Echofilter Documentation

Release 1.2.0

Scott C. Lowe

Nov 19, 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-1.2.0.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, in Python 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). This is also true of PowerShell scripts, which have the .ps1 extension.

```
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 in PowerShell:

```
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
```

or:

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

This can be specified multiple times to elevate the verbosity further:

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

or:

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

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

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

Processing a file and sending the output to a log file as well as the terminal by using the `tee` command. Note that on Windows this works in PowerShell and in PowerShell scripts, but not in Windows Command Prompt (cmd) or batch scripts as `tee` is not available:

```
echofilter "path/to/file_or_directory" 2>&1 | tee "path/to/log_file.txt"
```

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 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.

For details about how the Echofilter models were trained, and our findings about their empirical performance, 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.

An overview for of notable model checkpoints available in echofilter are provided below.

echofilter-v1_bifacing_700ep

- Trained on both *upfacing stationary* and *downfacing mobile* data.
- Overall IoU performance of **99.15%** on *downfacing mobile* and 93.0%–94.9% on *upfacing stationary test* data.
- Default model checkpoint.

echofilter-v1_bifacing_300ep

- Trained on both *upfacing stationary* and *downfacing mobile* data.
- Overall IoU performance of 99.02% on *downfacing mobile* and 93.2%–95.0% on *upfacing stationary test* data.

echofilter-v1_bifacing_100ep

- Trained on both *upfacing stationary* and *downfacing mobile* data.
- Overall IoU performance of 98.93% on *downfacing mobile* and **93.5%**–94.9% on *upfacing stationary test* data.
- *Sample* outputs on *upfacing stationary* data were thoroughly verified via manual inspection by trained analysts.

echofilter-v1_upfacing_600ep

- Trained on *upfacing stationary* data only.
- Overall IoU performance of 92.1%–**95.1%** on *upfacing stationary test* data.

echofilter-v1_upfacing_200ep

- Trained on *upfacing stationary* data only.
- Overall IoU performance of 93.3%–95.1% on *upfacing stationary test* data.
- *Sample* outputs thoroughly were thoroughly verified via manual inspection by trained analysts.

echofilter-v0.5_downfacing_300ep

- Trained on *downfacing mobile* data only.

1.5.2 Training Datasets

The machine learning model was trained on upfacing stationary and downfacing mobile data provided by Fundy Ocean Research Centre for Energy (FORCE). The training and evaluation data is available for download. Queries regarding dataset access should be directed to FORCE, info@fundyforce.ca.

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

1.6 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] [--show-cache-dir] [--list-checkpoints]
                  [--list-colors [{alphabetic,full,full-alphabetic,xkcd,xkcd-alphabetic}
↪]]
                  [-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]]
```

(continues on next page)

(continued from previous page)

```
                  [--suffix-csv SUFFIX_CSV] [--keep-ext]
                  [--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.

--show-cache-dir Show the path to the cache directory and exit.

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

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

Show the available line color names and exit. The available color palette can be viewed at https://matplotlib.org/stable/gallery/color/named_colors.html#css-colors. 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 alphabetic` to view it in alphabetic order. The default ordering is by hue. To show the full palette, run as `--list-colors full` or `--list-colors full-alphabetic`.

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: `"echofilter-v1_bifacing_700ep"`.

--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

API REFERENCE

3.1 echofilter package

3.1.1 Subpackages

echofilter.data package

Dataset creation and manipulation.

Submodules

echofilter.data.dataset module

Convert echograms into Pytorch dataset.

Tools for converting a dataset of echograms (transects) into a Pytorch dataset and sampling from it.

class echofilter.data.dataset.**ConcatDataset**(*datasets: Iterable[torch.utils.data.dataset.Dataset]*)

Bases: torch.utils.data.dataset.ConcatDataset

Dataset as a concatenation of multiple TransectDatasets.

This class is useful to assemble different existing datasets.

Parameters **datasets** (*sequence*) – List of datasets to be concatenated.

Notes

A subclass of torch.utils.data.ConcatDataset which supports the initialise_datapoints method.

cumulative_sizes: List[int]

datasets: List[torch.utils.data.dataset.Dataset[torch.utils.data.dataset.T_co]]

initialise_datapoints()

class echofilter.data.dataset.**StratifiedRandomSampler**(*data_source*)

Bases: torch.utils.data.sampler.Sampler

Sample elements randomly without repetition, stratified across datasets.

Parameters **data_source** (*torch.utils.data.ConcatDataset*) – Dataset to sample from. Must possess a cumulative_sizes attribute.

property num_samples

class echofilter.data.dataset.**TransectDataset**(*transect_paths*, *window_len=128*, *p_scale_window=0*, *window_sf=2*, *num_windows_per_transect=0*, *use_dynamic_offsets=True*, *crop_depth=None*, *transform=None*, *remove_nearfield=True*, *nearfield_distance=1.7*, *nearfield_visible_dist=0.0*, *remove_offset_turbulence=0*, *remove_offset_bottom=0*)

Bases: torch.utils.data.dataset.Dataset

Load a collection of transects as a PyTorch dataset.

Parameters

- **transect_paths** (*list*) – Absolute paths to transects.
- **window_len** (*int*) – Width (number of timestamps) to load. Default is `128`.
- **p_scale_window** (*float, optional*) – Probability of rescaling window. Default is `0`, which results in no randomization of the window widths.
- **window_sf** (*float, optional*) – Maximum window scale factor. Scale factors will be log-uniformly sampled in the range `1/window_sf` to `window_sf`. Default is `2`.
- **num_windows_per_transect** (*int*) – Number of windows to extract for each transect. Start indices for the windows will be equally spaced across the total width of the transect. If this is `0`, the number of windows will be inferred automatically based on `window_len` and the total width of the transect, resulting in a different number of windows for each transect. Default is `0`.
- **use_dynamic_offsets** (*bool*) – Whether starting indices for each window should be randomly offset. Set to `True` for training and `False` for testing. Default is `True`.
- **crop_depth** (*float*) – Maximum depth to include, in metres. Deeper data will be cropped away. Default is `None`.
- **transform** (*callable*) – Operations to perform to the dictionary containing a single sample. These are performed before generating the turbulence/bottom/overall mask. Default is `None`.
- **remove_nearfield** (*bool, optional*) – Whether to remove turbulence and bottom lines affected by nearfield removal. If `True` (default), targets for the line near to the sounder (bottom if upward facing, turbulence otherwise) which are closer than or equal to a distance of `nearfield_distance` become reduced to `nearfield_visible_dist`.
- **nearfield_distance** (*float, optional*) – Nearfield distance in metres. Regions closer than the nearfield may have been masked out from the dataset, but their effect will be removed from the targets if `remove_nearfield=True`. Default is `1.7`.
- **nearfield_visible_dist** (*float, optional*) – The distance at which the effect of being to close to the sounder is obvious to the naked eye, and hence the distance which nearfield will be mapped to if `remove_nearfield=True`. Default is `0.0`.
- **remove_offset_turbulence** (*float, optional*) – Line offset built in to the turbulence line. If given, this will be removed from the samples within the dataset. Default is `0`.
- **remove_offset_bottom** (*float, optional*) – Line offset built in to the bottom line. If given, this will be removed from the samples within the dataset. Default is `0`.

initialise_datapoints()

Parse `transect_paths` to generate sampling windows for each transect.

Manually calling this method will resample the transect offsets and widths if they were randomly generated.

echofilter.data.dataset.**fixup_dataset_sample**(*sample*, *remove_nearfield=True*, *nearfield_distance=1.7*, *nearfield_visible_dist=0.0*, *remove_offset_turbulence=0.0*, *remove_offset_bottom=0.0*, *crop_depth=None*, *transform=None*)

Handle a dataset transect sample.

Parameters

- **sample** (*dict*) – Transect dictionary.
- **remove_nearfield** (*bool, default=True*) – Whether to remove turbulence and bottom lines affected by nearfield removal. If `True` (default), targets for the line near to the sounder (bottom if upward facing, turbulence otherwise) which are closer than or equal to a distance of `nearfield_distance` become reduced to `nearfield_visible_dist`.
- **nearfield_distance** (*float, default=1.7*) – Nearfield distance in metres. Regions closer than the nearfield may have been masked out from the dataset, but their effect will be removed from the targets if `remove_nearfield=True`.
- **nearfield_visible_dist** (*float, default=0*) – The distance at which the effect of being to close to the sounder is obvious to the naked eye, and hence the distance which nearfield will be mapped to if `remove_nearfield=True`.
- **remove_offset_turbulence** (*float, default=0*) – Line offset built in to the turbulence line. If given, this will be removed from the samples within the dataset.
- **remove_offset_bottom** (*float, default=0*) – Line offset built in to the bottom line. If given, this will be removed from the samples within the dataset.
- **crop_depth** (*float*) – Maximum depth to include, in metres. Deeper data will be cropped away. Default is `None`.
- **transform** (*callable, optional*) – Operations to perform to the dictionary containing a single sample. These are performed before generating the turbulence/bottom/overall mask.

Returns Like `sample`, but contents fixed.

Return type dict

echofilter.data.transforms module

Transformations and augmentations to be applied to echogram transects.

class echofilter.data.transforms.**ColorJitter**(*brightness=0*, *contrast=0*)

Bases: `object`

Randomly change the brightness and contrast of a normalized image.

Note that changes are made inplace.

Parameters

- **brightness** (*float or tuple of float (min, max)*) – How much to jitter brightness. `brightness_factor` is chosen uniformly from `[-brightness, brightness]` or the given `[min, max]`. `brightness_factor` is then added to the image.

- **contrast** (*float or tuple of float (min, max)*) – How much to jitter contrast. `contrast_factor` is chosen uniformly from `[max(0, 1 - contrast), 1 + contrast]` or the given `[min, max]`. Should be non negative numbers.

class echofilter.data.transforms.**Normalize**(*center*, *deviation*, *robust2stdev=True*)

Bases: `object`

Normalize offset and scaling of image (mean and standard deviation).

Note that changes are made inplace.

Parameters

- **center** (*{"mean", "median", "pc10"} or float*) – If a float, a pre-computed centroid measure of the distribution of samples, such as the pixel mean. If a string, a method to use to determine the center value.
- **deviation** (*{"stdev", "mad", "iqr", "idr", "i7r"} or float*) – If a float, a pre-computed deviation measure of the distribution of samples. If a string, a method to use to determine the deviation.
- **robust2stdev** (*bool, optional*) – Whether to convert robust measures to estimates of the standard deviation. Default is `True`.

class echofilter.data.transforms.**OptimalCropDepth**

Bases: `object`

A transform which crops a sample depthwise to focus on the water column.

The output contains only the space between highest surface and deepest seafloor line measurements.

class echofilter.data.transforms.**RandomCropDepth**(*p_crop_is_none=0.1*, *p_crop_is_optimal=0.1*, *p_crop_is_close=0.4*, *p_nearfield_side_crop=0.5*, *fraction_close=0.25*)

Bases: `object`

Randomly crop a sample depthwise.

Parameters

- **p_crop_is_none** (*float, optional*) – Probability of not doing any crop. Default is `0.1`.
- **p_crop_is_optimal** (*float, optional*) – Probability of doing an "optimal" crop, running `optimal_crop_depth`. Default is `0.1`.
- **p_crop_is_close** (*float, optional*) – Probability of doing crop which is zoomed in and close to the "optimal" crop, running `optimal_crop_depth`. Default is `0.4`. If neither no crop, optimal, nor close-to-optimal crop is selected, the crop is randomly sized over the full extent of the range of depths.
- **p_nearfield_side_crop** (*float, optional*) – Probability that the nearfield side is cropped. Default is `0.5`.
- **fraction_close** (*float, optional*) – Fraction by which crop is increased/decreased in either direction when doing a close to optimal crop. Default is `0.25`.

class echofilter.data.transforms.**RandomCropWidth**(*max_crop_fraction*)

Bases: `object`

Randomly crop a sample in the width dimension.

Parameters **max_crop_fraction** (*float*) – Maximum amount of material to crop away, as a fraction of the total width. The `crop_fraction` will be sampled uniformly from the range `[0, max_crop_fraction]`. The crop is always centred.

class echofilter.data.transforms.**RandomElasticGrid**(*output_size*, *p=0.5*, *sigma=8.0*, *alpha=0.05*, *order=1*)

Bases: *echofilter.data.transforms.Rescale*

Resample data onto a new grid, elastically deformed from the original grid.

Parameters

- **output_size** (*tuple or int or None*) – Desired output size. If tuple, output is matched to output_size. If int, output is square. If `None`, the size remains unchanged from the input.
- **p** (*float, optional*) – Probability of performing the RandomGrid operation. Default is `0.5`.
- **sigma** (*float, optional*) – Gaussian filter kernel size. Default is `8.0`.
- **alpha** (*float, optional*) – Maximum size of image distortions, relative to the length of the side of the image. Default is `0.05`.
- **order** (*int or None, optional*) – Order of the interpolation, for both image and vector elements. For images-like components, the interpolation is 2d. The following values are supported:
  - 0: Nearest-neighbor
  - 1: Linear (default)
  - 2: Quadratic
  - 3: Cubic

  If `None`, the order is randomly selected from the set `{1, 2, 3}`.

class echofilter.data.transforms.**RandomGridSampling**(**args*, *p=0.5*, ***kwargs*)

Bases: *echofilter.data.transforms.Rescale*

Resample data onto a new grid, which is randomly resampled.

Parameters

- **output_size** (*tuple or int*) – Desired output size. If tuple, output is matched to output_size. If int, output is square.
- **p** (*float, optional*) – Probability of performing the RandomGrid operation. Default is `0.5`.
- **order** (*int or None, optional*) – Order of the interpolation, for both image and vector elements. For images-like components, the interpolation is 2d. The following values are supported:
  - 0: Nearest-neighbor
  - 1: Linear (default)
  - 2: Quadratic
  - 3: Cubic

  If `None`, the order is randomly selected from the set `{0, 1, 3}`.

class echofilter.data.transforms.**RandomReflection**(*axis=0*, *p=0.5*)

Bases: `object`

Randomly reflect a sample.

Parameters

- **axis** (*int, optional*) – Axis to reflect. Default is 0.
- **p** (*float, optional*) – Probability of reflection. Default is 0.5.

class echofilter.data.transforms.**ReplaceNan**(*nan_val=0.0*)

Bases: object

Replace NaNs with a finite float value.

Parameters **nan_val** (*float, optional*) – Value to replace NaNs with. Default is `0.0`.

class echofilter.data.transforms.**Rescale**(*output_size*, *order=1*)

Bases: object

Rescale the image(s) in a sample to a given size.

Parameters

- **output_size** (*tuple or int*) – Desired output size. If tuple, output is matched to output_size. If int, output is square.
- **order** (*int or None, optional*) – Order of the interpolation, for both image and vector elements. For images-like components, the interpolation is 2d. The following values are supported:
  - 0: Nearest-neighbor
  - 1: Linear (default)
  - 2: Quadratic
  - 3: Cubic

  If None, the order is randomly selected as either `0` or `1`.

order2kind = {0: 'nearest', 1: 'linear', 2: 'quadratic', 3: 'cubic'}

echofilter.data.transforms.**optimal_crop_depth**(*transect*)

Crop a sample depthwise to surround the water column.

The crop is to contain only the space between highest surface and deepest seafloor.

Parameters **transect** (*dict*) – Transect dictionary.

echofilter.data.utils module

Utility functions for dataset.

echofilter.data.utils.**worker_seed_fn**(*worker_id*)

Seed builtin random and numpy with `torch.randint()`.

A worker initialization function for `torch.utils.data.DataLoader` objects which seeds builtin random and numpy with `torch.randint()` (which is stable if torch is manually seeded in the main program).

Parameters **worker_id** (*int*) – The ID of the worker.

echofilter.data.utils.**worker_staticseed_fn**(*worker_id*)

Seed builtin random, numpy, and torch with `worker_id`.

A worker initialization function for `torch.utils.data.DataLoader` objects which produces the same seed for builtin random, numpy, and torch every time, so it is the same for every epoch.

Parameters **worker_id** (*int*) – The ID of the worker.

echofilter.nn package

Neural network building blocks.

Subpackages

echofilter.nn.modules package

Submodules

echofilter.nn.modules.activations module

Pytorch activation functions.

Swish and Mish implementations taken from https://github.com/fastai/fastai2 under the Apache License Version 2.0.

class echofilter.nn.modules.activations.**HardMish**(*inplace=True*)

Bases: torch.nn.modules.module.Module

A second-order approximation to the mish activation function.

Notes

https://forums.fast.ai/t/hard-mish-activation-function/59238

extra_repr()

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

class echofilter.nn.modules.activations.**HardSwish**(*inplace=True*)

Bases: torch.nn.modules.module.Module

A second-order approximation to the swish activation function.

See https://arxiv.org/abs/1905.02244

extra_repr()

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

class echofilter.nn.modules.activations.**Mish**

Bases: torch.nn.modules.module.Module

Apply the mish function elementwise.

mish(x) = x * tanh(softplus(x)) = x * tanh(ln(1 + exp(x)))

See https://arxiv.org/abs/1908.08681

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

class echofilter.nn.modules.activations.**Swish**

Bases: torch.nn.modules.module.Module

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

echofilter.nn.modules.activations.**mish**(*x*)

Apply the mish function elementwise.

mish(x) = x * tanh(softplus(x)) = x * tanh(ln(1 + exp(x)))

See https://arxiv.org/abs/1908.08681

echofilter.nn.modules.activations.**str2actfnfactory**(*actfn_name*)

Map an activation function name to a factory which generates that actfun.

Parameters **actfn_name** (*str*) – Name of the activation function.

Returns A generator which yields a subclass of `torch.nn.Module`.

Return type callable

`echofilter.nn.modules.activations.`**swish**(*x*, *inplace=False*)

echofilter.nn.modules.blocks module

Blocks of modules.

class `echofilter.nn.modules.blocks.`**MBConv**(*in_channels*, *out_channels=None*, *expansion=6*, *se_reduction=4*, *fused=False*, *residual=True*, *actfn='InplaceReLU'*, *bias=False*, ***conv_args*)

Bases: `torch.nn.modules.module.Module`

MobileNet style inverted residual block.

See https://arxiv.org/abs/1905.11946 and https://arxiv.org/abs/1905.02244.

Parameters

- **in_channels** (*int*) – Number of input channels.
- **out_channels** (*int, optional*) – Number of output channels. Default is to match `in_channels`.
- **expansion** (*int or float, optional*) – Exansion factor for the inverted-residual bottleneck. Default is `6`.
- **se_reduction** (*int, optional*) – Reduction factor for squeeze-and-excite block. Default is `4`. Set to `None` or `0` to disable squeeze-and-excitation.
- **fused** (*bool, optional*) – If `True`, the pointwise and depthwise convolution are fused together into a single regular convolution. Default is `False` (a depthwise separable convolution).
- **residual** (*bool, optional*) – If `True`, the block is residual with a skip-through connection. Default is `True`.
- **actfn** (*str or callable, optional*) – An activation class or similar generator. Default is an inplace ReLU activation. If this is a string, it is mapped to a generator with `activations.str2actfnfactory`.
- **bias** (*bool, optional*) – If `True`, the main convolution has a bias term. Default is `False`. Note that the pointwise convolutions never have bias terms.
- ****conv_args** – Additional arguments, such as kernel_size, stride, and padding, which will be passed to the convolution module.

extra_repr()

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

forward(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

> **Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

training: bool

class echofilter.nn.modules.blocks.**SqueezeExcite**(*in_channels*, *reduction=4*, *actfn='InplaceReLU'*)

Bases: torch.nn.modules.module.Module

Squeeze and excitation block.

See https://arxiv.org/abs/1709.01507

Parameters

- **in_channels** (*int*) – Number of input (and output) channels.
- **reduction** (*int or float, optional*) – Compression factor for the number of channels in the squeeze and excitation attention module. Default is 4.
- **actfn** (*str or callable, optional*) – An activation class or similar generator. Default is an inplace ReLU activation. If this is a string, it is mapped to a generator with `activations.str2actfnfactory`.

forward(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

> **Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

training: bool

echofilter.nn.modules.conv module

Convolutional layers.

class echofilter.nn.modules.conv.**Conv2dSame**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding='same'*, *dilation=1*, ***kwargs*)

Bases: torch.nn.modules.conv.Conv2d

2D Convolutions with same padding option.

Same padding will only produce an output size which matches the input size if the kernel size is odd and the stride is 1.

bias: Optional[torch.Tensor]

dilation: Tuple[int, ...]

groups: int

kernel_size: Tuple[int, ...]

out_channels: int

output_padding: Tuple[int, ...]

padding: Union[str, Tuple[int, ...]]

padding_mode: str

stride: Tuple[int, ...]

transposed: bool

weight: torch.Tensor

class echofilter.nn.modules.conv.**DepthwiseConv2d**(*in_channels*, *kernel_size=3*, *stride=1*, *padding='same'*, *dilation=1*, ***kwargs*)

Bases: torch.nn.modules.conv.Conv2d

2D Depthwise Convolution.

bias: Optional[torch.Tensor]

dilation: Tuple[int, ...]

groups: int

kernel_size: Tuple[int, ...]

out_channels: int

output_padding: Tuple[int, ...]

padding: Union[str, Tuple[int, ...]]

padding_mode: str

stride: Tuple[int, ...]

transposed: bool

weight: torch.Tensor

class echofilter.nn.modules.conv.**GaussianSmoothing**(*channels*, *kernel_size*, *sigma*, *padding='same'*, *pad_mode='replicate'*, *ndim=2*)

Bases: torch.nn.modules.module.Module

Apply gaussian smoothing on a 1d, 2d or 3d tensor.

Filtering is performed seperately for each channel in the input using a depthwise convolution.

Parameters

- **channels** (*int or sequence*) – Number of channels of the input tensors. Output will have this number of channels as well.
- **kernel_size** (*int or sequence*) – Size of the gaussian kernel.
- **sigma** (*float or sequence*) – Standard deviation of the gaussian kernel.
- **padding** (*int or sequence or "same", optional*) – Amount of padding to use, for each side of each dimension. If this is "same" (default) the amount of padding will be set automatically to ensure the size of the tensor is unchanged.

- **pad_mode** (*str, optional*) – Padding mode. See `torch.nn.functional.pad()` for options. Default is `"replicate"`.
- **ndim** (*int, optional*) – The number of dimensions of the data. Default value is 2 (spatial).

Notes

Based on https://discuss.pytorch.org/t/is-there-anyway-to-do-gaussian-filtering-for-an-image-2d-3d-in-pytorch/12351/10

forward(*input*)

Apply gaussian filter to input.

Parameters **input** (*torch.Tensor*) – Input to apply gaussian filter on.

Returns **filtered** – Filtered output, the same size as the input.

Return type torch.Tensor

training: bool

class echofilter.nn.modules.conv.**PointwiseConv2d**(*in_channels*, *out_channels*, ***kwargs*)

Bases: torch.nn.modules.conv.Conv2d

2D Pointwise Convolution.

bias: Optional[torch.Tensor]

dilation: Tuple[int, ...]

groups: int

kernel_size: Tuple[int, ...]

out_channels: int

output_padding: Tuple[int, ...]

padding: Union[str, Tuple[int, ...]]

padding_mode: str

stride: Tuple[int, ...]

transposed: bool

weight: torch.Tensor

class echofilter.nn.modules.conv.**SeparableConv2d**(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding='same'*, *dilation=1*, *groups=1*, ***kwargs*)

Bases: torch.nn.modules.module.Module

2D Depthwise Separable Convolution.

foward(*x*)

training: bool

echofilter.nn.modules.pathing module

Connectors and pathing modules.

class echofilter.nn.modules.pathing.**FlexibleConcat2d**

Bases: torch.nn.modules.module.Module

Concatenate two inputs of nearly the same shape.

forward(*x1*, *x2*)

Forward step.

Parameters

- **x1** (*torch.Tensor*) – Tensor, possibly smaller than `x2`.
- **x2** (*torch.Tensor*) – Tensor, at least as large as `x1`.

Returns Concatenated `x1` (padded if necessary) and `x2`, along dimension `1`.

Return type torch.Tensor

training: bool

class echofilter.nn.modules.pathing.**ResidualConnect**(*in_channels*, *out_channels*)

Bases: torch.nn.modules.module.Module

Joins up a residual connection, correcting for changes in number of channels.

forward(*residual*, *passed_thru*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

echofilter.nn.modules.utils module

nn.modules utility functions.

echofilter.nn.modules.utils.**init_cnn**(*m*)

Initialize biases and weights for a CNN layer.

Uses a Kaiming normal distribution for the weight and 0 for biases.

Function is applied recursively within the module.

Parameters **m** (*torch.nn.Module*) – Module

echofilter.nn.modules.utils.**same_to_padding**(*kernel_size*, *stride=1*, *dilation=1*, *ndim=None*)

Determine the amount of padding to use for a convolutional layer.

Parameters

- **kernel_size** (*int or sequence*) – Size of kernel for each dimension.

- **stride** (*int or sequence, optional*) – Amount of stride to apply in each dimension of the kernel. If `stride` is an int, the same value is applied for each dimension. Default is 1.
- **dilation** (*int or sequence, optional*) – Amount of dilation to apply in each dimension of the kernel. If `dilation` is an int, the same value is applied for each dimension. Default is 1.
- **ndim** (*int or None, optional*) – Number of dimensions of kernel to pad. If `None` (default), the number of dimensions is inferred from the number of dimensions to `kernel_size`.

Returns **padding** – Amount of padding to apply to each dimension before convolving with the kernel in order to preserve the size of input.

Return type tuple

Submodules

echofilter.nn.unet module

U-Net model.

class echofilter.nn.unet.**Down**(*mode='max'*, *compress_dims=True*)

Bases: `torch.nn.modules.module.Module`

Downscaling layer, downsampling by a factor of two in one or more dimensions.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

class echofilter.nn.unet.**UNet**(*in_channels*, *out_channels*, *initial_channels=32*, *bottleneck_channels=None*, *n_block=4*, *unet_expansion_factor=2*, *expand_only_on_down=False*, *blocks_per_downsample=1*, *blocks_before_first_downsample=1*, *always_include_skip_connection=True*, *deepest_inner='identity'*, *intrablock_expansion=6*, *se_reduction=4*, *downsampling_modes='max'*, *upsampling_modes='bilinear'*, *depthwise_separable_conv=True*, *residual=True*, *actfn='InplaceReLU'*, *kernel_size=5*)

Bases: `torch.nn.modules.module.Module`

UNet model.

Parameters

- **in_channels** (*int*) – Number of input channels.
- **out_channels** (*int*) – Number of output channels.
- **initial_channels** (*int, optional*) – Number of latent channels to output from the initial convolution facing the input layer. Default is 32.

- **bottleneck_channels** (*int, optional*) – Number of channels to output from the first block, before the first unet downsampling step can occur. Default is the same as `initial_channels`.
- **n_block** (*int, optional*) – Number of blocks, both up and down. Default is `4`.
- **unet_expansion_factor** (*int or float, optional*) – Channel expansion factor between unet blocks. Default is `2`.
- **expand_only_on_down** (*bool, optional*) – Whether to only apply `unet_expansion_factor` on unet blocks which actually containg a down/up sampling component, and not on vanilla blocks. Default is `False`.
- **blocks_per_downsample** (*int or sequence, optional*) – Block interval between dowsampling steps in the unet. If this is a sequence, it corresponds to the number of blocks for each spatial dimension. Default is `1`.
- **blocks_before_first_downsample** (*int, optional*) – Number of blocks to use before and after the main unet structure. Must be at least `1`. Default is `1`.
- **always_include_skip_connection** (*bool, optional*) – If `True`, a skip connection is included between all blocks equally far from the start and end of the UNet. If `False`, skip connections are only used between downsampling and upsampling operations. Default is `True`.
- **deepest_inner** (*{callable, "horizontal_block", "identity", None}, optional*) – A layer which should be applied at the deepest part of the network, before the first upsampling step. The parameter should either be a pre-instantiated layer, or the string `"horizontal_block"`, to indicate an additional block as generated by the `horizontal_block_factory`. If it is the string `"identity"` or `None` (default), no additional layer is included at the deepest point before upsampling begins.
- **intrablock_expansion** (*int or float, optional*) – Channel expansion factor within inverse residual block. Default is `6`.
- **se_reduction** (*int or float, optional*) – Channel reduction factor within squeeze and excite block. Default is `4`.
- **downsampling_modes** (*{"max", "avg", "stride"} or sequence, optional*) – The downsampling mode to use. If this is a string, the same downsampling mode is used for every downsampling step. If it is a sequence, it should contain a string for each downsampling step. If the input sequence is too short, the final value will be used for all remaining downsampling steps. Default is `"max"`.
- **upsampling_modes** (*str or sequence, optional*) – The upsampling mode to use. If this is a string, it must be `"conv"`, or something supported by `torch.nn.Upsample`; the same upsampling mode is used for every upsampling step. If it is a sequence, it should contain a string for each upsampling step. If the input sequence is too short, the final value will be used for all remaining upsampling steps. Default is `"bilinear"`.
- **depthwise_separable_conv** (*bool, optional*) – Whether to use depthwise separable convolutions in the MBConv block. Otherwise, the depth and pointwise convolutions are fused together into a regular convolution. Default is `True`.
- **residual** (*bool, optional*) – Whether to use a residual architecture for the MBConv blocks. Default is `True`.
- **actfn** (*str, optional*) – Name of the activation function to use. Default is `"InplaceReLU"`.
- **kernel_size** (*int, optional*) – Size of convolution kernel to use. Default is `5`.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

class echofilter.nn.unet.**UNetBlock**(*in_channels*, *horizontal_block_factory*, *n_block=1*, *block_expansion_factor=2*, *expand_only_on_down=False*, *blocks_per_downsample=1*, *blocks_before_first_downsample=0*, *always_include_skip_connection=True*, *deepest_inner='identity'*, *downsampling_modes='max'*, *upsampling_modes='bilinear'*, *_i_block=0*, *_i_down=0*)

Bases: torch.nn.modules.module.Module

Create a (cascading set of) UNet block(s).

Each block performs the steps:

- Store input to be used in skip connection
- Down step
- Horizontal block
- <Recursion>
- Up step
- Concatenate with skip connection
- Horizontal block

Where <Recursion> is a call generating a child UNetBlock instance.

Parameters

- **in_channels** (*int*) – Number of input channels to this block.
- **horizontal_block_factory** (*callable*) – A torch.nn.Module constructor or function which returns a block of layers. The resulting module must accept `in_channels` and `out_channels` as its first two arguments.
- **n_block** (*int, optional*) – The number of nested UNetBlocks to use. Default is 1 (no nesting).
- **block_expansion_factor** (*int or float, optional*) – Expansion factor for the number of channels between nested UNetBlocks. Default is 2.
- **expand_only_on_down** (*bool, optional*) – Whether to exand the number of channels only when one of the spatial dimensions is compressed. Default is `False`.
- **blocks_per_downsample** (*int or sequence, optional*) – How many blocks to include between each downsample operation. This can be a tuple of values for each spatial dimension, or an int which uses the same value for each spatial dimension. Default is 1.
- **blocks_before_first_downsample** (*int or sequence, optional*) – How many blocks to include before the first spatial downsampling occurs. Default is 1.

- **always_include_skip_connection** (*bool, optional*) – If `True`, a skip connection is included even if no dimensions were downsampled in this block. Default is `True`.
- **deepest_inner** (*{callable, "horizontal_block", "identity", None}, optional*) – A layer which should be applied at the deepest part of the network, before the first upsampling step. The parameter should either be a pre-instantiated layer, or the string `"horizontal_block"`, to indicate an additional block as generated by the `horizontal_block_factory`. If it is the string `"identity"` or `None` (default), no additional layer is included at the deepest point before upsampling begins.
- **downsampling_modes** (*{"max", "avg", "stride"} or sequence, optional*) – The downsampling mode to use. If this is a string, the same downsampling mode is used for every downsampling step. If it is a sequence, it should contain a string for each downsampling step. If the input sequence is too short, the final value will be used for all remaining downsampling steps. Default is `"max"`.
- **upsampling_modes** (*str or sequence, optional*) – The upsampling mode to use. If this is a string, it must be `"conv"`, or something supported by `torch.nn.Upsample`; the same upsampling mode is used for every upsampling step. If it is a sequence, it should contain a string for each upsampling step. If the input sequence is too short, the final value will be used for all remaining upsampling steps. Default is `"bilinear"`.
- **_i_block** (*int, optional*) – The current block number. Used internally to track recursion. Default is `0`.
- **_i_down** (*int, optional*) – Used internally to track downsampling depth. Default is `0`.

Notes

This class is defined recursively, and will instantiate itself as its own child until the number of blocks has been satisfied.

forward(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: `bool`

class echofilter.nn.unet.**Up**(*in_channels=None*, *up_dims=True*, *mode='bilinear'*)

Bases: `torch.nn.modules.module.Module`

Upscaling layer, upsampling by a factor of two in one or more dimensions.

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

echofilter.nn.utils module

echofilter.nn utility functions.

class echofilter.nn.utils.**TensorDict**(*tensors=None*)

Bases: torch.nn.modules.container.ParameterDict

Hold tensors in a dictionary.

TensorDict can be indexed like a regular Python dictionary, but implements methods such as `to` which operate on all elements within it.

TensorDict is an **ordered** dictionary that respects

- the order of insertion, and
- in `update()`, the order of the merged `OrderedDict` or another *TensorDict* (the argument to `update()`).

Note that `update()` with other unordered mapping types (e.g., Python's plain `dict`) does not preserve the order of the merged mapping.

Parameters **tensors** (*iterable*) – A mapping (dictionary) of (string : torch.Tensor) or an iterable of key-value pairs of type (string, torch.Tensor)

detach()

detach_()

extra_repr()

Set the extra representation of the module

To print customized extra information, you should re-implement this method in your own modules. Both single-line and multi-line strings are acceptable.

echofilter.nn.utils.**count_parameters**(*model*, *only_trainable=True*)

Count the number of (trainable) parameters within a model and its children.

Parameters

- **model** (*torch.nn.Model*) – The model.
- **only_trainable** (*bool, default=True*) – Whether the count should be restricted to only trainable parameters (ones which require grad), otherwise all parameters are included. Default is `True`.

Returns Total number of (trainable) parameters possessed by the model.

Return type int

echofilter.nn.utils.**logavgexp**(*input*, *dim*, *keepdim=False*, *temperature=None*, *internal_dtype=torch.float32*)

Take the log-average-exp.

Returns the log of meaned exponentials of each row of the `input` tensor in the given dimension `dim`. The computation is numerically stabilized.

If `keepdim` is `True`, the output tensor is of the same size as `input` except in the dimension `dim` where it is of size 1. Otherwise, `dim` is squeezed (see `torch.squeeze()`), resulting in the output tensor having 1 fewer dimension.

Parameters

- **input** (*torch.Tensor*) – The input tensor.
- **dim** (*int*) – The dimension to reduce.
- **keepdim** (*bool, optional*) – Whether the output tensor has `dim` retained or not. Default is `False`.
- **temperature** (*float or None, optional*) – A temperature which is applied to the log-its. Temperatures must be positive. Temperatures greater than `1` make the result closer to the average of `input`, whilst temperatures `0<t<1` make the result closer to the maximum of `input`. If `None` (default) or `1`, no temperature is applied.
- **internal_dtype** (*torch.dtype, optional*) – A data type which the `input` will be cast as before computing the log-sum-exp step. Default is `torch.float32`.

Returns The log-average-exp of `input`.

Return type torch.Tensor

`echofilter.nn.utils.`**seed_all**(*seed=None*, *only_current_gpu=False*, *mirror_gpus=False*)

Initialize the RNGs for random, numpy, and both CPU and GPU(s) for torch.

Parameters

- **seed** (*int, optional)*) – Seed value to use for the random number generators. If `seed` is `None` (default), seeds are picked at random using the methods built in to each RNG.
- **only_current_gpu** (*bool, default=False*) – Whether to only re-seed the current cuda device, or to seed all of them.
- **mirror_gpus** (*bool, default=False*) – Whether all cuda devices should receive the same seed, or different seeds. If `mirror_gpus` is `False` and `seed` is not `None`, each device receives a different but deterministically determined seed. Default is `False`.

Notes

Note that we override the settings for the cudnn backend whenever this function is called. If `seed` is not `None`, we set:

```
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
```

in order to ensure experimental results behave deterministically and are repeatible. However, enabling deterministic mode may result in an impact on performance. See link for more details. If `seed` is `None`, we return the cudnn backend to its performance-optimised default settings of:

```
torch.backends.cudnn.deterministic = False
torch.backends.cudnn.benchmark = True
```

echofilter.nn.wrapper module

Model wrapper.

class echofilter.nn.wrapper.**Echofilter**(*model*, *top='boundary'*, *bottom='boundary'*, *mapping=None*, *reduction_ispassive='logavgexp'*, *reduction_isremoved='logavgexp'*, *conditional=False*)

Bases: torch.nn.modules.module.Module

Echofilter logit mapping wrapper.

Parameters

- **model** (torch.nn.Module) – The model backbone, which converts inputs to logits.
- **top** (*str, optional*) – Type of output for top line and surface line. If "mask", the top output corresponds to logits, which are converted into probabilities with sigmoid. If "boundary" (default), the output corresponds to logits for the location of the line, which is converted into a probability mask using softmax and cumsum.
- **bottom** (*str, optional*) – As for top, but for the bottom line. Default is "boundary".
- **mapping** (*dict or None, optional*) – Mapping from logit names to output channels provided by model. If None, a default mapping is used. The mapping is stored as self.mapping.
- **reduction_ispassive** (*str, default="logavgexp"*) – Method used to reduce the depths dimension for the "logit_is_passive" output.
- **reduction_isremoved** (*str , default="logavgexp"*) – Method used to reduce the depths dimension for the "logit_is_removed" output.
- **conditional** (*bool, optional*) – Whether to build a conditional model as well as an unconditional model. If True, there are additional logits in the call output named "x|downfacing" and "x|upfacing", in addition to "x". For instance, "p_is_above_turbulence|downfacing". Default is False.

aliases = [('top', 'turbulence')]

forward(*x*, *output_device=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

class echofilter.nn.wrapper.**EchofilterLoss**(*reduction='mean'*, *conditional=False*, *turbulence_mask=1.0*, *bottom_mask=1.0*, *removed_segment=1.0*, *passive=1.0*, *patch=1.0*, *overall=0.0*, *surface=1.0*, *auxiliary=1.0*, *ignore_lines_during_passive=False*, *ignore_lines_during_removed=True*, *ignore_surface_during_passive=False*, *ignore_surface_during_removed=True*)

Bases: torch.nn.modules.loss._Loss

Evaluate loss for an Echofilter model.

Parameters

- **reduction** ("mean" or "sum", optional) – The reduction method, which is used to collapse batch and timestamp dimensions. Default is "mean".
- **turbulence_mask** (*float, optional*) – Weighting for turbulence line/mask loss term. Default is 1.0.
- **bottom_mask** (*float, optional*) – Weighting for bottom line/mask loss term. Default is 1.0.
- **removed_segment** (*float, optional*) – Weighting for is_removed loss term. Default is 1.0.
- **passive** (*float, optional*) – Weighting for is_passive loss term. Default is 1.0.
- **patch** (*float, optional*) – Weighting for mask_patch loss term. Default is 1.0.
- **overall** (*float, optional*) – Weighting for overall mask loss term. Default is 0.0.
- **surface** (*float, optional*) – Weighting for surface line/mask loss term. Default is 1.0.
- **auxiliary** (*float, optional*) – Weighting for auxiliary loss terms "turbulence-original", "bottom-original", "mask_patches-original", and "mask_patches-ntob". Default is 1.0.
- **ignore_lines_during_passive** (*bool, optional*) – Whether targets for turbulence and bottom lines should be excluded from the loss during passive data collection. Default is True.
- **ignore_lines_during_removed** (*bool, optional*) – Whether targets for turbulence and bottom lines should be excluded from the loss during entirely removed sections. Default is True.
- **ignore_surface_during_passive** (*bool, optional*) – Whether target for the surface line should be excluded from the loss during passive data collection. Default is False.
- **ignore_surface_during_removed** (*bool, optional*) – Whether target for the surface line should be excluded from the loss during entirely removed sections. Default is True.

forward(*input*, *target*)

Construct loss term.

Parameters

- **input** (*dict*) – Output from echofilter.wrapper.Echofilter layer.
- **target** (*dict*) – A transect, as provided by *echofilter.data.dataset.TransectDataset*.

reduction: *str*

echofilter.optim package

Optimization, criterions and metrics.

Submodules

echofilter.optim.criterions module

Evaluation criterions.

echofilter.optim.criterions.**mask_accuracy**(*input*, *target*, *threshold=0.5*, *ndim=None*, *reduction='mean'*)

Measure accuracy of input compared to binary targets.

Parameters

- **input** (*torch.Tensor*) – Input tensor.
- **target** (*torch.Tensor*) – Target tensor, the same shape as `input`.
- **threshold** (*float, optional*) – Threshold which entries in `input` and `target` must exceed to be binarised as the positive class. Default is `0.5`.
- **ndim** (*int or None*) – Number of dimensions to keep. If `None`, only the first (batch) dimension is kept and the rest are flattened. Default is `None`.
- **reduction** (`"none"` or `"mean"` or `"sum"`, optional) – Specifies the reduction to apply to the output: `"none"` | `"mean"` | `"sum"`. `"none"`: no reduction will be applied, `"mean"`: the sum of the output will be divided by the number of elements in the output, `"sum"`: the output will be summed. Default: `"mean"`.

Returns The fraction of `input` which has the same class as `target` after thresholding.

Return type torch.Tensor

echofilter.optim.criterions.**mask_accuracy_with_logits**(*input*, **args*, ***kwargs*)

Measure accuracy with logit inputs.

Pass through a sigmoid, binarize, then measure accuracy of predictions compared to ground truth target.

See also:

mask_accuracy

echofilter.optim.criterions.**mask_active_fraction**(*input*, *threshold=0.5*, *ndim=None*, *reduction='mean'*)

Measure the fraction of input which exceeds a threshold.

Parameters

- **input** (*torch.Tensor*) – Input tensor.
- **threshold** (*float, optional*) – Threshold which entries in `input` must exceed. Default is `0.5`.
- **ndim** (*int or None*) – Number of dimensions to keep. If `None`, only the first (batch) dimension is kept and the rest are flattened. Default is `None`.
- **reduction** (`"none"` or `"mean"` or `"sum"`, optional) – Specifies the reduction to apply to the output: `"none"` | `"mean"` | `"sum"`. `"none"`: no reduction will be applied, `"mean"`: the sum of the output will be divided by the number of elements in the output, `"sum"`: the output will be summed. Default: `"mean"`.

Returns The fraction of `input` which exceeds `threshold`, with shaped corresponding to `reduction`.

Return type torch.Tensor

echofilter.optim.criterions.**mask_active_fraction_with_logits**(*input*, **args*, ***kwargs*)

Convert logits to probabilities, and measure what fraction exceed threshold.

See also:

mask_active_fraction

echofilter.optim.criterions.**mask_f1_score**(*input*, *target*, *reduction='mean'*, ***kwargs*)

Measure F1-score of probability input.

Binarize, then measure the F1-score of the input vs ground truth target,

Parameters

- **input** (*torch.Tensor*) – Input tensor.
- **target** (*torch.Tensor*) – Target tensor, the same shape as `input`.
- **threshold** (*float, optional*) – Threshold which entries in `input` and `target` must exceed to be binarised as the positive class. Default is `0.5`.
- **ndim** (*int or None*) – Number of dimensions to keep. If `None`, only the first (batch) dimension is kept and the rest are flattened. Default is `None`.
- **reduction** (`"none"` or `"mean"` or `"sum"`, optional) – Specifies the reduction to apply to the output: `"none"` | `"mean"` | `"sum"`. `"none"`: no reduction will be applied, `"mean"`: the sum of the output will be divided by the number of elements in the output, `"sum"`: the output will be summed. Default: `"mean"`.

Returns The F1-score of `input` as compared to `target` after thresholding. The F1-score is the harmonic mean of precision and recall.

Return type torch.Tensor

See also:

mask_precision, *mask_recall*

echofilter.optim.criterions.**mask_f1_score_with_logits**(*input*, **args*, ***kwargs*)

Measure F1-score of logit input.

Convert logits to probabilities with sigmoid, apply a threshold, then measure the F1-score of the tensor as compared to ground truth.

See also:

mask_f1_score

echofilter.optim.criterions.**mask_jaccard_index**(*input*, *target*, *threshold=0.5*, *ndim=None*, *reduction='mean'*)

Measure Jaccard Index from probabilities.

Measure the Jaccard Index (intersection over union) of the input as compared to a ground truth target, after binarising with a threshold.

Parameters

- **input** (*torch.Tensor*) – Input tensor.
- **target** (*torch.Tensor*) – Target tensor, the same shape as `input`.

- **threshold** (*float, optional*) – Threshold which entries in `input` and `target` must exceed to be binarised as the positive class. Default is `0.5`.
- **ndim** (*int or None*) – Number of dimensions to keep. If `None`, only the first (batch) dimension is kept and the rest are flattened. Default is `None`.
- **reduction** (`"none"` or `"mean"` or `"sum"`, optional) – Specifies the reduction to apply to the output: `"none"` | `"mean"` | `"sum"`. `"none"`: no reduction will be applied, `"mean"`: the sum of the output will be divided by the number of elements in the output, `"sum"`: the output will be summed. Default: `"mean"`.

Returns The Jaccard Index of `input` as compared to `target`. The Jaccard Index is the number of elements where both `input` and `target` exceed `threshold`, divided by the number of elements where at least one of `input` and `target` exceeds `threshold`.

Return type torch.Tensor

echofilter.optim.criterions.**mask_jaccard_index_with_logits**(*input*, **args*, ***kwargs*)

Measure Jaccard Index from logits.

Convert logits to probabilities with sigmoid, apply a threshold, then measure the Jaccard Index (intersection over union) of the tensor as compared to ground truth.

See also:

mask_jaccard_index

echofilter.optim.criterions.**mask_precision**(*input*, *target*, *threshold=0.5*, *ndim=None*, *reduction='mean'*)

Measure precision of probability input.

Binarize with a threshold, then measure precision compared to a ground truth target.

Parameters

- **input** (*torch.Tensor*) – Input tensor.
- **target** (*torch.Tensor*) – Target tensor, the same shape as `input`.
- **threshold** (*float, optional*) – Threshold which entries in `input` and `target` must exceed to be binarised as the positive class. Default is `0.5`.
- **ndim** (*int or None*) – Number of dimensions to keep. If `None`, only the first (batch) dimension is kept and the rest are flattened. Default is `None`.
- **reduction** (`"none"` or `"mean"` or `"sum"`, optional) – Specifies the reduction to apply to the output: `"none"` | `"mean"` | `"sum"`. `"none"`: no reduction will be applied, `"mean"`: the sum of the output will be divided by the number of elements in the output, `"sum"`: the output will be summed. Default: `"mean"`.

Returns The precision of `input` as compared to `target` after thresholding. The fraction of predicted positive cases, *input > 0.5*, which are true positive cases (*input > 0.5 and `target > 0.5*). If there are no predicted positives, the output is `0` if there are any positives to predict and `1` if there are none.

Return type torch.Tensor

echofilter.optim.criterions.**mask_precision_with_logits**(*input*, **args*, ***kwargs*)

Measure precision of logit input.

Pass through sigmoid, threshold, then measure precision.

See also:

mask_precision

echofilter.optim.criterions.**mask_recall**(*input*, *target*, *threshold=0.5*, *ndim=None*, *reduction='mean'*)

Measure recall of probability input.

Binarize with a threshold, then measure the recall compared to a ground truth target.

Parameters

- **input** (*torch.Tensor*) – Input tensor.
- **target** (*torch.Tensor*) – Target tensor, the same shape as `input`.
- **threshold** (*float, optional*) – Threshold which entries in `input` and `target` must exceed to be binarised as the positive class. Default is `0.5`.
- **ndim** (*int or None*) – Number of dimensions to keep. If `None`, only the first (batch) dimension is kept and the rest are flattened. Default is `None`.
- **reduction** (`"none"` or `"mean"` or `"sum"`, optional) – Specifies the reduction to apply to the output: `"none"` | `"mean"` | `"sum"`. `"none"`: no reduction will be applied, `"mean"`: the sum of the output will be divided by the number of elements in the output, `"sum"`: the output will be summed. Default: `"mean"`.

Returns The recall of `input` as compared to `target` after thresholding. The fraction of true positive cases, *target > 0.5*, which are true positive cases (*input > 0.5 and `target > 0.5*). If there are no true positives, the output is `1`.

Return type torch.Tensor

echofilter.optim.criterions.**mask_recall_with_logits**(*input*, **args*, ***kwargs*)

Measure recall of logit input.

Pass through sigmoid, binarize, then measure recall of ground truth target.

See also:

mask_recall

echofilter.optim.meters module

Meters for tracking measurements during training.

class echofilter.optim.meters.**AverageMeter**(*name*, *fmt=':f'*)

Bases: `object`

Compute and store the average and current value.

reset()

update(*val*, *n=None*)

class echofilter.optim.meters.**ProgressMeter**(*num_batches*, *meters*, *prefix=''*)

Bases: `object`

display(*batch*)

echofilter.optim.schedulers module

class echofilter.optim.schedulers.**MesaOneCycleLR**(*optimizer*, *max_lr*, *total_steps=None*, *pct_start=0.25*, *pct_end=0.75*, ***kwargs*)

Bases: *echofilter.optim.torch_backports.OneCycleLR*

A 1-cycle learning rate schedule with a flat region at maximum learning rate.

Sets the learning rate of each parameter group according to the 1cycle learning rate policy. The 1cycle policy anneals the learning rate from an initial learning rate to some maximum learning rate and then from that maximum learning rate to some minimum learning rate much lower than the initial learning rate. This policy was initially described in the paper Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates.

The 1cycle learning rate policy changes the learning rate after every batch. `step` should be called after a batch has been used for training.

This scheduler is not chainable.

Note also that the total number of steps in the cycle can be determined in one of two ways (listed in order of precedence):

1. A value for total_steps is explicitly provided.
2. A number of epochs (epochs) and a number of steps per epoch (steps_per_epoch) are provided. In this case, the number of total steps is inferred by total_steps = epochs * steps_per_epoch

You must either provide a value for total_steps or provide a value for both epochs and steps_per_epoch.

Parameters

- **optimizer** (*torch.optim.optimizer.Optimizer*) – Wrapped optimizer.
- **max_lr** (*float or list*) – Upper learning rate boundaries in the cycle for each parameter group.
- **total_steps** (*int*) – The total number of steps in the cycle. Note that if a value is provided here, then it must be inferred by providing a value for epochs and steps_per_epoch. Default: None
- **epochs** (*int*) – The number of epochs to train for. This is used along with steps_per_epoch in order to infer the total number of steps in the cycle if a value for total_steps is not provided. Default: None
- **steps_per_epoch** (*int*) – The number of steps per epoch to train for. This is used along with epochs in order to infer the total number of steps in the cycle if a value for total_steps is not provided. Default: None
- **pct_start** (*float*) – The percentage of the cycle (in number of steps) spent increasing the learning rate. Default: 0.25
- **pct_end** (*float*) – The percentage of the cycle (in number of steps) spent before decreasing the learning rate. Default: 0.75
- **anneal_strategy** (*{"cos", "linear"}*) – Specifies the annealing strategy: “cos” for cosine annealing, “linear” for linear annealing. Default: “cos”.
- **cycle_momentum** (*bool, default=True*) – If `True`, momentum is cycled inversely to learning rate between “base_momentum” and “max_momentum”. Default: True
- **base_momentum** (*float or list*) – Lower momentum boundaries in the cycle for each parameter group. Note that momentum is cycled inversely to learning rate; at the peak of a cycle, momentum is “base_momentum” and learning rate is “max_lr”. Default: 0.85

- **max_momentum** (*float or list*) – Upper momentum boundaries in the cycle for each parameter group. Functionally, it defines the cycle amplitude (max_momentum - base_momentum). Note that momentum is cycled inversely to learning rate; at the start of a cycle, momentum is "max_momentum" and learning rate is "base_lr" Default: 0.95
- **div_factor** (*float*) – Determines the initial learning rate via initial_lr = max_lr/div_factor Default: 25
- **final_div_factor** (*float*) – Determines the minimum learning rate via min_lr = initial_lr/final_div_factor Default: 1e4
- **last_epoch** (*int*) – The index of the last batch. This parameter is used when resuming a training job. Since `step()` should be invoked after each batch instead of after each epoch, this number represents the total number of *batches* computed, not the total number of epochs computed. When last_epoch=-1, the schedule is started from the beginning. Default: -1

Example

```
>>> data_loader = torch.utils.data.DataLoader(...)
>>> optimizer = torch.optim.SGD(model.parameters(), lr=0.1, momentum=0.9)
>>> scheduler = MesaOneCycleLR(optimizer, max_lr=0.01, steps_per_epoch=len(data_
↪loader), epochs=10)
>>> for epoch in range(10):
>>>     for batch in data_loader:
>>>         train_batch(...)
>>>         scheduler.step()
```

get_lr()

echofilter.optim.torch_backports module

Pytorch classes backported from later versions.

This contains functions copied from newer versions of pytorch than v1.2.0, which is the latest version currently available from IBM compiled for ppc64 architectures.

From PyTorch:

Copyright (c) 2016- Facebook, Inc (Adam Paszke) Copyright (c) 2014- Facebook, Inc (Soumith Chintala) Copyright (c) 2011-2014 Idiap Research Institute (Ronan Collobert) Copyright (c) 2012-2014 Deepmind Technologies (Koray Kavukcuoglu) Copyright (c) 2011-2012 NEC Laboratories America (Koray Kavukcuoglu) Copyright (c) 2011-2013 NYU (Clement Farabet) Copyright (c) 2006-2010 NEC Laboratories America (Ronan Collobert, Leon Bottou, Iain Melvin, Jason Weston) Copyright (c) 2006 Idiap Research Institute (Samy Bengio) Copyright (c) 2001-2004 Idiap Research Institute (Ronan Collobert, Samy Bengio, Johnny Mariethoz)

From Caffe2:

Copyright (c) 2016-present, Facebook Inc. All rights reserved.

All contributions by Facebook: Copyright (c) 2016 Facebook Inc.

All contributions by Google: Copyright (c) 2015 Google Inc. All rights reserved.

All contributions by Yangqing Jia: Copyright (c) 2015 Yangqing Jia All rights reserved.

All contributions from Caffe: Copyright(c) 2013, 2014, 2015, the respective contributors All rights reserved.

All other contributions: Copyright(c) 2015, 2016 the respective contributors All rights reserved.

Caffe2 uses a copyright model similar to Caffe: each contributor holds copyright over their contributions to Caffe2. The project versioning records all such contribution and copyright details. If a contributor wants to further mark their specific copyright on a particular contribution, they should indicate their copyright solely in the commit message of the change when it is committed.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the names of Facebook, Deepmind Technologies, NYU, NEC Laboratories America and IDIAP Research Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

class echofilter.optim.torch_backports.**OneCycleLR**(*optimizer*, *max_lr*, *total_steps=None*, *epochs=None*, *steps_per_epoch=None*, *pct_start=0.3*, *anneal_strategy='cos'*, *cycle_momentum=True*, *base_momentum=0.85*, *max_momentum=0.95*, *div_factor=25.0*, *final_div_factor=10000.0*, *last_epoch=- 1*)

Bases: echofilter.optim.torch_backports._LRScheduler

Backported from pytorch 1.4.0.

Sets the learning rate of each parameter group according to the 1cycle learning rate policy. The 1cycle policy anneals the learning rate from an initial learning rate to some maximum learning rate and then from that maximum learning rate to some minimum learning rate much lower than the initial learning rate. This policy was initially described in the paper Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates.

The 1cycle learning rate policy changes the learning rate after every batch. `step` should be called after a batch has been used for training.

This scheduler is not chainable.

Note also that the total number of steps in the cycle can be determined in one of two ways (listed in order of precedence):

1. A value for total_steps is explicitly provided.
2. A number of epochs (epochs) and a number of steps per epoch (steps_per_epoch) are provided. In this case, the number of total steps is inferred by total_steps = epochs * steps_per_epoch

You must either provide a value for total_steps or provide a value for both epochs and steps_per_epoch.

Parameters

- **optimizer** (torch.optim.optimizer.Optimizer) – Wrapped optimizer.
- **max_lr** (*float or list*) – Upper learning rate boundaries in the cycle for each parameter group.
- **total_steps** (*int*) – The total number of steps in the cycle. Note that if a value is provided here, then it must be inferred by providing a value for epochs and steps_per_epoch. Default: None
- **epochs** (*int*) – The number of epochs to train for. This is used along with steps_per_epoch in order to infer the total number of steps in the cycle if a value for total_steps is not provided. Default: None
- **steps_per_epoch** (*int*) – The number of steps per epoch to train for. This is used along with epochs in order to infer the total number of steps in the cycle if a value for total_steps is not provided. Default: None
- **pct_start** (*float*) – The percentage of the cycle (in number of steps) spent increasing the learning rate. Default: 0.3
- **anneal_strategy** (*{'cos', 'linear'}*) – Specifies the annealing strategy: "cos" for cosine annealing, "linear" for linear annealing. Default: 'cos'
- **cycle_momentum** (*bool*) – If `True`, momentum is cycled inversely to learning rate between 'base_momentum' and 'max_momentum'. Default: True
- **base_momentum** (*float or list*) – Lower momentum boundaries in the cycle for each parameter group. Note that momentum is cycled inversely to learning rate; at the peak of a cycle, momentum is 'base_momentum' and learning rate is 'max_lr'. Default: 0.85
- **max_momentum** (*float or list*) – Upper momentum boundaries in the cycle for each parameter group. Functionally, it defines the cycle amplitude (max_momentum - base_momentum). Note that momentum is cycled inversely to learning rate; at the start of a cycle, momentum is 'max_momentum' and learning rate is 'base_lr' Default: 0.95
- **div_factor** (*float*) – Determines the initial learning rate via initial_lr = max_lr/div_factor Default: 25
- **final_div_factor** (*float*) – Determines the minimum learning rate via min_lr = initial_lr/final_div_factor Default: 1e4
- **last_epoch** (*int*) – The index of the last batch. This parameter is used when resuming a training job. Since `step()` should be invoked after each batch instead of after each epoch, this number represents the total number of *batches* computed, not the total number of epochs computed. When last_epoch=-1, the schedule is started from the beginning. Default: -1

Example

```
>>> data_loader = torch.utils.data.DataLoader(...)
>>> optimizer = torch.optim.SGD(model.parameters(), lr=0.1, momentum=0.9)
>>> scheduler = torch.optim.lr_scheduler.OneCycleLR(
>>>     optimizer, max_lr=0.01, steps_per_epoch=len(data_loader), epochs=10
>>> )
>>> for epoch in range(10):
>>>     for batch in data_loader:
>>>         train_batch(...)
>>>         scheduler.step()
```

get_lr()

echofilter.optim.utils module

Utility functions for interacting with optimizers.

echofilter.optim.utils.**get_current_lr**(*optimizer*)

Get the learning rate of an optimizer.

Parameters **optimizer** (*torch.optim.Optimizer*) – An optimizer, with a learning rate common to all parameter groups.

Returns The learning rate of the first parameter group.

Return type float

echofilter.optim.utils.**get_current_momentum**(*optimizer*)

Get the momentum of an optimizer.

Parameters **optimizer** (*torch.optim.Optimizer*) – An optimizer which implements momentum or betas (where momentum is the first beta, c.f. `torch.optim.Adam`) with a momentum common to all parameter groups.

Returns The momentum of the first parameter group.

Return type float

echofilter.raw package

Echoview output file loading and generation, post-processing and shard generation.

Submodules

echofilter.raw.loader module

Input/Output handling for raw Echoview files.

echofilter.raw.loader.**count_lines**(*filename*)

Count the number of lines in a file.

Parameters **filename** (*str*) – Path to file.

Returns Number of lines in file.

Return type int

echofilter.raw.loader.**evdtstr2timestamp**(*datestr*, *timestr=None*)

Convert an Echoview-compatible datetime string into a Unix epoch timestamp.

Parameters

- **datestr** (*str*) – Datetime string in the Echoview-compatible format `"CCYYMMDD HHmmSSssss"`, or (if timestr is also provided) just the date part, `"CCYYMMDD"`.
- **timestr** (*str, optional*) – Time string in the Echoview-compatible format "HHmmSSssss".

Returns **timestamp** – Number of seconds since Unix epoch.

Return type float

echofilter.raw.loader.**evl_loader**(*fname*, *special_to_nan=True*, *return_status=False*)

EVL file loader.

Parameters

- **fname** (*str*) – Path to .evl file.
- **special_to_nan** (*bool, optional*) – Whether to replace the special value, `-10000.99`, which indicates no depth value, with NaN. https://support.echoview.com/WebHelp/Reference/File_formats/Export_file_formats/Special_Export_Values.htm

Returns

- *numpy.ndarray of floats* – Timestamps, in seconds.
- *numpy.ndarary of floats* – Depth, in metres.
- *numpy.ndarary of ints, optional* – Status codes.

echofilter.raw.loader.**evl_reader**(*fname*)

EVL file reader.

Parameters **fname** (*str*) – Path to .evl file.

Returns A generator which yields the timestamp (in seconds), depth (in metres), and status (int) for each entry. Note that the timestamp is not corrected for timezone (so make sure your timezones are internally consistent).

Return type generator

echofilter.raw.loader.**evl_writer**(*fname*, *timestamps*, *depths*, *status=1*, *line_ending='\r\n'*, *pad=False*)

EVL file writer.

Parameters

- **fname** (*str*) – Destination of output file.
- **timestamps** (*array_like*) – Timestamps for each node in the line.
- **depths** (*array_like*) – Depths (in meters) for each node in the line.
- **status** (*0, 1, 2, or 3; optional*) – Status for the line.
  - `0` : none
  - `1` : unverified
  - `2` : bad
  - `3` : good

  Default is `1` (unverified). For more details on line status, see https://support.echoview.com/WebHelp/Using_Echoview/Echogram/Lines/About_Line_Status.htm
- **pad** (*bool, optional*) – Whether to pad the line with an extra datapoint half a pixel before the first and after the last given timestamp. Default is `False`.
- **line_ending** (*str, optional*) – Line ending. Default is `"\r\n"` the standard line ending on Windows/DOS, as per the specification for the file format. https://support.echoview.com/WebHelp/Using_Echoview/Exporting/Exporting_data/Exporting_line_data.htm Set to `"\n"` to get Unix-style line endings instead.

Notes

For more details on the format specification, see https://support.echoview.com/WebHelp/Using_Echoview/Exporting/Exporting_data/Exporting_line_data.htm#Line_definition_file_format

echofilter.raw.loader.**evr_reader**(*fname*, *parse_echofilter_regions=True*)

Echoview region file (EVR) reader.

Parameters

- **fname** (*str*) – Path to .evr file.
- **parse_echofilter_regions** (*bool, default=True*) – Whether to separate out echofilter generated regions (passive, removed vbands, and removed patches) from other regions.

Returns

- **regions_passive** (*list of tuples, optional*) – Start and end timestamps for passive regions.
- **regions_removed** (*list of tuples, optional*) – Start and end timestamps for removed vertical bands.
- **regions_patch** (*list of lists, optional*) – Start and end timestamps for bad data patches.
- **regions_other** (*list of dicts*) – Dictionary mapping creation type to points defining each region.

echofilter.raw.loader.**evr_writer**(*fname*, *rectangles=None*, *contours=None*, *common_notes=''*, *default_region_type=0*, *line_ending='\r\n'*)

EVR file writer.

Writes regions to an Echoview region file.

Parameters

- **fname** (*str*) – Destination of output file.
- **rectangles** (*list of dictionaries, optional*) – Rectangle region definitions. Default is an empty list. Each rectangle region must implement fields `"depths"` and `"timestamps"`, which indicate the extent of the rectangle. Optionally, `"creation_type"`, `"region_name"`, `"region_type"`, and `"notes"` may be set. If these are not given, the default creation_type is 4 and region_type is set by `default_region_type`.
- **contours** (*list of dictionaries*) – Contour region definitions. Default is an empty list. Each contour region must implement a `"points"` field containing a `numpy.ndarray` shaped *(n, 2)* defining the co-ordinates of nodes along the (open) contour in units of timestamp and depth. Optionally, `"creation_type"`, `"region_name"`, `"region_type"`, and `"notes"` may be set. If these are not given, the default creation_type is 2 and region_type is set by `default_region_type`.
- **common_notes** (*str, optional*) – Notes to include for every region. Default is `""`, an empty string.
- **default_region_type** (*int, optional*) – The region type to use for rectangles and contours which do not define a `"region_type"` field. Possible region types are
  - `0` : bad (no data)
  - `1` : analysis
  - `2` : marker
  - `3` : fishtracks

- 4 : bad (empty water)

  Default is 0.

- **line_ending** (*str, optional*) – Line ending. Default is "\r\n" the standard line ending on Windows/DOS, as per the specification for the file format. https://support.echoview.com/WebHelp/Using_Echoview/Exporting/Exporting_data/Exporting_line_data.htm Set to "\n" to get Unix-style line endings instead.

Notes

For more details on the format specification, see: https://support.echoview.com/WebHelp/Reference/File_formats/Export_file_formats/2D_Region_definition_file_format.htm

echofilter.raw.loader.**get_partition_data**(*partition*, *dataset='mobile'*, *partitioning_version='firstpass'*, *root_data_dir='/data/dsforce/surveyExports'*)

Load partition metadata.

Parameters

- **transect_pth** (*str*) – Relative path to transect, excluding "_Sv_raw.csv".
- **dataset** (*str, optional*) – Name of dataset. Default is "mobile".
- **partitioning_version** (*str, optional*) – Name of partitioning method.
- **root_data_dir** (*str*) – Path to root directory where data is located.

Returns Metadata for all transects in the partition. Each row is a single sample.

Return type pandas.DataFrame

echofilter.raw.loader.**get_partition_list**(*partition*, *dataset='mobile'*, *full_path=False*, *partitioning_version='firstpass'*, *root_data_dir='/data/dsforce/surveyExports'*, *sharded=False*)

Get a list of transects in a single partition.

Parameters

- **transect_pth** (*str*) – Relative path to transect, excluding "_Sv_raw.csv".
- **dataset** (*str, optional*) – Name of dataset. Default is "mobile".
- **full_path** (*bool, optional*) – Whether to return the full path to the sample. If False, only the relative path (from the dataset directory) is returned. Default is False.
- **partitioning_version** (*str, optional*) – Name of partitioning method.
- **root_data_dir** (*str, optional*) – Path to root directory where data is located.
- **sharded** (*bool, optional*) – Whether to return path to sharded version of data. Default is False.

Returns Path for each sample in the partition.

Return type list

echofilter.raw.loader.**list_from_file**(*fname*)

Get a list from a file.

Parameters **fname** (*str*) – Path to file.

Returns Contents of the file, one line per entry in the list. Trailing whitespace is removed from each end of each line.

Return type list

echofilter.raw.loader.**load_transect_data**(*transect_pth*, *dataset='mobile'*, *root_data_dir='/data/dsforce/surveyExports'*)

Load all data for one transect.

Parameters

- **transect_pth** (*str*) – Relative path to transect, excluding "_Sv_raw.csv".
- **dataset** (*str, optional*) – Name of dataset. Default is "mobile".
- **root_data_dir** (*str*) – Path to root directory where data is located.

Returns

- **timestamps** (*numpy.ndarray*) – Timestamps (in seconds since Unix epoch), with each entry corresponding to each row in the signals data.
- **depths** (*numpy.ndarray*) – Depths from the surface (in metres), with each entry corresponding to each column in the signals data.
- **signals** (*numpy.ndarray*) – Echogram Sv data, shaped (num_timestamps, num_depths).
- **turbulence** (*numpy.ndarray*) – Depth of turbulence line, shaped (num_timestamps,).
- **bottom** (*numpy.ndarray*) – Depth of bottom line, shaped (num_timestamps,).

echofilter.raw.loader.**regions2mask**(*timestamps*, *depths*, *regions_passive=None*, *regions_removed=None*, *regions_patch=None*, *regions_other=None*)

Convert regions to mask.

Takes the output from :func:evr_reader` and returns a set of masks.

Parameters

- **timestamps** (*array_like*) – Timestamps for each node in the line.
- **depths** (*array_like*) – Depths (in meters) for each node in the line.
- **regions_passive** (*list of tuples, optional*) – Start and end timestamps for passive regions.
- **regions_removed** (*list of tuples, optional*) – Start and end timestamps for removed vertical bands.
- **regions_patch** (*list of lists, optional*) – Start and end timestamps for bad data patches.
- **regions_other** (*list of dicts*) – Dictionary mapping creation type to points defining each region.

Returns

transect –

A dictionary with keys:

- **"is_passive"** [numpy.ndarray] Logical array showing whether a timepoint is of passive data. Shaped (num_timestamps,). All passive recording data should be excluded by the mask.
- **"is_removed"** [numpy.ndarray] Logical array showing whether a timepoint is entirely removed by the mask. Shaped (num_timestamps,).

- **"mask_patches"** [numpy.ndarray] Logical array indicating which datapoints are inside a patch from regions_patch (`True`) and should be excluded by the mask. Shaped `(num_timestamps, num_depths)`.
- **"mask"** [numpy.ndarray] Logical array indicating which datapoints should be kept (`True`) and which are marked as removed (`False`) by one of the other three outputs. Shaped `(num_timestamps, num_depths)`.

Return type dict

`echofilter.raw.loader.`**`remove_trailing_slash`**(*s*)

Remove trailing forward slashes from a string.

Parameters **s** (*str*) – String representing a path, possibly with trailing slashes.

Returns Same as `s`, but without trailing forward slashes.

Return type str

`echofilter.raw.loader.`**`timestamp2evdtstr`**(*timestamp*)

Convert a timestamp into an Echoview-compatible datetime string.

The output is in the format "CCYYMMDD HHmmSSssss", where:

CC: century
YY: year
MM: month
DD: day
HH: hour
mm: minute
SS: second
ssss: 0.1 milliseconds

Parameters **timestamp** (*float*) – Number of seconds since Unix epoch.

Returns **datetimestring** – Datetime string in the Echoview-compatible format "CCYYMMDD HHmmSSssss".

Return type str

`echofilter.raw.loader.`**`transect_loader`**(*fname*, *skip_lines=0*, *warn_row_overflow=None*, *row_len_selector='mode'*)

Load an entire survey transect CSV.

Parameters

- **fname** (*str*) – Path to survey CSV file.
- **skip_lines** (*int, optional*) – Number of initial entries to skip. Default is 0.
- **warn_row_overflow** (*bool or int, optional*) – Whether to print a warning message if the number of elements in a row exceeds the expected number. If this is an int, this is the number of times to display the warnings before they are supressed. If this is `True`, the number of outputs is unlimited. If `None`, the maximum number of underflow and overflow warnings differ: if `row_len_selector` is `"init"` or `"min"`, underflow always produces a message and the overflow messages stop at 2; otherwise the values are reversed. Default is `None`.

- **row_len_selector** (*{"init", "min", "max", "median", "mode"}, optional*) – The method used to determine which row length (number of depth samples) to use. Default is "mode", the most common row length across all the measurement timepoints.

Returns

- *numpy.ndarray* – Timestamps for each row, in seconds. Note: not corrected for timezone (so make sure your timezones are internally consistent).
- *numpy.ndarray* – Depth of each column, in metres.
- *numpy.ndarray* – Survey signal (Sv, for instance). Units match that of the file.

echofilter.raw.loader.**transect_reader**(*fname*)

Create a generator which iterates through a survey csv file.

Parameters **fname** (*str*) – Path to survey CSV file.

Returns Yields a tupule of *(metadata, data)*, where metadata is a dict, and data is a numpy.ndarray. Each yield corresponds to a single row in the data. Every row (except for the header) is yielded.

Return type generator

echofilter.raw.loader.**write_transect_regions**(*fname*, *transect*, *depth_range=None*, *passive_key='is_passive'*, *removed_key='is_removed'*, *patches_key='mask_patches'*, *collate_passive_length=0*, *collate_removed_length=0*, *minimum_passive_length=0*, *minimum_removed_length=0*, *minimum_patch_area=0*, *name_suffix=''*, *common_notes=''*, *line_ending='\r\n'*, *verbose=0*, *verbose_indent=0*)

Convert a transect dictionary to a set of regions and write as an EVR file.

Parameters

- **fname** (*str*) – Destination of output file.
- **transect** (*dict*) – Transect dictionary.
- **depth_range** (*array_like or None, optional*) – The minimum and maximum depth extents (in any order) of the passive and removed block regions. If this is None (default), the minimum and maximum of transect["depths"] is used.
- **passive_key** (*str, optional*) – Field name to use for passive data identification. Default is "is_passive".
- **removed_key** (*str, optional*) – Field name to use for removed blocks. Default is "is_removed".
- **patches_key** (*str, optional*) – Field name to use for the mask of patch regions. Default is "mask_patches".
- **collate_passive_length** (*int, optional*) – Maximum distance (in indices) over which passive regions should be merged together, closing small gaps between them. Default is 0.
- **collate_removed_length** (*int, optional*) – Maximum distance (in indices) over which removed blocks should be merged together, closing small gaps between them. Default is 0.
- **minimum_passive_length** (*int, optional*) – Minimum length (in indices) a passive region must have to be included in the output. Set to -1 to omit all passive regions from the output. Default is 0.

- **minimum_removed_length** (*int, optional*) – Minimum length (in indices) a removed block must have to be included in the output. Set to -1 to omit all removed regions from the output. Default is `0`.
- **minimum_patch_area** (*float, optional*) – Minimum amount of area (in input pixel space) that a patch must occupy in order to be included in the output. Set to `0` to include all patches, no matter their area. Set to `-1` to omit all patches. Default is `0`.
- **name_suffix** (*str, optional*) – Suffix to append to variable names. Default is "", an empty string.
- **common_notes** (*str, optional*) – Notes to include for every region. Default is "", an empty string.
- **line_ending** (*str, optional*) – Line ending. Default is "\r\n" the standard line ending on Windows/DOS, as per the specification for the file format, https://support.echoview.com/WebHelp/Using_Echoview/Exporting/Exporting_data/Exporting_line_data.htm Set to "\n" to get Unix-style line endings instead.
- **verbose** (*int, optional*) – Verbosity level. Default is `0`.
- **verbose_indent** (*int, optional*) – Level of indentation (number of preceding spaces) before verbosity messages. Default is `0`.

echofilter.raw.manipulate module

Manipulating lines and masks contained in Echoview files.

echofilter.raw.manipulate.**find_nonzero_region_boundaries**(*v*)

Find the start and end indices for nonzero regions of a vector.

Parameters **v** (*array_like*) – A vector.

Returns

- **starts** (*numpy.ndarray*) – Indices for start of regions of nonzero elements in vector `v`
- **ends** (*numpy.ndarray*) – Indices for end of regions of nonzero elements in vector `v` (exclusive).

Notes

For `i` in `range(len(starts))`, the set of values `v[starts[i]:ends[i]]` are nonzero. Values in the range `v[ends[i]:starts[i+1]]` are zero.

echofilter.raw.manipulate.**find_passive_data**(*signals*, *n_depth_use=38*, *threshold=25.0*, *deviation=None*)

Find segments of Sv recording which correspond to passive recording.

Parameters

- **signals** (*array_like*) – Two-dimensional array of Sv values, shaped *[timestamps, depths]*.
- **n_depth_use** (*int, optional*) – How many Sv depths to use, starting with the first depths (closest to the sounder device). If `None` all depths are used. Default is `38`.
- **threshold** (*float, optional*) – Threshold for start/end of passive regions. Default is `25`.

- **deviation** (*float, optional*) – Threshold for start/end of passive regions is `deviation` times the interquartile-range of the difference between samples at neigbouring timestamps. Default is `None`. Only one of `threshold` and `deviation` should be set.

Returns

- **passive_start** (*numpy.ndarray*) – Indices of rows of `signals` at which passive segments start.
- **passive_end** (*numpy.ndarray*) – Indices of rows of `signals` at which passive segments end.

Notes

Works by looking at the difference between consecutive recordings and finding large deviations.

echofilter.raw.manipulate.**find_passive_data_v2**(*signals*, *n_depth_use=38*, *threshold_inner=None*, *threshold_init=None*, *deviation=None*, *sigma_depth=0*, *sigma_time=1*)

Find segments of Sv recording which correspond to passive recording.

Parameters

- **signals** (*array_like*) – Two-dimensional array of Sv values, shaped *[timestamps, depths]*.
- **n_depth_use** (*int, optional*) – How many Sv depths to use, starting with the first depths (closest to the sounder device). If `None` all depths are used. Default is `38`. The median is taken across the depths, after taking the temporal derivative.
- **threshold_inner** (*float, optional*) – Theshold to apply to the temporal derivative of the signal when detected fine-tuned start/end of passive regions. Default behaviour is to use a threshold automatically determined using `deviation` if it is set, and otherwise use a threshold of `35.0`.
- **threshold_init** (*float, optional*) – Theshold to apply during the initial scan of the start/end of passive regions, which seeds the fine-tuning search. Default behaviour is to use a threshold automatically determined using `deviation` if it is set, and otherwise use a threshold of `12.0`.
- **deviation** (*float, optional*) – Set `threshold_inner` to be `deviation` times the standard deviation of the temporal derivative of the signal. The standard deviation is robustly estimated based on the interquartile range. If this is set, `threshold_inner` must not be `None`. Default is `None`
- **sigma_depth** (*float, optional*) – Width of kernel for filtering signals across second dimension (depth). Default is `0` (no filter).
- **sigma_time** (*float, optional*) – Width of kernel for filtering signals across second dimension (time). Default is `1`. Set to `0` to not filter.

Returns

- **passive_start** (*numpy.ndarray*) – Indices of rows of `signals` at which passive segments start.
- **passive_end** (*numpy.ndarray*) – Indices of rows of `signals` at which passive segments end.

Notes

Works by looking at the difference between consecutive recordings and finding large deviations.

echofilter.raw.manipulate.**fix_surface_line**(*timestamps*, *d_surface*, *is_passive*)

Fix anomalies in the surface line.

Parameters

- **timestamps** (*array_like sized (N,)*) – Timestamps for each ping.
- **d_surface** (*array_like sized (N,)*) – Surface line depths.
- **is_passive** (*array_like sized (N,)*) – Indicator for passive data. Values for the surface line during passive data collection will not be used.

Returns

- **fixed_surface** (*numpy.ndarray*) – Surface line depths, with anomalies replaced with median filtered values and passive data replaced with linear interpolation. Has the same size and dtype as `d_surface`.
- **is_replaced** (*boolean numpy.ndarray sized (N,)*) – Indicates which datapoints were replaced. Note that passive data is always replaced and is marked as such.

echofilter.raw.manipulate.**fixup_lines**(*timestamps*, *depths*, *mask*, *t_turbulence=None*, *d_turbulence=None*, *t_bottom=None*, *d_bottom=None*)

Extend existing turbulence/bottom lines based on masked target Sv output.

Parameters

- **timestamps** (*array_like*) – Shaped *(num_timestamps,)*.
- **depths** (*array_like*) – Shaped *(num_depths,)*.
- **mask** (*array_like*) – Boolean array, where `True` denotes kept entries. Shaped *(num_timestamps, num_depths)*.
- **t_turbulence** (*array_like, optional*) – Sampling times for existing turbulence line.
- **d_turbulence** (*array_like, optional*) – Depth of existing turbulence line.
- **t_bottom** (*array_like, optional*) – Sampling times for existing bottom line.
- **d_bottom** (*array_like, optional*) – Depth of existing bottom line.

Returns

- **d_turbulence_new** (*numpy.ndarray*) – Depth of new turbulence line.
- **d_bottom_new** (*numpy.ndarray*) – Depth of new bottom line.

echofilter.raw.manipulate.**join_transect**(*transects*)

Join segmented transects together into a single dictionary.

Parameters **transects** (*iterable of dict*) – Transect segments, each with the same fields and compatible shapes.

Yields *dict* – Transect data.

echofilter.raw.manipulate.**load_decomposed_transect_mask**(*sample_path*)

Load a raw and masked transect and decompose the mask.

The mask is decomposed into turbulence and bottom lines, and passive and removed regions.

Parameters **sample_path** (*str*) – Path to sample, without extension. The raw data should be located at `sample_path + "_Sv_raw.csv"`.

Returns

A dictionary with keys:

- **"timestamps"** [numpy.ndarray] Timestamps (in seconds since Unix epoch), for each recording timepoint.
- **"depths"** [numpy.ndarray] Depths from the surface (in metres), with each entry corresponding to each column in the `signals` data.
- **"Sv"** [numpy.ndarray] Echogram Sv data, shaped (num_timestamps, num_depths).
- **"mask"** [numpy.ndarray] Logical array indicating which datapoints were kept (`True`) and which removed (`False`) for the masked Sv output. Shaped (num_timestamps, num_depths).
- **"turbulence"** [numpy.ndarray] For each timepoint, the depth of the shallowest datapoint which should be included for the mask. Shaped (num_timestamps,).
- **"bottom"** [numpy.ndarray] For each timepoint, the depth of the deepest datapoint which should be included for the mask. Shaped (num_timestamps,).
- **"is_passive"** [numpy.ndarray] Logical array showing whether a timepoint is of passive data. Shaped (num_timestamps,). All passive recording data should be excluded by the mask.
- **"is_removed"** [numpy.ndarray] Logical array showing whether a timepoint is entirely removed by the mask. Shaped (num_timestamps,). Does not include periods of passive recording.
- **"is_upward_facing"** [bool] Indicates whether the recording source is located at the deepest depth (i.e. the seabed), facing upwards. Otherwise, the recording source is at the shallowest depth (i.e. the surface), facing downwards.

Return type dict

`echofilter.raw.manipulate.`**`make_lines_from_mask`**(*mask*, *depths=None*, *max_gap_squash=1.0*)

Determine turbulence and bottom lines for a mask array.

Parameters

- **mask** (*array_like*) – A two-dimensional logical array, where for each row dimension 1 takes the value `False` for some unknown continuous stretch at the start and end of the column, with `True` values between these two masked-out regions.
- **depths** (*array_like, optional*) – Depth of each sample point along dim 1 of `mask`. Must be either monotonically increasing or monotonically decreasing. Default is the index of `mask`, `arange(mask.shape[1])`.
- **max_gap_squash** (*float, optional*) – Maximum gap to merge together, in metres. Default is `1.`.

Returns

- **d_turbulence** (*numpy.ndarray*) – Depth of turbulence line. This is the line of smaller depth which separates the `False` region of `mask` from the central region of `True` values. (If `depths` is monotonically increasing, this is for the start of the columns of `mask`, otherwise it is at the end.)
- **d_bottom** (*numpy.ndarray*) – Depth of bottom line. As for `d_turbulence`, but for the other end of the array.

echofilter.raw.manipulate.**make_lines_from_masked_csv**(*fname*)

Load a masked csv file and convert its mask to lines.

Parameters **fname** (*str*) – Path to file containing masked Echoview output data in csv format.

Returns

- **timestamps** (*numpy.ndarray*) – Sample timestamps.
- **d_turbulence** (*numpy.ndarray*) – Depth of turbulence line.
- **d_bottom** (*numpy.ndarray*) – Depth of bottom line.

echofilter.raw.manipulate.**pad_transect**(*transect*, *pad=32*, *pad_mode='reflect'*, *previous_padding='diff'*)

Pad a transect in the timestamps dimension (axis 0).

Parameters

- **transect** (*dict*) – A dictionary of transect data.
- **pad** (*int, default=32*) – Amount of padding to add.
- **pad_mode** (*str, default="reflect"*) – Padding method for out-of-bounds inputs. Must be supported by `numpy.pad()`, such as `"contast"`, `"reflect"`, or `"edge"`. If the mode is `"contast"`, the array will be padded with zeros.
- **previous_padding** (*{"diff", "add", "noop"}, default="diff"*) – How to handle this padding if the transect has already been padded.

  "diff" Extend the padding up to the target `pad` value.

  "add" Add this padding irrespective of pre-existing padding.

  "noop" Don't add any new padding if previously padded.

Returns **transect** – Like input `transect`, but with all time-like dimensions extended with padding and fields `"_pad_start"` and `"_pad_end"` changed to indicate the total padding (including any pre-existing padding).

Return type dict

echofilter.raw.manipulate.**remove_anomalies_1d**(*signal*, *thr=5*, *thr2=4*, *kernel=201*, *kernel2=31*, *return_filtered=False*)

Remove anomalies from a temporal signal.

Apply a median filter to the data, and replaces datapoints which deviate from the median filtered signal by more than some threshold with the median filtered data. This process is repeated until no datapoints deviate from the filtered line by more than the threshold.

Parameters

- **signal** (*array_like*) – The signal to filter.
- **thr** (*float, optional*) – The initial threshold will be `thr` times the standard deviation of the residuals. The standard deviation is robustly estimated from the interquartile range. Default is `5`.
- **thr2** (*float, optional*) – The threshold for repeated iterations will be `thr2` times the standard deviation of the remaining residuals. The standard deviation is robustly estimated from interdecile range. Default is `4`.
- **kernel** (*int, optional*) – The kernel size for the initial median filter. Default is `201`.
- **kernel2** (*int, optional*) – The kernel size for subsequent median filters. Default is `31`.

- **return_filtered** (*bool, optional*) – If `True`, the median filtered signal is also returned. Default is `False`.

Returns

- **signal** (*numpy.ndarray like signal*) – The input signal with anomalies replaced with median values.
- **is_replaced** (*bool numpy.ndarray shaped like signal*) – Indicator for which datapoints were replaced.
- **filtered** (*numpy.ndarray like signal, optional*) – The final median filtered signal. Returned if `return_filtered=True`.

See also:

echofilter.raw.utils.medfilt1d

echofilter.raw.manipulate.**split_transect**(*timestamps=None*, *threshold=20*, *percentile=97.5*, *max_length=- 1*, *pad_length=32*, *pad_on='max'*, ***transect*)

Split a transect into segments each containing contiguous recordings.

Parameters

- **timestamps** (*array_like*) – A 1-d array containing the timestamp at which each recording was measured. The sampling is assumed to high-frequency with occassional gaps.
- **threshold** (*int, optional*) – Threshold for splitting timestamps into segments. Any timepoints further apart than `threshold` times the `percentile` percentile of the difference between timepoints will be split apart into new segments. Default is `20`.
- **percentile** (*float, optional*) – The percentile at which to sample the timestamp intervals to establish a baseline typical interval. Default is `97.5`.
- **max_length** (*int, default=-1*) – Maximum length of each segment. Set to `0` or `-1` to disable (default).
- **pad_length** (*int, default=32*) – Amount of overlap between the segments. Set to `0` to disable.
- **pad_on** (*{"max", "thr", "all", "none"}, default="max"*) – Apply overlap padding when the transect is split due to either the total length exceeding the maximum (`"max"`), the time delta exceeding the threshold (`"thr"`), or both (`"all"`).
- ****kwargs** – Arbitrary additional transect variables, which will be split into segments as appropriate in accordance with `timestamps`.

Yields *dict* – Containing segmented data, key/value pairs as per given in `**kwargs` in addition to `timestamps`.

echofilter.raw.manipulate.**write_lines_for_masked_csv**(*fname_mask*, *fname_turbulence=None*, *fname_bottom=None*)

Write turbulence and bottom lines based on masked csv file.

Parameters

- **fname_mask** (*str*) – Path to input file containing masked Echoview output data in csv format.
- **fname_turbulence** (*str, optional*) – Destination of generated turbulence line, written in evl format. If `None` (default), the output name is `<fname_base>_mask-turbulence.evl`, where `<fname_base>` is `fname_mask` without extension and without any occurence of the substrings `_Sv_raw` or `_Sv` in the base file name.

- **fname_bottom** (*str*) – Destination of generated bottom line, written in evl format. If `None` (default), the output name is `<fname_base>_mask-bottom.evl`.

echofilter.raw.metadata module

Dataset metadata, relevant for loading correct data.

`echofilter.raw.metadata.`**recall_passive_edges**(*sample_path*, *timestamps*)

Define passive data edges for samples within known datasets.

Parameters

- **sample_path** (*str*) – Path to sample.
- **timestamps** (*array_like vector*) – Vector of timestamps in sample.

Returns

- **passive_starts** (*numpy.ndarray or None*) – Indices indicating the onset of passive data collection periods, or `None` if passive metadata is unavailable for this sample.
- **passive_ends** (*numpy.ndarray or None*) – Indices indicating the offset of passive data collection periods, or `None` if passive metadata is unavailable for this sample.
- **finder_version** (*absent or str*) – If `passive_starts` and `passive_ends`, this string may be present to indicate which passive finder algorithm works best for this dataset.

echofilter.raw.shardloader module

Converting raw data into shards, and loading data from shards.

`echofilter.raw.shardloader.`**load_transect_from_shards**(*transect_rel_pth*, *i1=0*, *i2=None*, *dataset='mobile'*, *segment=0*, *root_data_dir='/data/dsforce/surveyExports'*, ***kwargs*)

Load transect data from shard files.

Parameters

- **transect_rel_pth** (*str*) – Relative path to transect.
- **i1** (*int, optional*) – Index of first sample to retrieve. Default is `0`, the first sample.
- **i2** (*int, optional*) – Index of last sample to retrieve. As-per python convention, the range `i1` to `i2` is inclusive on the left and exclusive on the right, so datapoint *i2 - 1* is the rightmost datapoint loaded. Default is `None`, which loads everything up to and including to the last sample.
- **dataset** (*str, optional*) – Name of dataset. Default is `"mobile"`.
- **segment** (*int, optional*) – Which segment to load. Default is `0`.
- **root_data_dir** (*str*) – Path to root directory where data is located.
- ****kwargs** – As per *load_transect_from_shards_abs()*.

Returns See *load_transect_from_shards_abs()*.

Return type dict

echofilter.raw.shardloader.**load_transect_from_shards_abs**(*transect_abs_pth*, *i1=0*, *i2=None*, *pad_mode='edge'*)

Load transect data from shard files.

Parameters

- **transect_abs_pth** (*str*) – Absolute path to transect shard directory.
- **i1** (*int, optional*) – Index of first sample to retrieve. Default is `0`, the first sample.
- **i2** (*int, optional*) – Index of last sample to retrieve. As-per python convention, the range `i1` to `i2` is inclusive on the left and exclusive on the right, so datapoint *i2 - 1* is the right-most datapoint loaded. Default is `None`, which loads everything up to and including to the last sample.
- **pad_mode** (*str, optional*) – Padding method for out-of-bounds inputs. Must be supported by `numpy.pad()`, such as `"contast"`, `"reflect"`, or `"edge"`. If the mode is `"contast"`, the array will be padded with zeros. Default is "edge".

Returns

A dictionary with keys:

- **"timestamps"** [numpy.ndarray] Timestamps (in seconds since Unix epoch), for each recording timepoint. The number of entries, `num_timestamps`, is equal to *i2 - i1*.
- **"depths"** [numpy.ndarray] Depths from the surface (in metres), with each entry corresponding to each column in the `signals` data.
- **"Sv"** [numpy.ndarray] Echogram Sv data, shaped (num_timestamps, num_depths).
- **"mask"** [numpy.ndarray] Logical array indicating which datapoints were kept (`True`) and which removed (`False`) for the masked Sv output. Shaped (num_timestamps, num_depths).
- **"turbulence"** [numpy.ndarray] For each timepoint, the depth of the shallowest datapoint which should be included for the mask. Shaped (num_timestamps,).
- **"bottom"** [numpy.ndarray] For each timepoint, the depth of the deepest datapoint which should be included for the mask. Shaped (num_timestamps,).
- **"is_passive"** [numpy.ndarray] Logical array showing whether a timepoint is of passive data. Shaped (num_timestamps,). All passive recording data should be excluded by the mask.
- **"is_removed"** [numpy.ndarray] Logical array showing whether a timepoint is entirely removed by the mask. Shaped (num_timestamps,). Does not include periods of passive recording.
- **"is_upward_facing"** [bool] Indicates whether the recording source is located at the deepest depth (i.e. the seabed), facing upwards. Otherwise, the recording source is at the shallowest depth (i.e. the surface), facing downwards.

Return type dict

echofilter.raw.shardloader.**load_transect_from_shards_rel**(*transect_rel_pth*, *i1=0*, *i2=None*, *dataset='mobile'*, *segment=0*, *root_data_dir='/data/dsforce/surveyExports'*, ***kwargs*)

Load transect data from shard files.

Parameters

- **transect_rel_pth** (*str*) – Relative path to transect.

- **i1** (*int, optional*) – Index of first sample to retrieve. Default is `0`, the first sample.
- **i2** (*int, optional*) – Index of last sample to retrieve. As-per python convention, the range `i1` to `i2` is inclusive on the left and exclusive on the right, so datapoint *i2 - 1* is the right-most datapoint loaded. Default is `None`, which loads everything up to and including to the last sample.
- **dataset** (*str, optional*) – Name of dataset. Default is `"mobile"`.
- **segment** (*int, optional*) – Which segment to load. Default is `0`.
- **root_data_dir** (*str*) – Path to root directory where data is located.
- ****kwargs** – As per *load_transect_from_shards_abs()*.

Returns See *load_transect_from_shards_abs()*.

Return type dict

echofilter.raw.shardloader.**load_transect_segments_from_shards_abs**(*transect_abs_pth*, *segments=None*)

Load transect data from shard files.

Parameters

- **transect_abs_pth** (*str*) – Absolute path to transect shard segments directory.
- **segments** (*iterable or None*) – Which segments to load. If `None` (default), all segments are loaded.

Returns See *load_transect_from_shards_abs()*.

Return type dict

echofilter.raw.shardloader.**load_transect_segments_from_shards_rel**(*transect_rel_pth*, *dataset='mobile'*, *segments=None*, *root_data_dir='/data/dsforce/surveyExports'*)

Load transect data from shard files.

Parameters

- **transect_rel_pth** (*str*) – Relative path to transect.
- **dataset** (*str, optional*) – Name of dataset. Default is `"mobile"`.
- **segments** (*iterable or None*) – Which segments to load. If `None` (default), all segments are loaded.
- **root_data_dir** (*str*) – Path to root directory where data is located.
- ****kwargs** – As per *load_transect_from_shards_abs()*.

Returns See *load_transect_from_shards_abs()*.

Return type dict

echofilter.raw.shardloader.**segment_and_shard_transect**(*transect_pth*, *dataset='mobile'*, *max_depth=None*, *shard_len=128*, *root_data_dir='/data/dsforce/surveyExports'*)

Create a sharded copy of a transect.

The transect is cut into segments based on recording starts/stops. Each segment is split across multiple files (shards) for efficient loading.

Parameters

- **transect_pth** (*str*) – Relative path to transect, excluding "_Sv_raw.csv".
- **dataset** (*str, optional*) – Name of dataset. Default is "mobile".
- **max_depth** (*float or None, optional*) – The maximum depth to include in the saved shard. Data corresponding to deeper locations is omitted to save on load time and memory when the shard is loaded. If None, no cropping is applied. Default is None.
- **shard_len** (*int, optional*) – Number of timestamp samples to include in each shard. Default is 128.
- **root_data_dir** (*str*) – Path to root directory where data is located.

Notes

The segments will be written to the directories <root_data_dir>_sharded/<dataset>/transect_path// For the contents of each directory, see write_transect_shards.

echofilter.raw.shardloader.**shard_transect**(*transect_pth*, *dataset='mobile'*, *max_depth=None*, *shard_len=128*, *root_data_dir='/data/dsforce/surveyExports'*)

Create a sharded copy of a transect.

The transect is cut into segments based on recording starts/stops. Each segment is split across multiple files (shards) for efficient loading.

Parameters

- **transect_pth** (*str*) – Relative path to transect, excluding "_Sv_raw.csv".
- **dataset** (*str, optional*) – Name of dataset. Default is "mobile".
- **max_depth** (*float or None, optional*) – The maximum depth to include in the saved shard. Data corresponding to deeper locations is omitted to save on load time and memory when the shard is loaded. If None, no cropping is applied. Default is None.
- **shard_len** (*int, optional*) – Number of timestamp samples to include in each shard. Default is 128.
- **root_data_dir** (*str*) – Path to root directory where data is located.

Notes

The segments will be written to the directories <root_data_dir>_sharded/<dataset>/transect_path// For the contents of each directory, see write_transect_shards.

echofilter.raw.shardloader.**write_transect_shards**(*dirname*, *transect*, *max_depth=None*, *shard_len=128*)

Create a sharded copy of a transect.

The transect is cut by timestamp and split across multiple files.

Parameters

- **dirname** (*str*) – Path to output directory.
- **transect** (*dict*) – Observed values for the transect. Should already be segmented.

- **max_depth** (*float or None, optional*) – The maximum depth to include in the saved shard. Data corresponding to deeper locations is omitted to save on load time and memory when the shard is loaded. If `None`, no cropping is applied. Default is `None`.
- **shard_len** (*int, optional*) – Number of timestamp samples to include in each shard. Default is `128`.

Notes

The output will be written to the directory `dirname`, and will contain:

- a file named `"shard_size.txt"`, which contains the sharding metadata: total number of samples, and shard size;
- a directory for each shard, named 0, 1, … Each shard directory will contain files:
  - depths.npy
  - timestamps.npy
  - Sv.npy
  - mask.npy
  - turbulence.npy
  - bottom.npy
  - is_passive.npy
  - is_removed.npy
  - is_upward_facing.npy

  which contain pickled numpy dumps of the matrices for each shard.

echofilter.raw.utils module

Loader utility functions.

echofilter.raw.utils.**fillholes2d**(*arr*, *nan_thr=2*, *interp_method='linear'*, *inplace=False*)

Interpolate to replace NaN values in 2d gridded array data.

Parameters

- **arr** (*2d numpy.ndarray*) – Array in 2d which, may contain NaNs.
- **nan_thr** (*int, default=2*) – Minimum number of NaN values needed in a row/column for it to be included in the (rectangular) area where NaNs are fixed.
- **interp_method** (*str, default="linear"*) – Interpolation method.
- **inplace** (*bool, default=False*) – Whether to update arr instead of a copy.

Returns **arr** – Like input `arr`, but with NaN values replaced with interpolated values.

Return type 2d numpy.ndarray

echofilter.raw.utils.**integrate_area_of_contour**(*x*, *y*, *closed=None*, *preserve_sign=False*)

Compute the area within a contour, using Green's algorithm.

Parameters

- **x** (*array_like vector*) – x co-ordinates of nodes along the contour.

- **y** (*array_like vector*) – y co-ordinates of nodes along the contour.
- **closed** (*bool or None, optional*) – Whether the contour is already closed. If `False`, it will be closed before deterimining the area. If `None` (default), it is automatically determined as to whether the contour is already closed, and is closed if necessary.
- **preserve_sign** (*bool, optional*) – Whether to preserve the sign of the area. If `True`, the area is positive if the contour is anti-clockwise and negative if it is clockwise oriented. Default is `False`, which always returns a positive area.

Returns **area** – The integral of the area witihn the contour.

Return type float

Notes

https://en.wikipedia.org/wiki/Green%27s_theorem#Area_calculation

echofilter.raw.utils.**interp1d_preserve_nan**(*x*, *y*, *x_samples*, *nan_threshold=0.0*, *bounds_error=False*, ***kwargs*)

Interpolate a 1-D function, preserving NaNs.

Inputs `x` and `y` are arrays of values used to approximate some function f: `y = f(x)`. We exclude NaNs for the interpolation and then mask out entries which are adjacent (or close to) a NaN in the input.

Parameters

- **x** (*(N,) array_like*) – A 1-D array of real values. Must not contain NaNs.
- **y** (*(...,N,...) array_like*) – A N-D array of real values. The length of `y` along the interpolation axis must be equal to the length of `x`. May contain NaNs.
- **x_samples** (*array_like*) – A 1-D array of real values at which the interpolation function will be sampled.
- **nan_threshold** (*float, optional*) – Minimum amount of influence a NaN must have on an output sample for it to become a NaN. Default is `0.` i.e. any influence.
- **bounds_error** (*bool, optional*) – If `True`, a ValueError is raised any time interpolation is attempted on a value outside of the range of `x` (where extrapolation is necessary). If `False` (default), out of bounds values are assigned value `fill_value` (whose default is NaN).
- ****kwargs** – Additional keyword arguments are as per `scipy.interpolate.interp1d()`.

Returns **y_samples** – The result of interpolating, with sample points close to NaNs in the input returned as NaN.

Return type (...,N,...) np.ndarray

echofilter.raw.utils.**medfilt1d**(*signal*, *kernel_size*, *axis=- 1*, *pad_mode='reflect'*)

Median filter in 1d, with support for selecting padding mode.

Parameters

- **signal** (*array_like*) – The signal to filter.
- **kernel_size** – Size of the median kernel to use.
- **axis** (*int, optional*) – Which axis to operate along. Default is `-1`.
- **pad_mode** (*str, optional*) – Method with which to pad the vector at the edges. Must be supported by `numpy.pad()`. Default is `"reflect"`.

Returns **filtered** – The filtered signal.

Return type array_like

See also:

`scipy.signal.medfilt`, *pad1d*

`echofilter.raw.utils.`**pad1d**(*array*, *pad_width*, *axis=0*, ***kwargs*)

Pad an array along a single axis only.

Parameters

- **array** (*numpy.ndarary*) – Array to be padded.
- **pad_width** (*int or tuple*) – The amount to pad, either a length two tuple of values for each edge, or an int if the padding should be the same for each side.
- **axis** (*int, optional*) – The axis to pad. Default is `0`.
- ****kwargs** – As per `numpy.pad()`.

Returns Padded array.

Return type numpy.ndarary

See also:

`numpy.pad`

`echofilter.raw.utils.`**squash_gaps**(*mask*, *max_gap_squash*, *axis=- 1*, *inplace=False*)

Merge small gaps between zero values in a boolean array.

Parameters

- **mask** (*boolean array*) – The input mask, with small gaps between zero values which will be squashed with zeros.
- **max_gap_squash** (*int*) – Maximum length of gap to squash.
- **axis** (*int, optional*) – Axis on which to operate. Default is `-1`.
- **inplace** (*bool, optional*) – Whether to operate on the original array. If `False`, a copy is created and returned.

Returns **merged_mask** – Mask as per the input, but with small gaps squashed.

Return type boolean array

echofilter.ui package

User interface.

Submodules

echofilter.ui.checkpoints module

Interacting with the list of available checkpoints.

class `echofilter.ui.checkpoints.`**ListCheckpoints**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: `argparse.Action`

class echofilter.ui.checkpoints.**ShowCacheDir**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: argparse.Action

echofilter.ui.checkpoints.**cannonise_checkpoint_name**(*name*)

Cannonises checkpoint name by removing extension.

Parameters **name** (*str*) – Name of checkpoint, possibly including extension.

Returns **name** – Name of checkpoint, with extension removed it matches a possible checkpoint file extension.

Return type str

echofilter.ui.checkpoints.**download_checkpoint**(*checkpoint_name*, *cache_dir=None*, *verbose=1*)

Download a checkpoint if it isn't already cached.

Parameters

- **checkpoint_name** (*str*) – Name of checkpoint to download.
- **cache_dir** (*str or None, optional*) – Path to local cache directory. If None (default), an OS-appropriate application-specific default cache directory is used.
- **verbose** (*int, optional*) – Verbosity level. Default is 1. Set to 0 to disable print statements.

Returns Path to downloaded checkpoint file.

Return type str

echofilter.ui.checkpoints.**get_checkpoint_list**()

List the currently available checkpoints, as stored in a local file.

Returns **checkpoints** – Dictionary with a key for each checkpoint. Each key maps to a dictionary whose elements describe the checkpoint.

Return type OrderedDict

echofilter.ui.checkpoints.**get_default_cache_dir**()

Determine the default cache directory.

echofilter.ui.checkpoints.**get_default_checkpoint**()

Get the name of the current default checkpoint.

Returns **checkpoint_name** – Name of current checkpoint.

Return type str

echofilter.ui.checkpoints.**load_checkpoint**(*ckpt_name=None*, *cache_dir=None*, *device='cpu'*, *return_name=False*, *verbose=1*)

Load a checkpoint, either from absolute path or the cache.

Parameters

- **checkpoint_name** (*str or None, optional*) – Path to checkpoint file, or name of checkpoint to download. Default is None.
- **cache_dir** (*str or None, optional*) – Path to local cache directory. If None (default), an OS-appropriate application-specific default cache directory is used.

- **device** (*str or torch.device or None, optional*) – Device onto which weight tensors will be mapped. If `None`, no mapping is performed and tensors will be loaded onto the same device as they were on when saved (which will result in an error if the device is not present). Default is `"cpu"`.
- **return_name** (*bool, optional*) – If `True`, a tuple is returned indicting the name of the checkpoint which was loaded. This is useful if the default checkpoint was loaded. Default is `False`.
- **verbose** (*int, optional*) – Verbosity level. Default is `1`. Set to `0` to disable print statements.

Returns

- **checkpoint** (*dict*) – Loaded checkpoint.
- **checkpoint_name** (*str, optional*) – If `return_name` is `True`, the name of the checkpoint is also returned.

echofilter.ui.formatters module

Provides extensions to argparse.

class echofilter.ui.formatters.**DedentTextHelpFormatter**(*prog*, *indent_increment=2*, *max_help_position=24*, *width=None*)

Bases: `argparse.HelpFormatter`

Help message formatter.

Retains formatting of all help text, except from indentation. Leading new lines are also stripped.

class echofilter.ui.formatters.**FlexibleHelpFormatter**(*prog*, *indent_increment=2*, *max_help_position=24*, *width=None*)

Bases: `argparse.HelpFormatter`

Help message formatter which can handle different formatting specifications.

The following formatters are supported:

"R|" Raw. will be left as is, processed using `argparse.RawTextHelpFormatter`.

"d|" Raw except for indentation. Will be dedented and leading newlines stripped only, processed using `argparse.RawTextHelpFormatter`.

The format specifier will be stripped from the text.

Notes

Based on https://stackoverflow.com/a/22157266/1960959 and https://sourceforge.net/projects/ruamel-std-argparse/.

echofilter.ui.formatters.**format_parser_for_sphinx**(*parser*)

Pre-format parser help for sphinx-argparse processing.

Parameters **parser** (*argparse.ArgumentParser*) – Initial argument parser.

Returns **parser** – The same argument parser, but with raw help text touched up so it renders correctly when passed through sphinx-argparse.

Return type argparse.ArgumentParser

echofilter.ui.inference_cli module

Provides a command line interface for the inference routine.

This is separated out from inference.py so the responsiveness for simple commands like `--help` and `--version` is faster, not needing to import the full dependency stack.

class echofilter.ui.inference_cli.**ListColors**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: argparse.Action

echofilter.ui.inference_cli.**cli**(*args=None*)

Run `run_inference()` with arguments taken from the command line.

echofilter.ui.inference_cli.**get_parser**()

Build parser for inference command line interface.

Returns **parser** – CLI argument parser for inference.

Return type argparse.ArgumentParser

echofilter.ui.inference_cli.**main**(*args=None*)

Run `cli`, with encapsulation for error messages.

echofilter.ui.style module

User interface styling, using ANSI codes and colorama.

class echofilter.ui.style.**AsideStyle**

Bases: `echofilter.ui.style._AbstractStyle`

Defines the style for aside text; dim style.

reset = '\x1b[22m'

start = '\x1b[2m'

class echofilter.ui.style.**DryrunStyle**

Bases: `echofilter.ui.style._AbstractStyle`

Defines the style for dry-run text; magenta foreground.

reset = '\x1b[39m'

start = '\x1b[35m'

class echofilter.ui.style.**ErrorStyle**

Bases: `echofilter.ui.style._AbstractStyle`

Defines the style for an error string; red foreground.

reset = '\x1b[39m'

start = '\x1b[31m'

class echofilter.ui.style.**HighlightStyle**

Bases: `echofilter.ui.style._AbstractStyle`

Defines the style for highlighted text; bright style.

reset = '\x1b[22m'

start = '\x1b[1m'

class echofilter.ui.style.**OverwriteStyle**

Bases: echofilter.ui.style._AbstractStyle

Defines the style for overwrite text; bright blue.

reset = '\x1b[39m\x1b[22m'

start = '\x1b[34m\x1b[1m'

class echofilter.ui.style.**ProgressStyle**

Bases: echofilter.ui.style._AbstractStyle

Defines the style for a progress string; green foreground.

reset = '\x1b[39m'

start = '\x1b[32m'

class echofilter.ui.style.**SkipStyle**

Bases: echofilter.ui.style._AbstractStyle

Defines the style for skip text; yellow foreground.

reset = '\x1b[39m'

start = '\x1b[33m'

class echofilter.ui.style.**WarningStyle**

Bases: echofilter.ui.style._AbstractStyle

Defines the style for a warning string; cyan foreground.

reset = '\x1b[39m'

start = '\x1b[36m'

echofilter.ui.style.**aside_fmt**(*string*)

Wrap a string in ANSI codes to render it in an aside (de-emphasised) style.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

echofilter.ui.style.**dryrun_fmt**(*string*)

Wrap a string in ANSI codes to render it in the style of dry-run text.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

echofilter.ui.style.**error_fmt**(*string*)

Wrap a string in ANSI codes to render it in the style of an error.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

class echofilter.ui.style.**error_message**(*message=''*)

Bases: contextlib.AbstractContextManager

Wrap an error message in ANSI codes to stylise its as red and bold.

The style is applied when printed at the terminal. If the context is exited with an error, that error message will be red.

Parameters **message** (*str*) – Text of the error message to stylise.

Returns Stylised message.

Return type str

echofilter.ui.style.**highlight_fmt**(*string*)

Wrap a string in ANSI codes to render it in a highlighted style.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

echofilter.ui.style.**overwrite_fmt**(*string*)

Wrap a string in ANSI codes to render it in the style of an overwrite.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

echofilter.ui.style.**progress_fmt**(*string*)

Wrap a string in ANSI codes to render it in the style of progress text.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

echofilter.ui.style.**skip_fmt**(*string*)

Wrap a string in ANSI codes to render it in the style of a skip message.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

echofilter.ui.style.**warning_fmt**(*string*)

Wrap a string in ANSI codes to render it in the style of a warning.

The style is applied when printed at the terminal.

Parameters **string** (*str*) – Input string to format.

Returns **formatted_string** – String prepended with a start ANSI code and appended with a reset ANSI code which undoes the start code.

Return type str

class echofilter.ui.style.**warning_message**(*message=''*)

Bases: contextlib.AbstractContextManager

Wrap a warning message in ANSI codes to stylise it as cyan and bold.

The style is applied when printed at the terminal. All statements printed during the context will be in cyan.

Parameters **message** (*str*) – Text of the warning message to stylise.

Returns Stylised message.

Return type str

echofilter.ui.train_cli module

Provides a command line interface for the training routine.

This is separated out from train.py so the documentation can be accessed without having all the training dependencies installed.

echofilter.ui.train_cli.**get_parser**()

Build parser for training command line interface.

Returns **parser** – CLI argument parser for training.

Return type argparse.ArgumentParser

echofilter.ui.train_cli.**main**(*args=None*)

Run command line interface for model training.

echofilter.win package

Window management and Echoview integration.

Submodules

echofilter.win.ev module

Echoview interface management.

echofilter.win.ev.**maybe_open_echoview**(*app=None*, *do_open=True*, *minimize=False*, *hide='new'*)

If the current pointer to the Echoview is invalid, open an Echoview window.

Parameters

- **app** (*COM object or None, optional*) – Existing COM object to interface with Echoview.
- **do_open** (*bool, optional*) – If `False` (dry-run mode), we don't actually need Echoview open and so don't try to open it. In this case, `None` is yielded. Present so a context manager can be used even if the application isn't opened. Default is `True`, do open Echoview.
- **minimize** (*bool, optional*) – If `True`, the Echoview window being used will be minimized while the code runs. Default is `False`.
- **hide** (*{"never", "new", "always"}, optional*) – Whether to hide the Echoview window entirely. If `hide="new"`, the application is only hidden if it was created by this context, and not if it was already running. If `hide="always"`, the application is hidden even if it was already running. In the latter case, the window will be revealed again when leaving this context. Default is `"new"`.

echofilter.win.ev.**open_ev_file**(*filename*, *app=None*)

Open an EV file within a context.

Parameters

- **filename** (*str*) – Path to file to open.
- **app** (*COM object or None, optional*) – Existing COM object to interface with Echoview. If `None`, a new COM interface is created. If that requires opening a new instance of Echoview, it is hidden while the file is in use.

echofilter.win.manager module

Window management for Windows.

class echofilter.win.manager.**WindowManager**(*title=None*, *class_name=None*, *title_pattern=None*)

Bases: `object`

Encapsulates calls to window management using the Windows api.

Notes

Based on: https://stackoverflow.com/a/2091530 and https://stackoverflow.com/a/4440622

check_handles_visible()

Check and remember which of self.handles are currently visible.

find_window(*class_name=None*, *title=None*)

Find a window by its exact title.

find_window_regex(*pattern*)

Find a window whose title matches a regular expression.

hide()

Hide the window.

hide_all()

Hide all the windows.

reset()

Clear handle attribute state.

set_foreground()

Bring the window to the foreground.

show()

Show the window.

show_all(*only_hidden=True*)

Show all the windows.

echofilter.win.manager.**opencom**(*com_name*, *can_make_anew=False*, *title=None*, *title_pattern=None*, *minimize=False*, *hide='never'*)

Open a connection to an application with a COM object.

The application may or may not be open before this context begins. If it was not already open, the application is closed when leaving the context.

Parameters

- **com_name** (*str*) – Name of COM object to dispatch.
- **can_make_anew** (*bool, optional*) – Whether arbitrarily many sessions of the COM object can be created, and if so whether they should be. Default is `False`, in which case the context manager will check to see if the application is already running before connecting to it. If it was already running, it will not be closed when this context closes.
- **title** (*str, optional*) – Exact title of window. If the title can not be determined exactly, use `title_pattern` instead.
- **title_pattern** (*str, optional*) – Regular expression for the window title.
- **minimize** (*bool, optional*) – If `True`, the application will be minimized while the code runs. Default is `False`.
- **hide** (*{"never", "new", "always"}, optional*) – Whether to hide the application window entirely. Default is `"never"`. If this is enabled, at least one of `title` and `title_pattern` must be specified. If `hide="new"`, the application is only hidden if it was created by this context, and not if it was already running. If `hide="always"`, the application is hidden even if it was already running. In the latter case, the window will be revealed again when leaving this context.

Yields *win32com.gen_py* – Interface to COM object.

3.1.2 Submodules

3.1.3 echofilter.ev2csv module

Export raw EV files in CSV format.

echofilter.ev2csv.**ev2csv**(*input*, *destination*, *variable_name='Fileset1: Sv pings T1'*, *export_raw=True*, *ev_app=None*, *verbose=0*)

Export a single EV file to CSV.

Parameters

- **input** (*str*) – Path to input file.
- **destination** (*str*) – Filename of output destination.
- **variable_name** (*str, optional*) – Name of the Echoview acoustic variable to export. Default is *"Fileset1: Sv pings T1"*.
- **export_raw** (*bool, optional*) – If `True` (default), exclusion and threshold settings in the EV file are temporarily disabled before exporting the CSV, in order to ensure all raw data is exported.
- **ev_app** (*win32com.client.Dispatch object or None, optional*) – An object which can be used to interface with the Echoview application, as returned by `win32com.client.Dispatch`. If `None` (default), a new instance of the application is opened (and closed on completion).
- **verbose** (*int, optional*) – Level of verbosity. Default is `0`.

Returns **destination** – Absolute path to `destination`.

Return type str

echofilter.ev2csv.**get_parser**()

Build parser for ev2csv command line interface.

Returns **parser** – CLI argument parser for ev2csv.

Return type argparse.ArgumentParser

echofilter.ev2csv.**main**(*args=None*)

Run ev2csv command line interface.

echofilter.ev2csv.**run_ev2csv**(*paths*, *variable_name='Fileset1: Sv pings T1'*, *export_raw=True*, *source_dir='.'*, *recursive_dir_search=True*, *output_dir=''*, *suffix=None*, *keep_ext=False*, *skip_existing=False*, *overwrite_existing=False*, *minimize_echoview=False*, *hide_echoview='new'*, *verbose=1*, *dry_run=False*)

Export EV files to raw CSV files.

Parameters

- **paths** (*iterable*) – Paths to input EV files to process, or directories containing EV files. These may be full paths or paths relative to `source_dir`. For each folder specified, any files with extension `"csv"` within the folder and all its tree of subdirectories will be processed.
- **variable_name** (*str, optional*) – Name of the Echoview acoustic variable to export. Default is *"Fileset1: Sv pings T1"*.
- **export_raw** (*bool, optional*) – If `True` (default), exclusion and threshold settings in the EV file are temporarily disabled before exporting the CSV, in order to ensure all raw data is exported. If `False`, thresholds and exclusions are used as per the EV file.

- **source_dir** (*str, optional*) – Path to directory where files are found. Default is ".".
- **recursive_dir_search** (*bool, optional*) – How to handle directory inputs in paths. If False, only files (with the correct extension) in the directory will be included. If True, subdirectories will also be walked through to find input files. Default is True.
- **output_dir** (*str, optional*) – Directory where output files will be written. If this is an empty string ("", default), outputs are written to the same directory as each input file. Otherwise, they are written to output_dir, preserving their path relative to source_dir if relative paths were used.
- **suffix** (*str, optional*) – Output filename suffix. Default is "_Sv_raw.csv" if keep_ext=False, or ".Sv_raw.csv" if keep_ext=True. The "_raw" component is excluded if export_raw is False.
- **keep_ext** (*bool, optional*) – Whether to preserve the file extension in the input file name when generating output file name. Default is False, removing the extension.
- **skip_existing** (*bool, optional*) – Whether to skip processing files whose destination paths already exist. If False (default), an error is raised if the destination file already exists.
- **overwrite_existing** (*bool, optional*) – Whether to overwrite existing output files. If False (default), an error is raised if the destination file already exists.
- **minimize_echoview** (*bool, optional*) – If True, the Echoview window being used will be minimized while this function is running. Default is False.
- **hide_echoview** (*{"never", "new", "always"}, optional*) – Whether to hide the Echoview window entirely while the code runs. If hide_echoview="new", the application is only hidden if it was created by this function, and not if it was already running. If hide_echoview="always", the application is hidden even if it was already running. In the latter case, the window will be revealed again when this function is completed. Default is "new".
- **verbose** (*int, optional*) – Level of verbosity. Default is 1.
- **dry_run** (*bool, optional*) – If True, perform a trial run with no changes made. Default is False.

Returns Paths to generated CSV files.

Return type list of str

3.1.4 echofilter.generate_shards module

Convert dataset of CSV exports from Echoview into shards.

echofilter.generate_shards.**generate_shard**(*transect_pth*, *verbose=False*, *fail_gracefully=True*, ***kwargs*)

Shard a single transect.

Wrapper around echofilter.raw.shardloader.segment_and_shard_transect which adds verboseness and graceful failure options.

Parameters

- **transect_pth** (*str*) – Relative path to transect.
- **verbose** (*bool, optional*) – Whether to print which transect is being processed. Default is False.

- **fail_gracefully** (*bool, optional*) – If `True`, any transect which triggers an errors during processing will be printed out, but processing the rest of the transects will continue. If `False`, the process will halt with an error as soon as any single transect hits an error. Default is `True`.
- ****kwargs** – See *echofilter.raw.shardloader.segment_and_shard_transect()*.

echofilter.generate_shards.**generate_shards**(*partition*, *dataset*, *partitioning_version='firstpass'*, *progress_bar=False*, *ncores=None*, *verbose=False*, *fail_gracefully=True*, *root_data_dir='/data/dsforce/surveyExports'*, ***kwargs*)

Shard all transections in one partition of a dataset.

Wrapper around echofilter.raw.shardloader.segment_and_shard_transect which adds verboseness and graceful failure options.

Parameters

- **partition** (*str*) – Name of the partition to process (`'train'`, `'validate'`, `'test'`, etc).
- **dataset** (*str*) – Name of the dataset to process (`'mobile'`, `'MinasPassage'`, etc).
- **partitioning_version** (*str, optional*) – Name of the partition version to use process. Default is `'firstpass'`.
- **progress_bar** (*bool, optional*) – Whether to output a progress bar using `tqdm`. Default is `False`.
- **ncores** (*int, optional*) – Number of cores to use for multiprocessing. To disable multiprocessing, set to `1`. Set to `None` to use all available cores. Default is `None`.
- **verbose** (*bool, optional*) – Whether to print which transect is being processed. Default is `False`.
- **fail_gracefully** (*bool, optional*) – If `True`, any transect which triggers an errors during processing will be printed out, but processing the rest of the transects will continue. If `False`, the process will halt with an error as soon as any single transect hits an error. Default is `True`.
- ****kwargs** – See *echofilter.raw.shardloader.segment_and_shard_transect()*.

echofilter.generate_shards.**get_parser**()

Build parser for command line interface for generating shards.

Returns **parser** – CLI argument parser for generating shards.

Return type argparse.ArgumentParser

echofilter.generate_shards.**main**(*args=None*)

Command line interface for generating dataset shards from CSV files.

3.1.5 echofilter.inference module

Inference routine.

echofilter.inference.**get_color_palette**(*include_xkcd=True*, *sort_colors=True*)

Provide a mapping of named colors from matplotlib.

Parameters

- **include_xkcd** (*bool, default=True*) – Whether to include the XKCD color palette in the output. Note that XKCD colors have `"xkcd:"` prepended to their names to prevent collisions with official named colors from CSS4. See https://xkcd.com/color/rgb/ and https://blog.xkcd.com/2010/05/03/color-survey-results/ for the XKCD colors.
- **sort_colors** (*bool, default=True*) – Whether to sort the colors by hue. Otherwise the colors are grouped together by source, and maintain their default ordering (alphabetized).

Returns **colors** – Mapping from names of colors as strings to color value, either as an RGB tuple (fractional, 0 to 1 range) or a hexadecimal string.

Return type dict

echofilter.inference.**hexcolor2rgb8**(*color*)

Map hexadecimal colors to uint8 RGB.

Parameters **color** (*str*) – A hexadecimal color string, with leading `"#"`. If the input is not a string beginning with `"#"`, it is returned as-is without raising an error.

Returns RGB color tuple, in uint8 format (0–255).

Return type tuple

echofilter.inference.**import_lines_regions_to_ev**(*ev_fname*, *files*, *target_names=None*, *nearfield_depth=None*, *add_nearfield_line=True*, *lines_cutoff_at_nearfield=None*, *offsets=None*, *line_colors=None*, *line_thicknesses=None*, *ev_app=None*, *overwrite=False*, *common_notes=''*, *verbose=1*)

Write lines and regions to EV file.

Parameters

- **ev_fname** (*str*) – Path to Echoview file to import variables into.
- **files** (*dict*) – Mapping from output keys to filenames.
- **target_names** (*dict, optional*) – Mapping from output keys to output variable names.
- **nearfield_depth** (*float, optional*) – Depth at which nearfield line will be placed. By default, no nearfield line will be added, irrespective of `add_nearfield_line`.
- **add_nearfield_line** (*bool, default=True*) – Whether to add a nearfield line.
- **lines_cutoff_at_nearfield** (*list of str, optional*) – Which lines (if any) should be clipped at the nearfield depth. By default, no lines will be clipped.
- **offsets** (*dict, optional*) – Amount of offset for each line.
- **line_colors** (*dict, optional*) – Mapping from output keys to line colours.
- **line_thicknesses** (*dict, optional*) – Mapping from output keys to line thicknesses.

- **ev_app** (*win32com.client.Dispatch object, optional*) – An object which can be used to interface with the Echoview application, as returned by `win32com.client.Dispatch`. By default, a new instance of the application is opened (and closed on completion).
- **overwrite** (*bool, default=False*) – Whether existing lines with target names should be replaced. If a line with the target name already exists and `overwrite=False`, the line is named with the current datetime to prevent collisions.
- **common_notes** (*str, default=""*) – Notes to include for every region.
- **verbose** (*int, default=1*) – Verbosity level.

echofilter.inference.**inference_transect**(*model*, *timestamps*, *depths*, *signals*, *device*, *image_height*, *facing='auto'*, *crop_min_depth=None*, *crop_max_depth=None*, *autocrop_threshold=0.35*, *force_unconditioned=False*, *data_center='mean'*, *data_deviation='stdev'*, *prenorm_nan_value=None*, *postnorm_nan_value=- 3*, *dtype=torch.float32*, *verbose=0*)

Run inference on a single transect.

Parameters

- **model** (*echofilter.wrapper.Echofilter*) – A pytorch Module wrapped in an Echofilter UI layer.
- **timestamps** (*array_like*) – Sample recording timestamps (in seconds since Unix epoch). Must be a vector.
- **depths** (*array_like*) – Recording depths from the surface (in metres). Must be a vector.
- **signals** (*array_like*) – Echogram Sv data. Must be a matrix shaped *(len(timestamps), len(depths))*.
- **image_height** (*int*) – Height to resize echogram before passing through model.
- **facing** (*{"downward", "upward", "auto"}, default="auto"*) – Orientation in which the echosounder is facing. Default is `"auto"`, in which case the orientation is determined from the ordering of the depth values in the data (increasing = `"upward"`, decreasing = `"downward"`).
- **crop_min_depth** (*float, optional*) – Minimum depth to include in input. By default, there is no minimum depth.
- **crop_max_depth** (*float, optional*) – Maxmimum depth to include in input. By default, there is no maximum depth.
- **autocrop_threshold** (*float, default=0.35*) – Minimum fraction of input height which must be found to be removable for the model to be re-run with an automatically cropped input.
- **force_unconditioned** (*bool, optional*) – Whether to always use unconditioned logit outputs when deteriming the new depth range for automatic cropping.
- **data_center** (*float or str, default="mean"*) – Center point to use, which will be subtracted from the Sv signals (i.e. the overall sample mean). If `data_center` is a string, it specifies the method to use to determine the center value from the distribution of intensities seen in this sample transect.
- **data_deviation** (*float or str, default="stdev"*) – Deviation to use to normalise the Sv signals in divisive manner (i.e. the overall sample standard deviation). If

data_deviation is a string, it specifies the method to use to determine the center value from the distribution of intensities seen in this sample transect.

- **prenorm_nan_value** (*float, optional*) – If this is set, replace NaN values with a given Sv value before the data normalisation (Gaussian standardisation) step. By default, NaNs are left as they are until after standardising the data.
- **postnorm_nan_value** (*float, default=-3*) – Placeholder value to replace NaNs with. Does nothing if prenorm_nan_value is set.
- **dtype** (*torch.dtype, default=torch.float*) – Datatype to use for model input.
- **verbose** (*int, default=0*) – Level of verbosity.

Returns Dictionary with fields as output by echofilter.wrapper.Echofilter, plus timestamps and depths.

Return type dict

echofilter.inference.**run_inference**(*paths*, *source_dir='.'*, *recursive_dir_search=True*, *extensions='csv'*, *skip_existing=False*, *skip_incompatible=False*, *output_dir=''*, *dry_run=False*, *continue_on_error=False*, *overwrite_existing=False*, *overwrite_ev_lines=False*, *import_into_evfile=True*, *generate_turbulence_line=True*, *generate_bottom_line=True*, *generate_surface_line=True*, *add_nearfield_line=True*, *suffix_file=''*, *suffix_var=None*, *color_turbulence='orangered'*, *color_turbulence_offset=None*, *color_bottom='orangered'*, *color_bottom_offset=None*, *color_surface='green'*, *color_surface_offset=None*, *color_nearfield='mediumseagreen'*, *thickness_turbulence=2*, *thickness_turbulence_offset=None*, *thickness_bottom=2*, *thickness_bottom_offset=None*, *thickness_surface=1*, *thickness_surface_offset=None*, *thickness_nearfield=1*, *cache_dir=None*, *cache_csv=None*, *suffix_csv=''*, *keep_ext=False*, *line_status=3*, *offset_turbulence=1.0*, *offset_bottom=1.0*, *offset_surface=1.0*, *nearfield=1.7*, *cutoff_at_nearfield=None*, *lines_during_passive='interpolate-time'*, *collate_passive_length=10*, *collate_removed_length=10*, *minimum_passive_length=10*, *minimum_removed_length=- 1*, *minimum_patch_area=- 1*, *patch_mode=None*, *variable_name='Fileset1: Sv pings T1'*, *export_raw_csv=True*, *row_len_selector='mode'*, *facing='auto'*, *use_training_standardization=False*, *prenorm_nan_value=None*, *postnorm_nan_value=None*, *crop_min_depth=None*, *crop_max_depth=None*, *autocrop_threshold=0.35*, *image_height=None*, *checkpoint=None*, *force_unconditioned=False*, *logit_smoothing_sigma=0*, *device=None*, *hide_echoview='new'*, *minimize_echoview=False*, *verbose=2*)

Perform inference on input files, and generate output files.

Outputs are written as lines in EVL and regions in EVR file formats.

Parameters

- **paths** (*iterable or str*) – Files and folders to be processed. These may be full paths or paths relative to source_dir. For each folder specified, any files with extension "csv" within the folder and all its tree of subdirectories will be processed.
- **source_dir** (*str, default="."*) – Path to directory where files are found.

- **recursive_dir_search** (*bool, default=True*) – How to handle directory inputs in `paths`. If `False`, only files (with the correct extension) in the directory will be included. If `True`, subdirectories will also be walked through to find input files.
- **extensions** (*iterable or str, default="csv"*) – File extensions to detect when running on a directory.
- **skip_existing** (*bool, default=False*) – Skip processing files which already have all outputs present.
- **skip_incompatible** (*bool, default=False*) – Skip processing CSV files which do not seem to contain an exported Echoview transect. If `False`, an error is raised.
- **output_dir** (*str, default=""*) – Directory where output files will be written. If this is an empty string, outputs are written to the same directory as each input file. Otherwise, they are written to `output_dir`, preserving their path relative to `source_dir` if relative paths were used.
- **dry_run** (*bool, default=False*) – If `True`, perform a trial run with no changes made.
- **continue_on_error** (*bool, default=False*) – Continue running on remaining files if one file hits an error.
- **overwrite_existing** (*bool, default=False*) – Overwrite existing outputs without producing a warning message. If `False`, an error is generated if files would be overwritten.
- **overwrite_ev_lines** (*bool, default=False*) – Overwrite existing lines within the Echoview file without warning. If `False` (default), the current datetime will be appended to line variable names in the event of a collision.
- **import_into_evfile** (*bool, default=True*) – Whether to import the output lines and regions into the EV file, whenever the file being processed in an EV file.
- **generate_turbulence_line** (*bool, default=True*) – Whether to output an evl file for the turbulence line. If this is `False`, the turbulence line is also never imported into Echoview.
- **generate_bottom_line** (*bool, default=True*) – Whether to output an evl file for the bottom line. If this is `False`, the bottom line is also never imported into Echoview.
- **generate_surface_line** (*bool, default=True*) – Whether to output an evl file for the surface line. If this is `False`, the surface line is also never imported into Echoview.
- **add_nearfield_line** (*bool, default=True*) – Whether to add a nearfield line to the EV file in Echoview.
- **suffix_file** (*str, default=""*) – 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.
- **suffix_var** (*str, optional*) – Suffix to append to line and region names when imported back into EV file. If `suffix_var` begins with an alphanumeric character, `"-"` is prepended. By default, `suffix_var` will match `suffix_file` if it is set, and will be "_echofilter" otherwise.
- **color_turbulence** (*str, default="orangered"*) – Color to use for the turbulence line when it is imported into Echoview. This can either be the name of a supported color from matplotlib.colors, or a hexadecimal color, or a string representation of an RGB color to supply directly to Echoview (such as "(0,255,0)").
- **color_turbulence_offset** (*str, optional*) – Color to use for the offset turbulence line when it is imported into Echoview. By default, `color_turbulence` is used.

- **color_bottom** (*str, default="orangered"*) – Color to use for the bottom line when it is imported into Echoview. This can either be the name of a supported color from matplotlib.colors, or a hexadecimal color, or a string representation of an RGB color to supply directly to Echoview (such as "(0,255,0)").
- **color_bottom_offset** (*str, optional*) – Color to use for the offset bottom line when it is imported into Echoview. By default, `color_bottom` is used.
- **color_surface** (*str, default="green"*) – Color to use for the surface line when it is imported into Echoview. This can either be the name of a supported color from matplotlib.colors, or a hexadecimal color, or a string representation of an RGB color to supply directly to Echoview (such as "(0,255,0)").
- **color_surface_offset** (*str, optional*) – Color to use for the offset surface line when it is imported into Echoview. By default, `color_surface` is used.
- **color_nearfield** (*str, default="mediumseagreen"*) – Color to use for the nearfield line when it is created in Echoview. This can either be the name of a supported color from matplotlib.colors, or a hexadecimal color, or a string representation of an RGB color to supply directly to Echoview (such as "(0,255,0)").
- **thickness_turbulence** (*int, default=2*) – Thickness with which the turbulence line will be displayed in Echoview.
- **thickness_turbulence_offset** (*str, optional*) – Thickness with which the offset turbulence line will be displayed in Echoview. By default, `thickness_turbulence` is used.
- **thickness_bottom** (*int, default=2*) – Thickness with which the bottom line will be displayed in Echoview.
- **thickness_bottom_offset** (*str, optional*) – Thickness with which the offset bottom line will be displayed in Echoview. By default, `thickness_bottom` is used.
- **thickness_surface** (*int, default=1*) – Thickness with which the surface line will be displayed in Echoview.
- **thickness_surface_offset** (*str, optional*) – Thickness with which the offset surface line will be displayed in Echoview. By default, `thickness_surface` is used.
- **thickness_nearfield** (*int, default=1*) – Thickness with which the nearfield line will be displayed in Echoview.
- **cache_dir** (*str, optional*) – Path to directory where downloaded checkpoint files should be cached. By default, an OS-appropriate application-specific default cache directory is used.
- **cache_csv** (*str, optional*) – Path to directory where CSV files generated from EV inputs should be cached. By default, EV files which are exported to CSV files are temporary files, deleted after this program has completed. If `cache_csv=""`, the CSV files are cached in the same directory as the input EV files.
- **suffix_csv** (*str, default=""*) – Suffix used for cached CSV files which are exported from EV files. If `suffix_file` begins with an alphanumeric character, a delimiter is prepended. The delimiter is `"."` if `keep_ext=True` or `"-"` if `keep_ext=False`.
- **keep_ext** (*bool, default=False*) – Whether to preserve the file extension in the input file name when generating output file name. Default is `False`, removing the extension.
- **line_status** (*int, default=3*) – Status to use for the lines. Must be one of:
  - `0` : none
  - 1 : unverified

  - 2 : bad
  - 3 : good

- **offset_turbulence** (*float, default=1.0*) – Offset for turbulence line, which moves the turbulence line deeper.
- **offset_bottom** (*float, default=1.0*) – Offset for bottom line, which moves the line to become more shallow.
- **offset_surface** (*float, default=1.0*) – Offset for surface line, which moves the surface line deeper.
- **nearfield** (*float, default=1.7*) – Nearfield approach distance, in metres. If the echogram is downward facing, the nearfield cutoff depth will be at a depth equal to the nearfield distance. 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** (*bool, optional*) – Whether to cut-off the turbulence line (for downfacing data) or bottom line (for upfacing) when it is closer to the echosounder than the `nearfield` distance. By default, the bottom line is clipped (for upfacing data), but the turbulence line is not clipped (even with downfacing data).
- **lines_during_passive** (*str, default="interpolate-time"*) – 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`.
- **collate_passive_length** (*int, default=10*) – Maximum interval, in ping indices, between detected passive regions which will removed to merge consecutive passive regions together into a single, collated, region.
- **collate_passive_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.
- **minimum_passive_length** (*int, default=10*) – 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.
- **minimum_removed_length** (*int, default=-1*) – 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). Recommended minimum length is 10.

- **minimum_patch_area** (*int, default=-1*) – 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). Recommended minimum length 25.
- **patch_mode** (*str, optional*) – 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.

  By default, "merged" is used if downfacing and "ntob" is used if upfacing.
- **variable_name** (*str, default="Fileset1: Sv pings T1"*) – Name of the Echoview acoustic variable to load from EV files.
- **export_raw_csv** (*bool, default=True*) – If True (default), exclusion and threshold settings in the EV file are temporarily disabled before exporting the CSV, in order to ensure all raw data is exported. If False, thresholds and exclusions are used as per the EV file.
- **row_len_selector** (*str, default="mode"*) – Method used to handle input csv files with different number of Sv values across time (i.e. a non-rectangular input). See *echofilter.raw.loader.transect_loader()* for options.
- **facing** (*{"downward", "upward", "auto"}, default="auto"*) – Orientation in which the echosounder is facing. Default is "auto", in which case the orientation is determined from the ordering of the depth values in the data (increasing = "upward", decreasing = "downward").
- **use_training_standardization** (*bool, default=False*) – Whether to use the exact normalization center and deviation values as used during training. If False (default), the center and deviation are determined per sample, using the same method methodology as used to determine the center and deviation values for training.
- **prenorm_nan_value** (*float, optional*) – If this is set, replace NaN values with a given Sv value before the data normalisation (Gaussian standardisation) step. By default, NaNs are left as they are until after standardising the data.
- **postnorm_nan_value** (*float, optional*) – Placeholder value to replace NaNs with. Does nothing if prenorm_nan_value is set. By default this is set to the value used to train the model.
- **crop_min_depth** (*float, optional*) – Minimum depth to include in input. By default, there is no minimum depth.
- **crop_max_depth** (*float, optional*) – Maxmimum depth to include in input. By default, there is no maximum depth.
- **autocrop_threshold** (*float, default=0.35*) – Minimum fraction of input height which must be found to be removable for the model to be re-run with an automatically cropped input.
- **image_height** (*int, optional*) – Height in pixels of input to model. The data loaded from the csv will be resized to this height (the width of the image is unchanged). By default, the height matches that used when the model was trained.

- **checkpoint** (*str, optional*) – A path to a checkpoint file, or name of a checkpoint known to this package (listed in `echofilter/checkpoints.yaml`). By default, the first checkpoint in `checkpoints.yaml` is used.
- **force_unconditioned** (*bool, default=False*) – Whether to always use unconditioned logit outputs. If `False` (default) conditional logits will be used if the checkpoint loaded is for a conditional model.
- **logit_smoothing_sigma** (*float, optional*) – Standard deviation over which logits will be smoothed before being converted into output. Disabled by default.
- **device** (*str or torch.device, optional*) – Name of device on which the model will be run. By default, the first available CUDA GPU is used if any are found, and otherwise the CPU is used. Set to `"cpu"` to use the CPU even if a CUDA GPU is available.
- **hide_echoview** (*{"never", "new", "always"}, default="new"*) – Whether to hide the Echoview window entirely while the code runs. If `hide_echoview="new"`, the application is only hidden if it was created by this function, and not if it was already running. If `hide_echoview="always"`, the application is hidden even if it was already running. In the latter case, the window will be revealed again when this function is completed.
- **minimize_echoview** (*bool, default=False*) – If `True`, the Echoview window being used will be minimized while this function is running.
- **verbose** (*int, default=2*) – Verbosity level. Set to `0` to disable print statements, or elevate to a higher number to increase verbosity.

3.1.6 echofilter.path module

Path utilities.

`echofilter.path.`**check_if_windows**`()`

Check if the operating system is Windows.

Returns Whether the OS is Windows.

Return type bool

`echofilter.path.`**determine_destination**`(`*fname*, *fname_full*, *source_dir*, *output_dir*`)`

Determine where destination should be placed for a file, preserving subtree paths.

Parameters

- **fname** (*str*) – Original input path.
- **fname_full** (*str*) – Path to file, either absolute or relative; possibly containing `source_dir`.
- **source_dir** (*str*) – Path to a directory where the file bearing name `fname` is expected to be located.
- **output_dir** (*str*) – Path to root output directory.

Returns Path to where file can be found, either absolute or relative.

Return type str

`echofilter.path.`**determine_file_path**`(`*fname*, *source_dir*`)`

Determine the path to use to an input file.

Parameters

- **fname** (*str*) – Path to an input file. Either an absolute path, or a path relative to to `source_dir`, or a path relative to the working directory.
- **source_dir** (*str*) – Path to a directory where the file bearing name `fname` is expected to be located.

Returns Path to where file can be found, either absolute or relative.

Return type str

`echofilter.path.`**parse_files_in_folders**(*files_or_folders*, *source_dir*, *extension*, *recursive=True*)

Walk through folders and find suitable files.

Parameters

- **files_or_folders** (*iterable*) – List of files and folders.
- **source_dir** (*str*) – Root directory within which elements of `files_or_folders` may be found.
- **extension** (*str or Collection*) – Extension (or list of extensions) which files within directories must bear to be included, without leading `'.'`, for instance `'.csv'`. Note that explicitly given files are always used.
- **recursive** (*bool, optional*) – Whether to walk through the tree of files in a subfolders of a directory input. If `False`, only files in the folder itself and not its child folders will be included.

Yields *str* – Paths to explicitly given files and files within directories with extension `extension`.

3.1.7 echofilter.plotting module

Plotting utilities.

`echofilter.plotting.`**ensure_axes_inverted**(*axes=None*, *dir='y'*)

Invert axis direction, if not already inverted.

Parameters

- **axes** (*matplotlib.axes or None*) – The axes to invert. If `None`, the current axes are used (default).
- **dir** (*{"x", "y", "xy"}*) – The axis to invert. Default is `"y"`.

`echofilter.plotting.`**plot_indicator_hatch**(*indicator*, *xx=None*, *ymin=None*, *ymax=None*, *hatch='//'*, *color='k'*)

Plot a hatch across indicated segments along the x-axis of a plot.

Parameters

- **indicator** (*numpy.ndarray vector*) – Whether to include or exclude each column along the x-axis. Included columns are indicated with non-zero values.
- **xx** (*numpy.ndarray vector, optional*) – Values taken by indicator along the x-axis. If `None` (default), the indices of `indicator` are used: `arange(len(indicator))`.
- **ymin** (*float, optional*) – The lower y-value of the extent of the hatching. If `None` (default), the minimum y-value of the current axes is used.
- **ymax** (*float, optional*) – The upper y-value of the extent of the hatching. If `None` (default), the maximum y-value of the current axes is used.
- **hatch** (*str, optional*) – Hatching pattern to use. Default is `"//"`.

- **color** (*color, optional*) – Color of the hatching pattern. Default is black.

echofilter.plotting.**plot_mask_hatch**(**args*, *hatch='//'*, *color='k'*, *border=False*)

Plot hatching according to a mask shape.

Parameters

- **X** (*array-like, optional*) – The coordinates of the values in Z.

  X and Y must both be 2-D with the same shape as `Z` (e.g. created via numpy.meshgrid), or they must both be 1-D such that `len(X) == M` is the number of columns in `Z` and `len(Y) == N` is the number of rows in `Z`.

  If not given, they are assumed to be integer indices, i.e. `X = range(M)`, `Y = range(N)`.

- **Y** (*array-like, optional*) – The coordinates of the values in Z.

  X and Y must both be 2-D with the same shape as `Z` (e.g. created via numpy.meshgrid), or they must both be 1-D such that `len(X) == M` is the number of columns in `Z` and `len(Y) == N` is the number of rows in `Z`.

  If not given, they are assumed to be integer indices, i.e. `X = range(M)`, `Y = range(N)`.

- **Z** (*array-like(N, M)*) – Indicator for which locations should be hatched. If `Z` is not a boolean array, any location where `Z > 0` will be hatched.

- **hatch** (*str, optional*) – The hatching pattern to apply. Default is "//".

- **color** (*color, optional*) – The color of the hatch. Default is black.

- **border** (*bool, optional*) – Whether to include border around hatch. Default is `False`.

echofilter.plotting.**plot_transect**(*transect*, *signal_type=None*, *x_scale='index'*, *show_regions=True*, *turbulence_color='#a6cee3'*, *bottom_color='#b2df8a'*, *surface_color='#4ba82a'*, *passive_color=[0.4, 0.4, 0.4]*, *removed_color=None*, *linewidth=1*, *cmap=None*)

Plot a transect.

Parameters

- **transect** (*dict*) – Transect values.

- **signal_type** (*str, optional*) – The signal to plot as a heatmap. Default is `"Sv"` if present, or "signals" if not. If this is `"Sv_masked"`, the mask (given by `transect["mask"]`) is used to mask `transect["Sv"]` before plotting.

- **x_scale** (*{"index", "timestamp" "time"}, optional*) – Scaling for x-axis. If `"timestamp"`, the number of seconds since the Unix epoch is shown; if `"time"`, the amount of time in seconds since the start of the transect is shown. Default is `"index"`.

- **show_regions** (*bool, optional*) – Whether to show segments of data maked as removed or passive with hatching. Passive data is shown with `"/"` oriented lines, other removed timestamps with `"\"` oriented lines. Default is `True`.

- **turbulence_color** (*color, optional*) – Color of turbulence line. Default is `"#a6cee3"`.

- **bottom_color** (*color, optional*) – Color of bottom line. Default is `"#b2df8a"`.

- **surface_color** (*color, optional*) – Color of surface line. Default is `"#d68ade"`.

- **passive_color** (*color, optional*) – Color of passive segment hatching. Default is *[.4, .4, .4]*.

- **removed_color** (*color, optional*) – Color of removed segment hatching. Default is "r" if cmap is "viridis", and "b" otherwise.
- **linewidth** (*int*) – Width of lines. Default is 2.
- **cmap** (*str, optional*) – Name of a registered matplotlib colormap. If None (default), the current default colormap is used.

echofilter.plotting.**plot_transect_predictions**(*transect*, *prediction*, *linewidth=1*, *cmap=None*)

Plot the generated output for a transect against its ground truth data.

- Ground truth data is shown in black, predictions in white.
- Passive regions are hatched in / direction for ground truth, for prediciton.
- Removed regions are hatched in direction for ground truth, / for prediction.

Parameters

- **transect** (*dict*) – Ground truth data for the transect.
- **prediction** (*dict*) – Predictions for the transect.
- **linewidth** (*int*) – Width of lines. Default is 2.
- **cmap** (*str, optional*) – Name of a registered matplotlib colormap. If None (default), the current default colormap is used.

3.1.8 echofilter.train module

Model training routine.

echofilter.train.**build_dataset**(*dataset_name*, *data_dir*, *sample_shape*, *train_partition=None*, *val_partition=None*, *crop_depth=None*, *random_crop_args=None*)

Construct a pytorch Dataset.

Parameters

- **dataset_name** (*str*) – Name of the dataset. This can optionally be a list of multiple datasets joined with "+".
- **data_dir** (*str*) – Path to root data directory, containing the dataset.
- **sample_shape** (*iterable of length 2*) – The shape which will be used for training.
- **train_partition** (*str, optional*) – Name of the partition to use for training. Can optionally be a list of multiple partitions joined with "+". Default is "train" (except for stationary2 where it is mixed).
- **val_partition** (*str, optional*) – Name of the partition to use for validation. Can optionally be a list of multiple partitions joined with "+". Default is "validate" (except for stationary2 where it is mixed).
- **crop_depth** (*float or None, optional*) – Depth at which to crop samples. Default is None.
- **random_crop_args** (*dict, optional*) – Arguments to control the random crop used during training. Default is an empty dict, which uses the default arguments of :class`echofilter.data.transforms.RandomCropDepth`.

Returns

- **dataset_train** (*echofilter.data.dataset.TransectDataset*) – Dataset of training samples.

- **dataset_val** (*echofilter.data.dataset.TransectDataset*) – Dataset of validation samples.
- **dataset_augval** (*echofilter.data.dataset.TransectDataset*) – Dataset of validation samples, appyling the training augmentation stack.

echofilter.train.**generate_from_file**(*fname*, **args*, ***kwargs*)

Generate an output for a sample transect, specified by its file path.

echofilter.train.**generate_from_shards**(*fname*, **args*, ***kwargs*)

Generate an output for a sample transect, specified by a path to sharded data.

echofilter.train.**generate_from_transect**(*model*, *transect*, *sample_shape*, *device*, *dtype=torch.float32*)

Generate an output for a sample transect, .

echofilter.train.**meters_to_csv**(*meters*, *is_best*, *dirname='.'*, *filename='meters.csv'*)

Export performance metrics to CSV format.

Parameters

- **meters** (*dict of dict*) – Collection of output meters, as a nested dictionary.
- **is_best** (*bool*) – Whether this model state is the best so far. If `True`, the CSV file will be copied to `"model_best.meters.csv"`.
- **dirname** (*str, optional*) – Path to directory in which the checkpoint will be saved. Default is `"."` (current directory of the executed script).
- **filename** (*str, optional*) – Format for the output file. Default is `"meters.csv"`.

echofilter.train.**save_checkpoint**(*state*, *is_best*, *dirname='.'*, *fname_fmt='checkpoint{}.pt'*, *dup=None*)

Save a model checkpoint, using `torch.save()`.

Parameters

- **state** (*dict*) – Model checkpoint state to record.
- **is_best** (*bool*) – Whether this model state is the best so far. If `True`, the best checkpoint (by default named `"checkpoint_best.pt"`) will be overwritten with this `state`.
- **dirname** (*str, optional*) – Path to directory in which the checkpoint will be saved. Default is `"."` (current directory of the executed script).
- **fname_fmt** (*str, optional*) – Format for the file name(s) of the saved checkpoint(s). Must include one string argument output. Default is `"checkpoint{}.pt"`.
- **dup** (*str or None*) – If this is not `None`, a duplicate copy of the checkpoint is recorded in accordance with `fname_fmt`. By default the duplicate output file name will be styled as `"checkpoint_<dup>.pt"`.

echofilter.train.**train**(*data_dir='/data/dsforce/surveyExports'*, *dataset_name='mobile'*, *train_partition=None*, *val_partition=None*, *sample_shape=(128, 512)*, *crop_depth=None*, *resume=''*, *restart=''*, *log_name=None*, *log_name_append=None*, *conditional=False*, *n_block=6*, *latent_channels=32*, *expansion_factor=1*, *expand_only_on_down=False*, *blocks_per_downsample=(2, 1)*, *blocks_before_first_downsample=(2, 1)*, *always_include_skip_connection=True*, *deepest_inner='horizontal_block'*, *intrablock_expansion=6*, *se_reduction=4*, *downsampling_modes='max'*, *upsampling_modes='bilinear'*, *depthwise_separable_conv=True*, *residual=True*, *actfn='InplaceReLU'*, *kernel_size=5*, *use_mixed_precision=None*, *amp_opt='O1'*, *device='cuda'*, *multigpu=False*, *n_worker=8*, *batch_size=16*, *stratify=True*, *n_epoch=20*, *seed=None*, *print_freq=50*, *optimizer='adam'*, *schedule='constant'*, *lr=0.1*, *momentum=0.9*, *base_momentum=None*, *weight_decay=1e-05*, *warmup_pct=0.2*, *warmdown_pct=0.7*, *anneal_strategy='cos'*, *overall_loss_weight=0.0*)

Train a model.

echofilter.train.**train_epoch**(*loader*, *model*, *criterion*, *optimizer*, *device*, *epoch*, *dtype=torch.float32*, *print_freq=10*, *schedule_data=None*, *use_mixed_precision=False*, *continue_through_error=True*)

Train a model through a single epoch of the dataset.

Parameters

- **loader** (*iterable, torch.utils.data.DataLoader*) – Dataloader.
- **model** (*callable, echofilter.nn.wrapper.Echofilter*) – Model.
- **criterion** (*callable, torch.nn.modules.loss._Loss*) – Loss function.
- **device** (*str or torch.device*) – Which device the data should be loaded onto.
- **epoch** (*int*) – Which epoch is being performed.
- **dtype** (*str or torch.dtype*) – Datatype which which the data should be loaded.
- **print_freq** (*int, optional*) – Number of batches between reporting progress. Default is `10`.
- **schedule_data** (*dict or None*) – If a learning rate schedule is being used, this may be passed as a dictionary with the key `"scheduler"` mapping to the learning rate schedule as a callable.
- **use_mixed_precision** (*bool*) – Whether to use `apex.amp.scale_loss()` to automatically scale the loss. Default is `False`.
- **continue_through_error** (*bool*) – Whether to catch errors within an individual batch, ignore them and continue running training on the rest of the batches. If there are five or more errors while processing the batch, training will halt regardless of `continue_through_error`. Default is `True`.

Returns

- **average_loss** (*float*) – Average loss as given by criterion (weighted equally for each sample in `loader`).
- **meters** (*dict of dict*) – Each key is a strata of the model output, each mapping to a their own dictionary of evaluation criterions: "Accuracy", "Precision", "Recall", "F1 Score", "Jaccard".
- **examples** (*tuple of torch.Tensor*) – Tuple of *(example_input, example_data, example_output)*.
- **timing** (*tuple of floats*) – Tuple of *(batch_time, data_time)*.

echofilter.train.**validate**(*loader*, *model*, *criterion*, *device*, *dtype=torch.float32*, *print_freq=10*, *prefix='Test'*, *num_examples=32*)

Validate the model's performance on the validation partition.

Parameters

- **loader** (*iterable, torch.utils.data.DataLoader*) – Dataloader.
- **model** (*callable, echofilter.nn.wrapper.Echofilter*) – Model.
- **criterion** (*callable, torch.nn.modules.loss._Loss*) – Loss function.
- **device** (*str or torch.device*) – Which device the data should be loaded onto.

- **dtype** (*str or torch.dtype*) – Datatype which which the data should be loaded.
- **print_freq** (*int, optional*) – Number of batches between reporting progress. Default is 10.
- **prefix** (*str, optional*) – Prefix string to prepend to progress meter names. Default is "Test".
- **num_examples** (*int, optional*) – Number of example inputs to return. Default is 32.

Returns

- **average_loss** (*float*) – Average loss as given by criterion (weighted equally for each sample in loader).
- **meters** (*dict of dict*) – Each key is a strata of the model output, each mapping to a their own dictionary of evaluation criterions: "Accuracy", "Precision", "Recall", "F1 Score", "Jaccard".
- **examples** (*tuple of torch.Tensor*) – Tuple of *(example_input, example_data, example_output)*.

3.1.9 echofilter.utils module

General utility functions.

echofilter.utils.**first_nonzero**(*arr*, *axis=- 1*, *invalid_val=- 1*)

Find the index of the first non-zero element in an array.

Parameters

- **arr** (*numpy.ndarray*) – Array to search.
- **axis** (*int, optional*) – Axis along which to search for a non-zero element. Default is -1.
- **invalid_val** (*any, optional*) – Value to return if all elements are zero. Default is -1.

echofilter.utils.**get_indicator_onoffsets**(*indicator*)

Find the onsets and offsets of nonzero entries in an indicator.

Parameters **indicator** (*1d numpy.ndarray*) – Input vector, which is sometimes zero and sometimes nonzero.

Returns

- **onsets** (*list*) – Onset indices, where each entry is the start of a sequence of nonzero values in the input indicator.
- **offsets** (*list*) – Offset indices, where each entry is the last in a sequence of nonzero values in the input indicator, such that indicator[onsets[i] : offsets[i] + 1] != 0.

echofilter.utils.**last_nonzero**(*arr*, *axis=- 1*, *invalid_val=- 1*)

Find the index of the last non-zero element in an array.

Parameters

- **arr** (*numpy.ndarray*) – Array to search.
- **axis** (*int, optional*) – Axis along which to search for a non-zero element. Default is -1.
- **invalid_val** (*any, optional*) – Value to return if all elements are zero. Default is -1.

echofilter.utils.**mode**(*a*, *axis=None*, *keepdims=False*, ***kwargs*)

Return an array of the modal (most common) value in the passed array.

If there is more than one such value, only the smallest is returned.

Parameters

- **a** (*array_like*) – n-dimensional array of which to find mode(s).
- **axis** (*int or None, optional*) – Axis or axes along which the mode is computed. The default, `axis=None`, will sum all of the elements of the input array. If axis is negative it counts from the last to the first axis.
- **keepdims** (*bool, optional*) – If this is set to `True`, the axes which are reduced are left in the result as dimensions with size one. With this option, the result will broadcast correctly against the input array. Default is `False`.
- ****kwargs** – Additional arguments as per `scipy.stats.mode()`.

Returns **mode_along_axis** – An array with the same shape as `a`, with the specified axis removed. If `keepdims=True` and either `a` is a 0-d array or `axis` is None, a scalar is returned.

Return type numpy.ndarray

See also:

`scipy.stats.mode`

CHAPTER

FOUR

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.

4.1 Version 1.2.0

Release date: 2022-11-18. Full commit changelog.

4.1.1 Changed

Inference

- The checkpoints have been renamed to a simpler structure: `echofilter-v{MAJOR}_{ORIENTATION}_ep{EPOCHS}`. For example, the current default checkpoint is now named as `"echofilter-v1_bifacing_700ep"`. This is in keeping with the format used in the manuscript. The previous names for the checkpoints will still work (as an alias to the new names). (#330)
- The ordering of the colour palette listing produced by `--show-colors` has changed. By default, the colours are now ordered by hue (subsorted by value) instead of sorted alphabetically. The previous default behaviour can be obtained by using the new "alphabetic" argument: `--show-colors "alphabetic"`. (#322)

4.1.2 Fixed

Inference

- Fix a bug with Echoview interaction which was causing the code to fail when overwriting an existing line in the EV file. (#332)
- Fix hiding the Echoview window while echofilter is running and interacting with Echoview. This bug was first seen when using Echoview 13, and was not affecting interactions with Echoview 10 or 11. (#328)

Metadata

- Add missing `requests` package to requirements. (#320)

4.1.3 Added

Inference

- Add API option `--show-cache-dir`, which shows the path to the directory in which downloaded checkpoints will be cached. (#321)
- Add `"alphabetic"` and `"full-alphabetic"` options to `--show-colors` which let the colours be sorted by alphabetically, instead of by hue. (#322)

4.2 Version 1.1.1

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

4.2.1 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)

4.3 Version 1.1.0

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

4.3.1 Changed

Inference

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

4.3.2 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)

4.3.3 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)

4.4 Version 1.0.3

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

This minor patch fix addresses package metadata.

4.4.1 Fixed

Metadata

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

4.5 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.

4.5.1 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)

4.6 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.

4.6.1 Changed

Requirements

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

4.6.2 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)

4.7 Version 1.0.0

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

This is the first major release of echofilter.

4.7.1 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)

4.8 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.

4.8.1 Fixed

Inference

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

4.9 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.

4.9.1 Fixed

Inference

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

4.10 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.

4.10.1 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)

4.10.2 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)

4.10.3 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)

4.11 Version 1.0.0b4

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

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

4.11.1 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)

4.11.2 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)

4.11.3 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` 1 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)

4.12 Version 1.0.0b3

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

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

4.12.1 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)

4.12.2 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)

4.12.3 Added

Inference

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

4.13 Version 1.0.0b2

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

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

4.13.1 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)

4.13.2 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)

4.13.3 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)

4.14 Version 1.0.0b1

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

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

4.14.1 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)

4.14.2 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)

4.14.3 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)

4.15 Version 0.1.4

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

4.15.1 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)

4.16 Version 0.1.3

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

4.16.1 Fixed

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

4.16.2 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)

4.17 Version 0.1.2

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

4.17.1 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)

4.17.2 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)

4.18 Version 0.1.1

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

4.18.1 Fixed

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

4.19 Version 0.1.0

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

CHAPTER

FIVE

MODULE INDEX

CHAPTER

SIX

INDEX

PYTHON MODULE INDEX

e

INDEX

F

G

H

N

O

P

R

S

T

U

V

W