into ui subpackage. (#198)
- Move inference command line interface to its own module to increase responsiveness for non-processing actions (`--help`, `--version`, `--list-checkpoints`, `--list-colors`). (#199)

3.10.2 Fixed

Inference

- Fix depth extent of region boxes. (#186)
- EVL and EVR outputs extend half a timestamp interval so it is clear what is inside their extent. (#200)

Training

- Labels for passive collection times in Minas Passage and Grand Passage datasets are manually set for samples where automatic labeling failed. (#191)
- Interpolate surface depths during passive periods. (#191)
- Smooth out anomalies in the surface line, and exclude the smoothed version from the training loss. (#191)
- Use a looser nearfield removal process when removing the nearfield zone from the bottom line targets, so nearfield is removed from all samples where it needs to be. (#191)
- When reshaping samples, don't use higher order interpolation than first for the bottom line with upfacing data, as the boundaries are rectangular (#191)
- The precision criterion's measurement value when there are no predicted positives equals 1 and if there are no true positives and 0 otherwise (previously 0.5 regardless of target). (#195)

3.10.3 Added

Inference

- Add nearfield line to EV file when importing lines, and add `--no-nearfield-line` argument to disable this. (#203)
- Add arguments to control display of nearfield line, `--color-nearfield` and `--thickness-nearfield`. (#203)
- Add `-r` and `-R` short-hand arguments for recursive and non-recursive directory search. (#189)
- Add `-s` short-hand argument for `--skip` (#189)
- Add two new model checkpoints to list of available checkpoints, `conditional_mobile-stationary2_effunet6x2-1_lc32_v1` and `conditional_mobile-stationary2_effunet6x2-1_lc32_v2.0`. (#208)
- Use YAML file to define list of available checkpoints. (#208, #209)
- Default checkpoint is shown with an asterisk in checkpoint list. (#202)

Training

- Add cold/warm restart option, for training a model with initial weights from the output of a previously trained model. (#196)
- Add option to manually specify training and validation partitions. (#205)

3.11 Version 1.0.0b3

Release date: 2020-06-25. Full commit changelog.

This is a beta pre-release of v1.0.0.

3.11.1 Changed

Inference

- Rename `--crop-depth-min` argument to `--crop-min-depth`, and `--crop-depth-max` argument to `--crop-max-depth`. (#174)
- Rename `--force_unconditioned` argument to `--force-unconditioned`. (#166)
- Default offset of surface line is now 1m. (#168)
- Change default `--checkpoint` so it is always the same (the conditional model), independent of the `--facing` argument. (#177)
- Change default `--lines-during-passive` from `"redact"` to `"predict"`. (#176)
- Change `--sufix-csv` behaviour so it should no longer include `".csv"` extension, matching how `--suffix-file` is handled. (#171, #175)
- Change handling of `--suffix-var` and `--sufix-csv` to prepend with `"-"` as a delimiter if none is included in the string, as was already the case for `--sufix-file`. (#170, #171)
- Include `--suffix-var` string in region names. (#173)

- Improved UI help and verbosity messages. (#166, #167, #170, #179, #180, #182)
- Increase default verbosity level from 1 to 2. (#179)

3.11.2 Fixed

Inference

- Autocrop with upward facing was running with reflected data as its input, resulting in the data being processed upside down and by the wrong conditional model. (#172)
- Remove duplicate leading byte order mark character from evr file output, which was preventing the file from importing into Echoview. (#178)
- Fix \r\n line endings being mapped to \r\r\n on Windows in evl and evr output files. (#178)
- Show error message when importing the evr file into the ev file fails. (#169)
- Fix duplicated Segments tqdm progress bar. (#180)

3.11.3 Added

Inference

- Add `--offset-surface` argument, which allows the surface line to be adjusted by a fixed distance. (#168)

3.12 Version 1.0.0b2

Release date: 2020-06-18. Full commit changelog.

This is a beta pre-release of v1.0.0.

3.12.1 Changed

Inference

- Change default value of `--offset` to 1m. (#159)
- Use a default `--nearfield-cutoff` of 1.7m. (#159, #161)
- Show total run time when inference is finished. (#156)
- Only ever report number of skipped regions if there were some which were skipped. (#156)

3.12.2 Fixed

Inference

- When using the "redact" method for `--lines-during-passive` (the default option), depths were redacted but the timestamps were not, resulting in a temporal offset which accumulated with each passive region. (#155)
- Fix behaviour with `--suffix-file`, so files are written to the filename with the suffix. (#160)
- Fix type of `--offset-top` and `--offset-bottom` arguments from `int` to `float`. (#159)
- Documentation for `--overwrite-ev-lines` argument. (#157)

3.12.3 Added

Inference

- Add ability to specify whether to use recursive search through subdirectory tree, or just files in the specified directory, to both inference.py and ev2csv.py. Add `--no-recursive-dir-search` argument to enable the non-recursive mode. (#158)
- Add option to cap the top or bottom line (depending on orientation) so it cannot go too close to the echosounder, with `--nearfield-cutoff` argument. (#159)
- Add option to skip outputting individual evl lines, with `--no-top-line`, `--no-bottom-line`, `--no-surface-line` arguments. (#162)

3.13 Version 1.0.0b1

Release date: 2020-06-17. Full commit changelog.

This is a beta pre-release of v1.0.0.

3.13.1 Changed

Training

- Built-in line offsets and nearfield line are removed from training targets. (#82)
- Training validation is now against data which is cropped by depth to zoom in on only the "optimal" range of depths (from the shallowest ground truth surface line to the deepest bottom line), using `echofilter.data.transforms.OptimalCropDepth`. (#83, #109)
- Training augmentation stack. (#79, #83, #106, #124)
- Train using normalisation based on the 10th percentile as the zero point and standard deviation robustly estimated from the interdecile range. (#80)
- Use log-avg-exp for `logit_is_passive` and `logit_is_removed`. (#97)
- Exclude data during removed blocks from top and bottom line targets. (#92, #110, #136)
- Seeding of workers and random state during training. (#93, #126)
- Change names of saved checkpoints and log. (#122, #132)
- Save UNet state to checkpoint, not the wrapped model. (#133)

- Change and reduce number of images generated when training. (#95, #98, #99, #101, #108, #112, #114, #127)

Inference

- Change checkpoints available to be used for inference. (#147)
- Change default checkpoint to be dependent on the `--facing` argument. (#147)
- Default line status of output lines changed from `1` to `3`. (#135)
- Default handling of lines during passive data collection changed from implicit `"predict"` to `"redact"`. (#138)
- By default, output logits are smoothed using a Gaussian with width of 1 pixel (relative to the model's latent output space) before being converted into output probibilities. (#144)
- By default, automatically cropping to zoom in on the depth range of interest if the fraction of the depth which could be removed is at least 35% of the original depth. (#149)
- Change default normalisation behaviour to be based on the current input's distribution of Sv values instead of the statistics used for training. (#80)
- Output surface line as an evl file. (f829cb7)
- Output regions as an evr file. (#141, #142, #143)
- By default, when running on a .ev file, the generated lines and regions are imported into the file. (#152)
- Renamed `--csv-suffix` argument to `--suffix-csv`. (#152)
- Improved UI help and verbosity messages. (#81, #129, #137, #145)

Miscellaneous

- Set Sv values outside the range (-1e37, 1e37) to be NaN (previously values lower than -1e6 were set to NaN). (#140)
- Move modules into subpackages. (#104, #130)
- General code tidy up and refactoring. (#85, #88, #89, #94, #96, #146)
- Change code to use the black style. (#86, #87)

3.13.2 Fixed

Training

- Edge-cases when resizing data such as lines crossing; surface lines marked as undefined with value `-10000.99`. (#90)
- Seeding numpy random state for dataloader workers during training. (#93)
- Resume train schedule when resuming training from existing checkpoint. (#120)
- Setting state for RangerVA when resuming training from existing checkpoint. (#121)
- Running LRFinder after everything else is set up for the model. (#131)

Inference

- Exporting raw data in ev2csv required more Echoview parameters to be disabled, such as the minimum value threshold. (#100)

Miscellaneous

- Fixed behaviour when loading data from CSVs with different number of depth samples and range of depths for different rows in the CSV file. (#102, #103)

3.13.3 Added

Training

- New augmentations: RandomCropDepth, RandomGrid, ElasticGrid, (#83, #105, #124)
- Add outputs and loss terms for auxiliary targets: original top and bottom line, variants of the patches mask. (#91)
- Add option to exclude passive and removed blocks from line targets. (#92)
- Interpolation method option added to Rescale, randomly selected for training. (#79)
- More input scaling options. (#80)
- Add option to specify pooling operation for `logit_is_passive` and `logit_is_removed`. (#97)
- Support training on Grand Passage dataset. (#101)
- Support training on multiple datasets. (#111, #113)
- Add `stationary2` dataset which contains both MinasPassage and two copies of GrandPassage with different augmentations, and `mobile+stationary2` dataset. (#111, #113)
- Add conditional model architecture training wrapper. (#116)
- Add outputs for conditional targets to tensorboard. (#125, #134)
- Add stratified data sampler, which preserves the balance between datasets in each training batch. (#117)
- Training process error catching. (#119)
- Training on multiple GPUs on the same node for a single model. (#123, #133)

Inference

- Add `--line-status` argument, which controls the status to use in the evl output for the lines. (#135)
- Add multiple methods of how to handle lines during passive data, and argument `--lines-during-passive` to control which method to use. (#138, #148)
- Add `--offset`, `--offset-top`, `--offset-bottom` arguments, which allows the top and bottom lines to be adjusted by a fixed distance. (#139)
- Write regions to evr file. (#141, #142, #143)
- Add `--logit-smoothing-sigma` argument, which controls the kernel width for Gaussian smoothing applied to the logits before converting to predictions. (#144)
- Generating outputs from conditional models, adding `--unconditioned` argument to disable usage of conditional probability outputs. (#147)

- Add automatic cropping to zoom in on the depth range of interest. Add `--auto-crop-threshold` argument, which controls the threshold for when this occurs. (#149)
- Add `--list-checkpoints` action, which lists the available checkpoints. (#150)
- Fast fail if outputs already exist before processing already begins (and overwrite mode is not enabled). (#151)
- Import generated line and region predictions from the .evl and .evr files into the .ev file and save it with the new lines and regions included. The `--no-ev-import` argument prevents this behaviour. (#152)
- Add customisation of imported lines. The `--suffix-var` argument controls the suffix append to the name of the line variable. The `--overwrite-ev-lines` argument controls whether lines are overwritten if lines already exist with the same name. Also add arguments to customise the colour and thickness of the lines. (#152)
- Add `--suffix-file` argument, will allows a suffix common to all the output files to be set. (#152)

Miscellaneous

- Add `-V` alias for `--version` to all command line interfaces. (#84)
- Loading data from CSV files which contain invalid characters outside the UTF-8 set (seen in the Grand Passage dataset's csv files). (#101)
- Handle raw and masked CSV data of different sizes (occuring in Grand Passage's csv files due to dropped rows containing invalid chararcters). (#101)
- Add seed argument to separation script. (#56)
- Add sample script to extract raw training data from ev files. (#55)

3.14 Version 0.1.4

Release date: 2020-05-19. Full commit changelog.

3.14.1 Added

- Add ability to set orientation of echosounder with `--facing` argument (#77) The orientation is shown to the user if it was automatically detected as upward-facing (#76)

3.15 Version 0.1.3

Release date: 2020-05-16. Full commit changelog.

3.15.1 Fixed

- EVL writer needs to output time to nearest 0.1ms. (#72)

3.15.2 Added

- Add `--suffix` argument to the command line interface of `ev2csv`. (#71)
- Add `--variable-name` argument to `inference.py` (the main command line interface). (#74)

3.16 Version 0.1.2

Release date: 2020-05-14. Full commit changelog.

3.16.1 Fixed

- In `ev2csv`, the files generator needed to be cast as a list to measure the number of files. (#66)
- Echoview is no longer opened during dry-run mode. (#66)
- In `parse_files_in_folders` (affecting `ev2csv`), string inputs were not being handled correctly. (#66)
- Relative paths need to be converted to absolute paths before using them in Echoview. (#68, #69)

3.16.2 Added

- Support hiding or minimizing Echoview while the script is running. The default behaviour is now to hide the window if it was created by the script. The same Echoview window is used throughout the the processing. (#67)

3.17 Version 0.1.1

Release date: 2020-05-12. Full commit changelog.

3.17.1 Fixed

- Padding in echofilter.modules.pathing.FlexibleConcat2d when only one dim size doesn't match. (#64)

3.18 Version 0.1.0

Release date: 2020-05-12. Initial release.

CHAPTER

FOUR

MODULE INDEX

CHAPTER

FIVE

INDEX

INDEX