# A Appendix

This appendix contains supporting figures and implementation details for reproducibility.

## A.1 Data collection set-up

Fig. 1 shows the ROV used to collect the JAMBO dataset, and Fig. 2 shows a screenshot of the labelling interface used by all annotators. The ROV's GoPro was configured to record with a frame rate of 60 FPS, a resolution of 3840x2160 pixels, and using the *linear* digital lens setting with a field of view of 92°, 61°, and 100° in the horizontal, vertical, and diagonal direction, respectively.

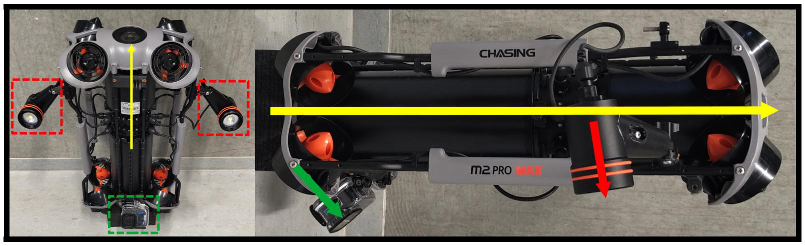
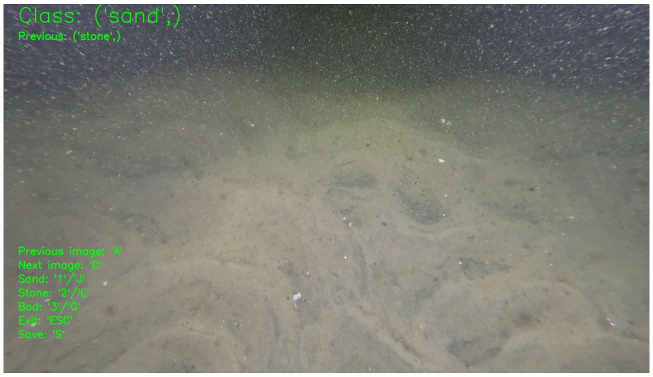


**Fig. 1:** The underwater ROV captured from two different angles. The position and orientation of the camera and light sources are highlighted by green and red markings, respectively. The yellow arrows indicate the forward orientation of the drone.



```
Class: ('sand',)
Previous: ('stone',)
```

```
Previous image: 'A'
Next image: 'D'
Sand: '1'/'J'
Stone: '2'/'L'
Bad: '3'/'G'
Exit: 'ESC'
Save: 'S'
```

**Fig. 2:** Screenshot of the labelling interface.

### A.2 Classification pipeline

Before being fed to a pre-trained feature extractor, images are resized to a resolution of 224 × 224 using bilinear interpolation. The baseline models in our experiments were implemented as follows:

- The Sup-IN21K and DINO models are taken from the HuggingFace library [3] (model cards `google/vit-base-patch16-224-in21k` and `facebook/dino-vitb16` respectively).
- The OpenCLIP and BioCLIP models are taken from the OpenCLIP library (`open_clip.create_model_and_transforms('ViT-B-16', pretrained='laion400m_e32')` and `open_clip.create_model_and_transforms('hf-hub:imageomics/bioclip')` respectively).
- For all four pre-trained backbones, a 1-dimensional feature vector of size 768 is extracted by taking the pooling output (we do not apply the projection layer in the CLIP models).
- The logistic regression classifier is trained using sklearn's L-BFGS implementation [1] with cross-entropy loss as training objective, and a maximum of 1000 iterations (similarly to [2]). Samples are inversely weighted based on class frequency to account for the heavy class imbalance in our dataset. The inverse regularization parameter $C$ is kept to its default value of 1.0 (cf. Appendix A.4).
- As an alternative to logistic regression, the k-Nearest Neighbour classifier is trained using sklearn's implementation [1] based on Euclidean distance. Features are zero-centered based on training data statistics and normalized before being fed to the classifier. The parameter $K$ is kept to its default value of 5 (cf. Appendix A.4).

### A.3 Dataset splits

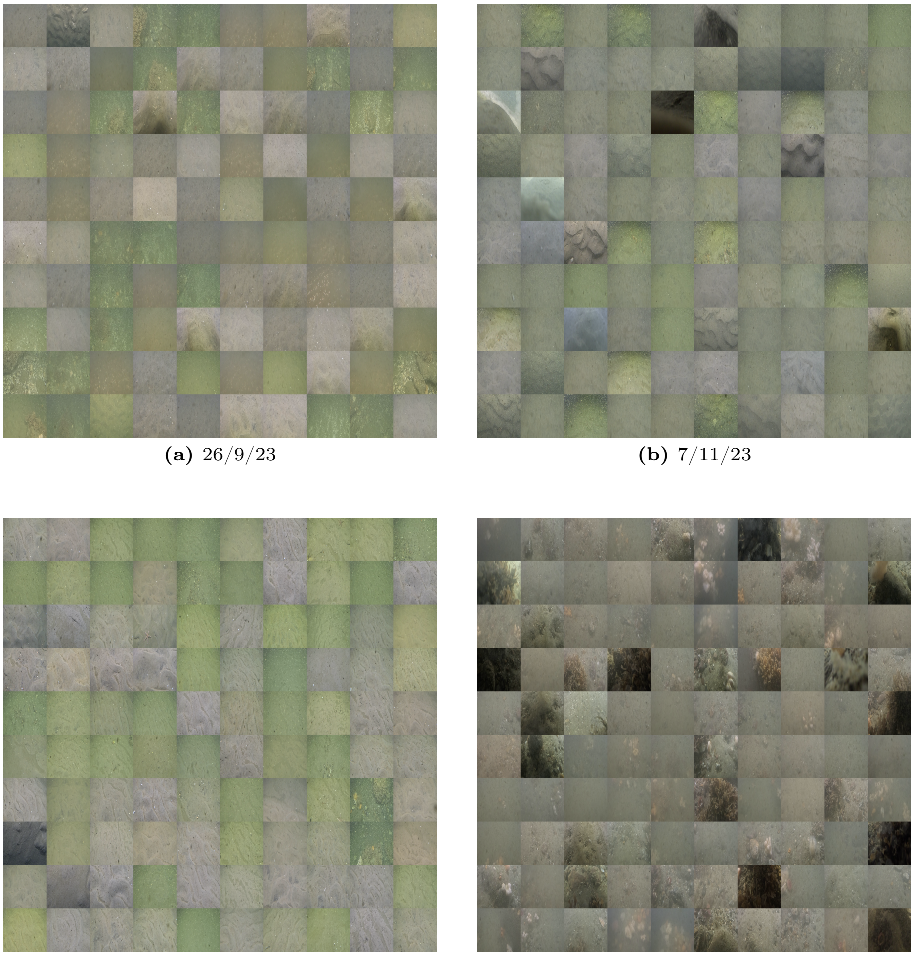
Here we describe the train/test splits used in the 2 benthic habitat classification experiments. Cross-validation is less challenging, since the test set can contain images taken at a very similar location to images in the training set. The date-based splits are designed to evaluate generalization to new locations.

**Cross-validation** - We apply random stratified cross-validation as implemented by sklearn [1] via the `StratifiedShuffleSplit` method, with 20 splits and a test size of 10%. That is, 20 different train/test splits are created randomly with 90% of images in the train set and 10% in the test set, while also preserving the original class distribution in each set. Using randomized cross-validation rather than K-fold cross-validation ensures that the minority *bad* class has at least one example per train and test test, despite there only being six images unanimously labelled as *bad* in the whole dataset. This means that some test examples are repeated across different splits.

**Date-based test splits** - As illustrated in Figure 1 (main text), videos were collected on 4 different days, covering a different area each day. To evaluate model generalization, we create three train/test splits by holding out data from a specific day (cf. Tab. 1). Images from the 8/11 is kept as training data in all three splits due to the large number of images collected on that day. Examples from each day are shown in Fig. 3.

**Table 1:** Number of dataset images for each data collection day, and the three dataset splits used to evaluate generalization.

| *date* | 26/9/23 | 7/11/23 | 8/11/23 | 10/11/23 | **total** |
|---|---|---|---|---|---|
| *num. images* | 710 | 289 | 2189 | 102 | **3290** |
| split Sep.26 | **test** | train | train | train | |
| split Nov.07 | train | **test** | train | train | |
| split Nov.10 | train | train | train | **test** | |



**(a)** 26/9/23

**(b)** 7/11/23

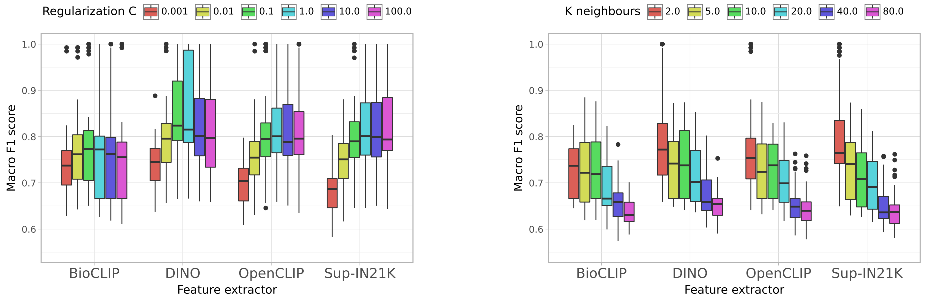
**(c)** 8/11/23

**(d)** 10/11/23

**Fig. 3:** 100 randomly selected dataset images from each data collection day, after being resized to 224x224.

## A.4 Classifier hyper-parameters

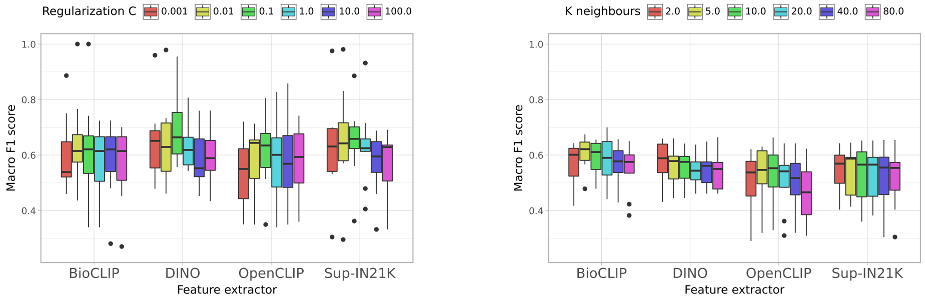
Here we zoom in to the choice of classifier hyperparameters: $C$ in the logistic regression classifier determines the inverse of the regularization strength (stronger regularization as $C$ decreases), and $K$ in the KNN classifier determines the number of neighbours used for majority voting (larger $K$ leads to smoother decision boundary). We sweep 6 values of $C$ and $K$ and record classification performance on the cross-validation splits in Fig. 4 and the date-based splits in Fig. 5.



**(a)** Logistic regression classifier

**(b)** KNN classifier

**Fig. 4:** Effect of the classifier's hyperparameter values on test set classification performance across the **20 cross-validation splits** and across the 3 supervision schemes ($20 \times 3 = 60$ points per boxplot).



**(a)** Logistic regression classifier

**(b)** KNN classifier

**Fig. 5:** Effect of the classifier's hyperparameter values on test set classification performance across the **3 date-based test splits** and across the 3 supervision schemes ($3 \times 3 = 9$ points per boxplot).

## References

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011). https://doi.org/10.48550/ARXIV.1201.0490
2. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR (18–24 Jul 2021), https://proceedings.mlr.press/v139/radford21a.html
3. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020), https://www.aclweb.org/anthology/2020.emnlp-demos.6