Operationalizing and Testing Machine Learning Models for Acoustic Target Classification

Nils Olav Handegard, Silje Smith-Johnsen, Arne Johannes Holmin, Cristian Muños Mas
Institute of Marine Research
Bergen, Norway
email: nilsolav@hi.no; silje.smith-johnsen@hi.no; arnejh@hi.no; cristian.munoz.mas@hi.no

Ingrid Utseth
Norwegian Computing Centre
Oslo, Norway
email: utseth@nr.no

Daniel Dondorp
Sopra Steria
Bergen, Norway
email: Daniel.Dondorp@hi.no

***Abstract*—Acoustic trawl surveys use echosounders to collect acoustic backscatter, which is categorized and combined with trawl samples to generate abundance indices for fisheries assessment models. Machine learning models are being developed to automate the acoustic target classification step, and it is necessary to evaluate their performance in comparison to manual processes and earlier model versions. The data processing pipeline consists of several stages, utilizing various software, versions, and libraries. Docker containers provide flexibility, especially for advanced pipeline steps. Some steps use Python virtual environments. Clearly defining data models between processing steps is necessary and adopting community standards where applicable is recommended. We have set up a system to combine and run the pipeline steps, and we have used it to compare different ML models. We are currently working to further streamline the process.**

Keywords- acoustic trawl surveys; machine learning; micro services; containerisation; data linage; MLOps; CI/CD.

I. INTRODUCTION

Data from acoustic trawl surveys are an important source of information for fisheries assessments [1]. Data are collected with echosounders mounted on research vessels and individual fish are sampled, usually from trawls. The echosounder data are calibrated and used to enumerate the abundance of fish, whereas the physical samples are used for ground truthing the acoustic registrations and to obtain age and other biological parameters used by the assessment models. More recently, autonomous platforms are being used to augment the acoustic data collection [2].

The process of allocating the acoustic backscatter to species, the Acoustic Target Classification (ATC) step, is in most cases a manual process. There are methods to automate ATC [3], and both the shape of the marks on the echogram [4], as well as the response between different echosounder frequencies [5][6] are being used.

Machine learning methods are well suited for this process, and convolutional neural networks have been developed for ATC [7][8]. Training supervised machine learning models requires training data, and the historical records of manually annotated acoustic data is a rich source of information but requires careful preparations before use. The data sets are highly imbalanced since most of the echograms contain no fish, and different methods for training these methods have been developed [9]. Methods that require less labels during training are also being developed, and both semi supervised and fully supervised models [10][11] are available [12].

To estimate the effect of the different models on the survey results, the different models are run on the same data set and are further processed using the standard data processing pipeline for the survey. This way we can evaluate the effect of the different models and compare them to the official estimates [13].

To efficiently run the data processing pipeline with various configurations and a range of different software and software environments, a digital infrastructure is needed. The objective is to have a pipeline that is modular, has sufficient data provenance, i.e., track the dataset and model used in the different runs, and has the flexibility to be run with different settings and input data to evaluate different methods. The objective of this paper is to describe the current approach, and the plans for extending and operationalizing the pipeline. We start by describing the different steps in the data processing pipeline, followed by how we operationalize it. Finally, we describe the plans and future developments.

II. THE DATA PROCESSING PIPELINE

The different steps in the pipeline (Figure 1) have been described earlier, c.f. the supplementary material of [13], and are briefly summarized here. The different steps have their own git repositories and are versioned and maintained separately.

 ISBN: 978-1-68558-329-3

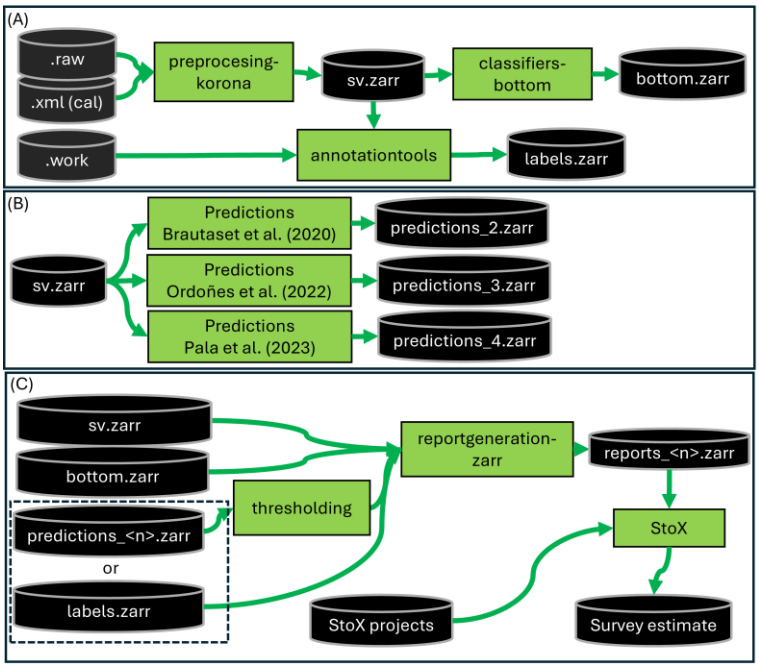

Figure 1. The data procesing pipeline for automated acoustic target classificaiton. Figure reprinted from [13] under CC-BY 4.0. (A) The preprosessing step converting the the raw data and annotations and creating a mask that removes data below the sea bed. (B) The Machine learning models (C) The survey estimation step.

A. Data preprocessing

The data are preprocessed (Figure 1A) from the native data format from the echosounder manufacturer (Kongsberg Discovery, Horten, Norway) to a self-documented gridded format (Figure 2). The data is gridded to the same grid as the primary 38 kHz frequency. We have used the Zarr store [14], which is a chunked cloud friendly format similar to NetCDF. The Python Xarray package [15] can work seamlessly between NetCDF and Zarr, and offer a convenient method for working with the data. The data processing is done using Korona (Marec, Bergen Norway) combined with the python libraries Zarr and Xarray, and results in a three-dimensional gridded data structure with dimensions “range”, “time” and “acoustic frequency”.

The labels are read from the Large Scale Survey System (LSSS) [16] “work” files (Figure 3). LSSS is the standard tool at the Institute of Marine Research (IMR) for annotating the acoustic data. The annotations are converted to the same grid as the echosounder files resulting in a gridded data structure with dimensions “range”, “time” and “acoustic category”, where the “range” and “time” dimensions are identical to the preprocessed echosounder data. A bottom detection algorithm is used to mask the data below the seabed.

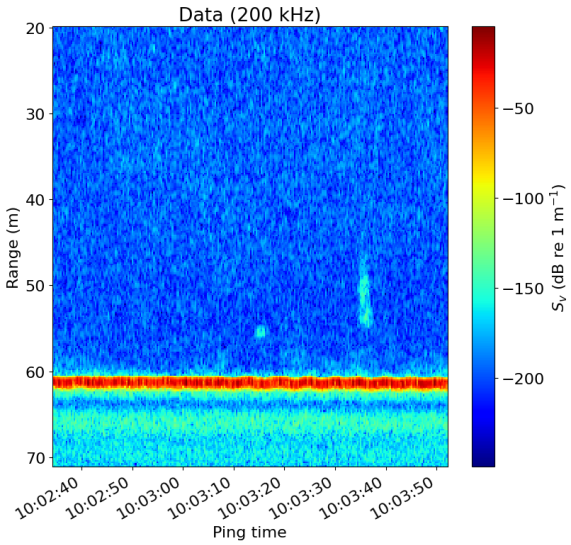

Figure 2. The 200 kHz channel slice in range and time from the regridded data structure.

 ISBN: 978-1-68558-329-3

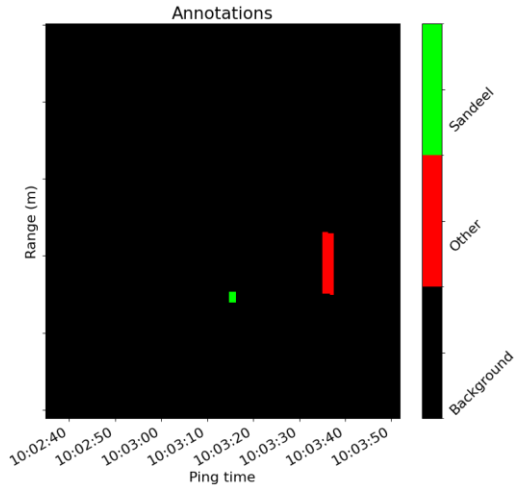

Figure 3. The annotations converted from the LSSS system.

B. *Running ML model predictions*

Several machine learning models are developed, mostly using PyTorch [17], and trained on segments of processed acoustic data to detect and classify acoustic categories and background. To prevent empty water segments from dominating the training dataset, an effective sampling strategy must be used during training [7][9].

The ML models were trained on a NVIDIA GTX 1080 Ti GPU using the PyTorch framework. The inference is run on a Dell PowerEdge R730 server with 80x Intel(R) Xeon(R) CPU E5-2698, 768GB memory, two Tesla P100-PCIE-12GB graphical processing units running CUDA version 13.0 and rootless docker.

The models return predictions of size 256 x 256 samples, and the data segments are extracted using a sliding window with 40 samples overlap to mitigate edge effects. SoftMax is applied to the predictions to obtain normalized class scores for each sample, and the predictions are subsequently stitched together [7] to a data structure that is the same as the annotations, except that the values are SoftMax predictions instead of binary annotation data.

C. *Model thresholding and abundance estimates*

To convert the SoftMax predictions to binary masks identical to the annotations, a thresholding approach is applied, c.f. [13] for details. The predictions can then be used interchangeably with the annotations to calculate the integrated backscatter (nautical area scattering coefficient, NASC) in a 0.1 nautical mile by 10 m depth channels.

The survey estimates are calculated using the StoX software [18], where the NASC values from the manual annotation process are replaced with the ML generated reports to evaluate the effect on the survey estimate. This has allowed us to compare the performance of different ML models in relation to the survey estimate. The general impression is that which training years are used are more important than the various model and training approaches, c.f. [13] for more details.

III. OPERATIONALIZING THE PIPELINE

Each step in the pipeline is an individual step connected by data structures shared by the connected components. The individual steps use various software and tools with different programming languages and versions of libraries. Maintaining computational environments for all the individual steps can be challenging to set up in a conventional compute environment, especially when the pipelines are composed of components across different disciplines or organizational units.

In our case, each individual step in the pipeline is currently handled somewhat different depending on its maturity. The preprocessing steps (Figure 1A) are embedded in individual Docker containers, and the Docker image is defined in a versioned Dockerfile that describes the environment. The machine learning models (Figure 1B) and the thresholding and report generation steps (Figure 1C) are run in Python using virtual environments (venv), and the environments are specified in a requirements file in individual git repositories. A working python environment is needed to run these. The survey estimate is packaged in a Docker container running R and StoX [18] and are specified in a Dockerfile.

In cases where a step is fully coded in python and uses a set of python libraries, the use of Python virtual environments works well. However, different environments are used for different steps, and the approach is restricted to python only. Container technologies, e.g., Docker, to package the individual steps are a powerful technology that allows us to set up the compute environment with a range of different software and programming languages. It also allows the different steps in the pipeline to be upgraded and rolled out independently of each other. This is highly flexible and allows a pipeline to be coded up using any programming language or software as long as it can be run in a container, i.e., headless and scripted.

After the individual steps are coded and set up in separate containers or environments, they need to be orchestrated. This must also include documentation of the data provenance, i.e., which versions of the various data and algorithms were used to generate the result. Except for the preprocessing step, we are currently manually running the steps using shell scripts and ad hoc provenance.

We have deployed an automated continuous integration and continuous delivery (CI/CD) pipeline using a GitLab instance running on IMRs servers. This approach improves efficiency and reproducibility and keep track of version control throughout the preprocessing stages. When a new version of the preprocessor is made, the data set is updated. We do not keep earlier versions of the data sets, but we keep track of which version of the preprocessor that was used. The pipeline facilitates the orchestration of the three preprocessing tasks (Figure 1A), where each step is maintained in separate Git repositories and connected to a continuous integration and continuous delivery (CI/CD) pipeline. Each time modifications are made to the

 ISBN: 978-1-68558-329-3

repositories, it triggers a job within the pipeline, and this is manually initiated to execute the preprocessing workflow.

The current CI/CD pipeline is effective in converting raw data to preprocessed data. However, it remains static in configuring experiments and managing different combinations of data and processing steps. It lacks flexibility for efficiently modifying or replacing components within the processing chain and for handling data provenance.

IV. CONCLUSION AND FUTURE WORK

Setting up the pipelines is a work in progress, and the maturity of the different steps varies. The preprocessing pipeline is at a more advanced stage, and we use that to test various approaches for orchestration.

Regardless of technology, the data model that sits between the steps must be agreed upon and specified. This can be a challenging task unless the pipeline is well established. Community data conventions and open data formats should be used where possible. Since the pipeline is scripted, reprocessing the data is straight forward, but downstream code must be adjusted to accommodate changes in data formats.

The containerization of the processing steps is particularly useful since it allows separation of concerns. Different groups and have full flexibility to use and maintain their part of the pipeline as they prefer, but it is important to define the steps to reflect the organizational units.

The CI/CD is still in the works. Currently we have a system for the preprocessing step where the data is updated when a new version of the preprocessor is available. We are currently working on methods to more flexibly set up data processing pipelines where we can use different models or preprocessed data to generate the results.

In conclusion, we have a clearly defined data lineage for the problem. The containerization approach is particularly useful and helps separation of concerns. The data models need to be upgraded to community standards where applicable, and we need a better orchestration of the pipeline to set up and track experiments and test the effects on new algorithms for the data processing.

ACKNOWLEDGMENT

This project is supported by the Research Council of Norway (CRIMAC grant no. 309512).

REFERENCES

[1] D. MacLennan and E. J. Simmonds, Fisheries Acoustics, 2nd ed. in Fish and aquatic resources series 10. London: Chapman & Hall, 2005.

[2] N. O. Handegard et al., “Uncrewed surface vehicles (USVs) as platforms for fisheries and plankton acoustics,” ICES J. Mar. Sci., pp. 1712–1723,, 2024, doi: 10.1093/icesjms/fsae130.

[3] R. Korneliussen, Ed., “Acoustic target classification,” ICES Coop. Res. Rep., vol. 344, pp. 104, 2018, doi: 10.17895/ices.pub.4567.

[4] D. G. Reid, “Report on Echo Trace Classification,” International Council for the Exploration of the Sea, Copenhagen, Denmark, ICES Cooperative Research Report No. 238, pp. 107, 2000. doi: 10.17895/ices.pub.5371

[5] R. J. Korneliussen and E. Ona, “Synthetic echograms generated from the relative frequency response,” ICES J. Mar. Sci. J. Cons., vol. 60, no. 3, pp. 636–640, 2003, doi: 10.1016/S1054-3139(03)00035-3.

[6] R. J. Kloser, T. Ryan, P. Sakov, A. Williams, and J. A. Koslow, “Species identification in deep water using multiple acoustic frequencies,” Can. J. Fish. Aquat. Sci., vol. 59, pp. 1065–1077, 2002, doi: 10.1139/f02-076.

[7] O. Brautaset et al., “Acoustic classification in multifrequency echosounder data using deep convolutional neural networks,” ICES J. Mar. Sci., vol. 77, no. 4, pp. 1391–1400, 2020, doi: 10.1093/icesjms/fsz235.

[8] A. Ordoñez, I. Utseth, O. Brautaset, R. Korneliussen, and N. O. Handegard, “Evaluation of echosounder data preparation strategies for modern machine learning models,” Fish. Res., vol. 254, pp. 106411, 2022, doi: 10.1016/j.fishres.2022.106411.

[9] A. Pala, A. Oleynik, I. Utseth, and N. O. Handegard, “Addressing class imbalance in deep learning for acoustic target classification,” ICES J. Mar. Sci., vol. 80, pp. 2530–2544, 2023, doi: 10.1093/icesjms/fsad165.

[10] C. Choi et al., “Semi-supervised target classification in multi-frequency echosounder data,” ICES J. Mar. Sci., vol. 78, no. 7, pp. 2615–2627, 2021, doi: 10.1093/icesjms/fsab140.

[11] C. Choi, M. Kampffmeyer, N. O. Handegard, A.-B. Salberg, and R. Jenssen, “Deep Semisupervised Semantic Segmentation in Multifrequency Echosounder Data,” IEEE J. Ocean. Eng., pp. 1–17, 2023, doi: 10.1109/JOE.2022.3226214.

[12] A. Pala, A. Oleynik, K. Malde, and N. O. Handegard, “Self-supervised feature learning for acoustic data analysis,” Ecol. Inform., vol. 84, pp. 102878, 2024, doi: 10.1016/j.ecoinf.2024.102878.

[13] N. O. Handegard, A. J. Holmin, A. Pala, I. Utseth, and E. Johnsen, “Integrating and assessing machine learning acoustic target classification models for fish survey estimations,” ICES J. Mar. Sci., vol. 82, no. 5, pp. 1712–1723,, 2025, doi: 10.1093/icesjms/fsaf069.

[14] Zarr.dev: Zarr core specification. Accessed: Sep. 19, 2025. [Online]. Available from: https://zarr-specs.readthedocs.io/en/latest/v3/core/index.html

[15] S. Hoyer and J. Hamman, “xarray: N-D labeled Arrays and Datasets in Python,” J. Open Res. Softw., vol. 5, no. 1, 2017, doi: 10.5334/jors.148.

[16] R. J. Korneliussen, Y. Heggelund, G. J. Macaulay, D. Patel, E. Johnsen, and I. K. Eliassen, “Acoustic identification of marine species using a feature library,” Methods Oceanogr., vol. 17, pp. 187–205,2016, doi: 10.1016/j.mio.2016.09.002.

[17] A. Paszke et al.: Automatic differentiation in PyTorch, 2017, [Online]. Available from: https://openreview.net/forum?id=BJJsrmfCZ

[18] E. Johnsen et al., “StoX: An open source software for marine survey analyses,” Methods Ecol. Evol., vol. 10, no. 9, pp. 1523–1528, 2019, doi: 10.1111/2041-210X.13250.