The Journal of Open Source Software

EchoFlow: End-to-end self-supervised sonar-image pipeline

Erling Devold [1]

1 SINTEF AS, Trondheim, Norway

DOI: 10.21105/joss.09240

Software

- Review
- Repository
- Archive

Editor: Neea Rusch

Reviewers:

- @subhk
- @nkrusch

Submitted: 10 June 2025
Published: 21 April 2026

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

EchoFlow is a three-stage, containerised workflow that converts raw Kongsberg EK80 echosounder files into human-readable echograms *and* machine-interpretable attention maps. The EK80 is the dominant wideband echosounder from Kongsberg, the leading manufacturer in fisheries acoustics; surveys with these instruments routinely produce terabytes of multi-frequency data.

1. **Conversion** – raw `.raw` pings are decoded and calibrated to volume-back-scattering strength with *PyEcholab* (Wall et al., 2018).
2. **Pre-processing** – echograms are contrast-stretched, down-sampled and tiled as PNG images.
3. **Inference** – a Vision Transformer trained with DINO (Caron et al., 2021) yields per-patch attention heat-maps that highlight fish schools, seabed returns, and other salient structures.

Each stage is encapsulated in its own Docker image and orchestrated with Docker Compose. Continuous Integration (CI) ensures that a test file always produces at least one attention map per echogram frequency. The test file can be inspected as artifacts from the CI actions.

Statement of need

Marine-acoustics researchers collect **terabytes** of multi-frequency sonar data per survey but lack an open-source tool-chain that

- converts heterogeneous raw formats,
- scales from a laptop to multi-core servers, and
- integrates state-of-the-art computer-vision models.

Today the typical workflow relies on ad-hoc scripts combining PyEcholab or Echopype for format conversion, followed by manual preprocessing and separate ML tooling—or on proprietary software such as Echoview. No single open-source pipeline bridges raw data and ML-ready outputs.

Previous work (Lee et al., 2024; Wall et al., 2018) addresses the first bullet; EchoFlow fills the remaining gap by chaining **conversion → pre-processing → self-supervised inference** in a single, reproducible workflow. This lowers the barrier for fisheries scientists, marine-robotics engineers, and citizen scientists who want modern ML without bespoke pipelines. The containerised design also facilitates deployment on shared compute environments via tools such as Singularity/Apptainer. This pipeline also serves as a foundation for incorporating modern frameworks into marine science.

EchoFlow was developed as part of the research project *Robotics underneath sub-zero waters and outer space* (RCN project 328193), where it is used to process acoustic data from

autonomous underwater vehicles for ocean biomass estimation.

Implementation and architecture

Each stage lives in its own Docker image and communicates through bind-mounted volumes (`./data/`). A Python watchdog triggers the pipeline when new `.raw` files arrive, and pre-trained DINO weights are cached on first use. Images are multi-platform (`linux/amd64`, `linux/arm64`).

Performance

Stages 1–2 (conversion and pre-processing) are CPU- and I/O-bound; their throughput scales near-linearly with `MAX_WORKERS`. Stage 3 (inference) is GPU-bound, controlled by `BATCH_SIZE`. The included benchmark script (`benchmark.py`) runs the full three-stage pipeline inside Docker on real EK80 data from the NOAA Water-Column Sonar Data archive, varying worker counts from 1 to 8. Users can reproduce these numbers with:

```
bash download_bench_data.sh   # 64 files, ~6.4 GB
docker compose build
python benchmark.py
```

Table 1 shows throughput on a 12-core Intel i7-12700 (20 threads, 32 GB RAM, NVIDIA RTX 3090) processing 64 EK80 files (6.7 GB total) from the NOAA WCSD archive. Stages 1–2 are CPU-bound and scale near-linearly with `MAX_WORKERS` until I/O saturates. Stage 3 (GPU inference at 1000×1000 px, batch size 4) takes 393 s for 320 images on a single RTX 3090. At eight workers the full pipeline processes one ~105 MB echogram in approximately 8.7 seconds end-to-end, implying roughly 23 hours per terabyte; in production the stages run concurrently via a filesystem watchdog, so wall-clock time is dominated by the slowest stage.

Table 1: End-to-end benchmark. Stages 1–2 vary with `MAX_WORKERS`; stage 3 is GPU-bound and constant.

Workers	Stage 1 (s)	Stage 2 (s)	Stage 3 (s)	Total (s)	s/file	Speedup
1	139.6	514.0	393	1046.6	16.35	1.00x
2	86.5	275.7	393	755.2	11.80	1.39x
4	56.3	156.2	393	605.5	9.46	1.73x
8	56.9	97.2	393	547.1	8.55	1.91x

Illustrative example

```
# 0 - fetch a sample EK80 file (NOAA public bucket)
BUCKET="s3://noaa-wcsd-pds/data/raw"
FILE="Bell_M._Shimada/SH2306/EK80/Hake-D20230811-T165727.raw"
aws s3 cp --no-sign-request "${BUCKET}/${FILE}" data/input

# 1 - run the full pipeline
docker compose up --build raw preprocessing infer

# 2 - open the resulting echogram and attention map
xdg-open data/preprocessing/Hake-D20230811-T165727/38000_debug.jpg
xdg-open data/inference/Hake-D20230811-T165727/70000.png
```

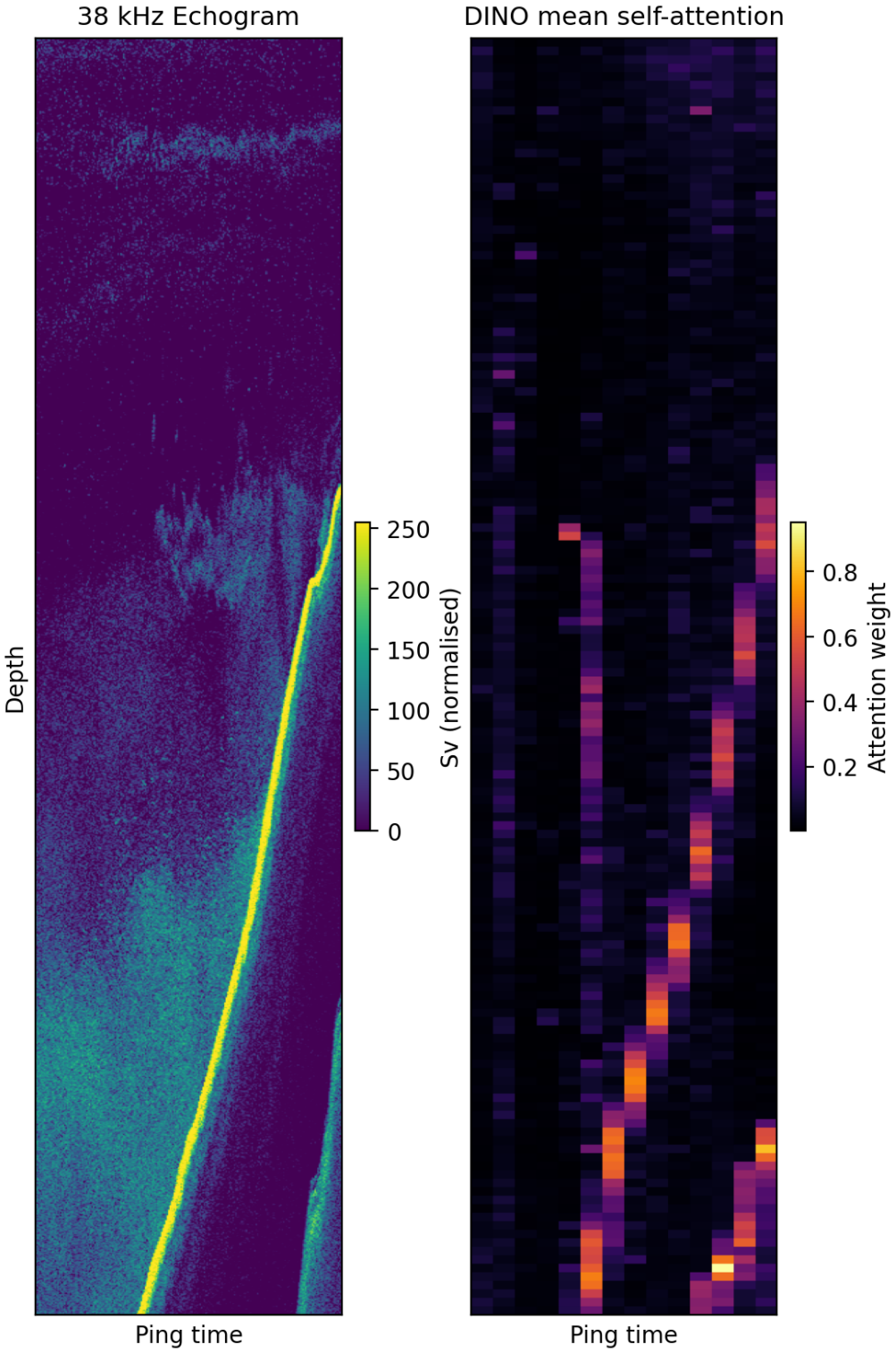

Figure 1: Preprocessed 38 kHz echogram (left) and DINO attention map at 70 kHz (right) for file Hake-D20230811-T165727.

Acknowledgements

This work was supported by the Research Council of Norway under project 328193 (*Robotics underneath sub-zero waters and outer space*, 2022–2025).

References

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging properties in self-supervised vision transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9639–9650. https://doi.org/10.1109/ICCV48922.2021.00951

Lee, W.-J., Setiawan, L., Tuguinay, C., Mayorga, E., & Staneva, V. (2024). Interoperable and scalable echosounder data processing with Echopype. *ICES Journal of Marine Science*, *81*(10), 1941–1951. https://doi.org/10.1093/icesjms/fsae133

Wall, C. C., Towler, R., Anderson, C., Cutter, R., & Jech, J. M. (2018). PyEcholab: An open-source Python-based toolkit to process and visualize echosounder data. *The Journal of the Acoustical Society of America*, *144*(3), 1778. https://doi.org/10.1121/1.5067860