Onboard Image Classification of Biological Habitats Using Underwater Vehicles

Miguel Quinaz Pereira

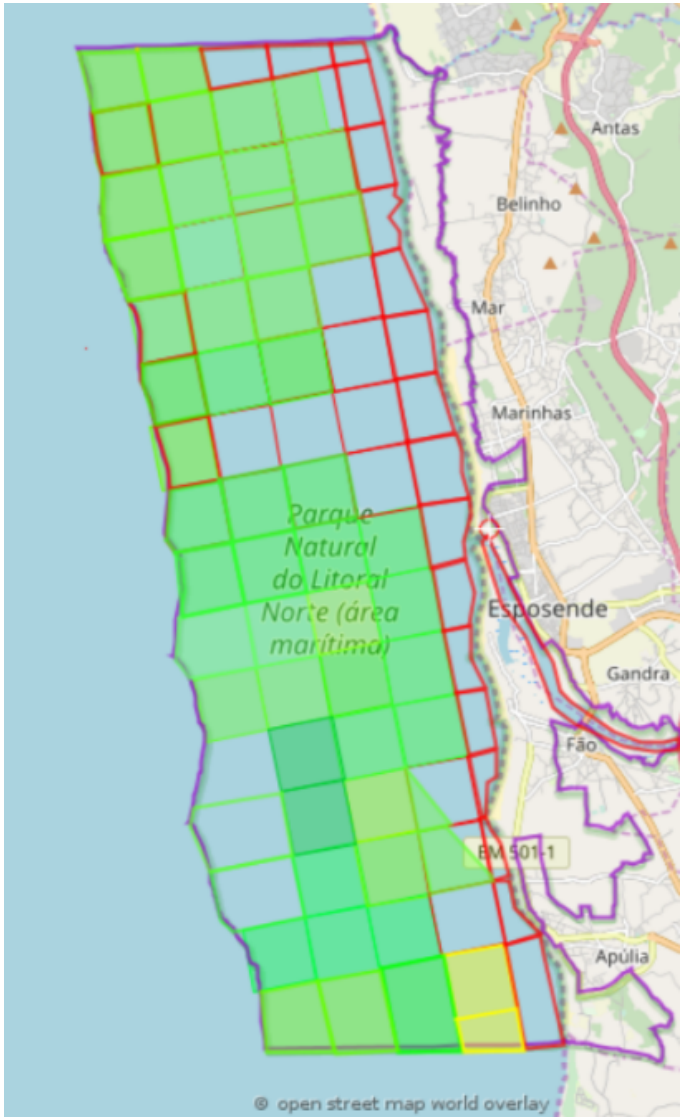
Departamento de Ciências dos Computadores

December 2, 2021

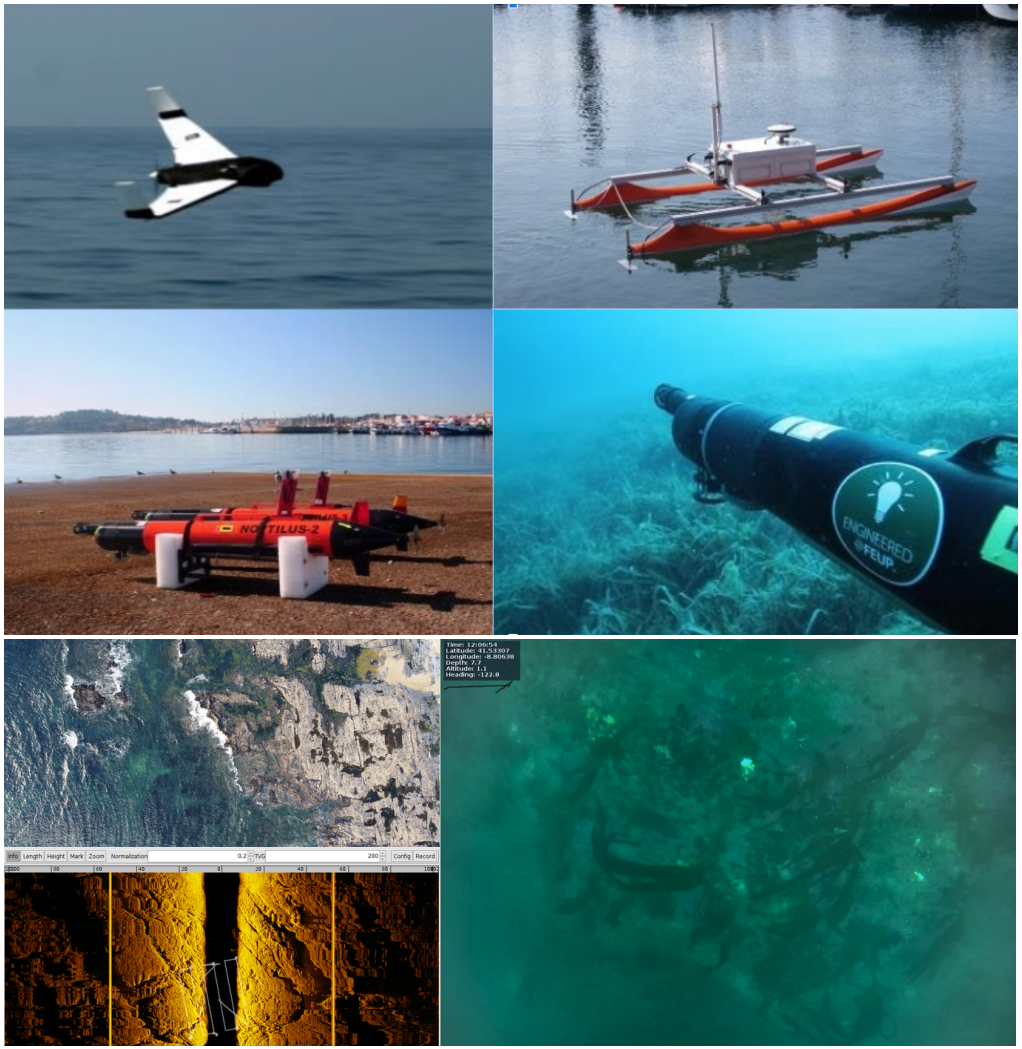
Outline

OMARE Project

- In the OMARE project, biologists classify natural habitats of Parque do Litoral Norte according to the EUNIS guidelines (80 Km^2 along the park, 70 Km^2 underwater);
- Using biologists' classifications, CNN models were developed for automated classification of habitats.

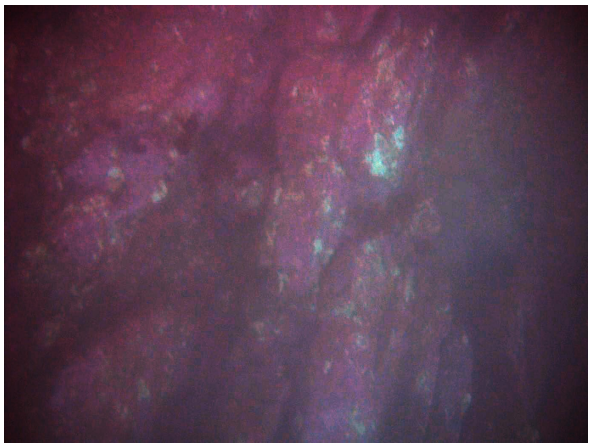
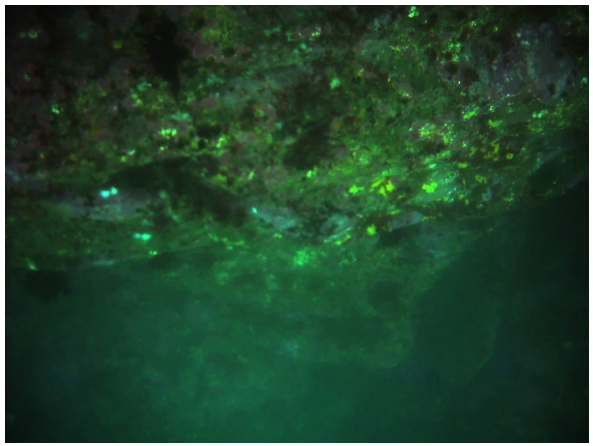
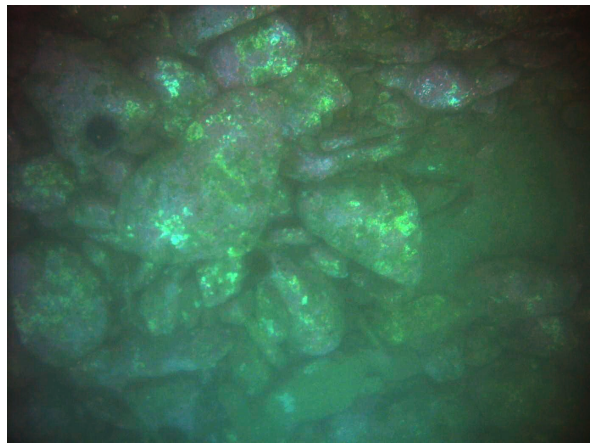


Autonomous vehicles



- LSTS owns different kinds of vehicles AUV, UAV, ROV, ASV;
- Light AUV retrieved underwater images for the used dataset;
- Other imagery like aerial or sidescan sonar can be used as well;
- LSTS made a partnership with OMARE, for the use of autonomous vehicles to aid the task of habitat mapping for monitoring purposes.

EUNIS classification standard

A1. Littoral rock and other hard substrata

A2. Littoral sediment

A3. **Infralittoral rock and other hard substrata**

A4. **Circalittoral rock and other hard substrata**

A5. **Sublittoral sediment**

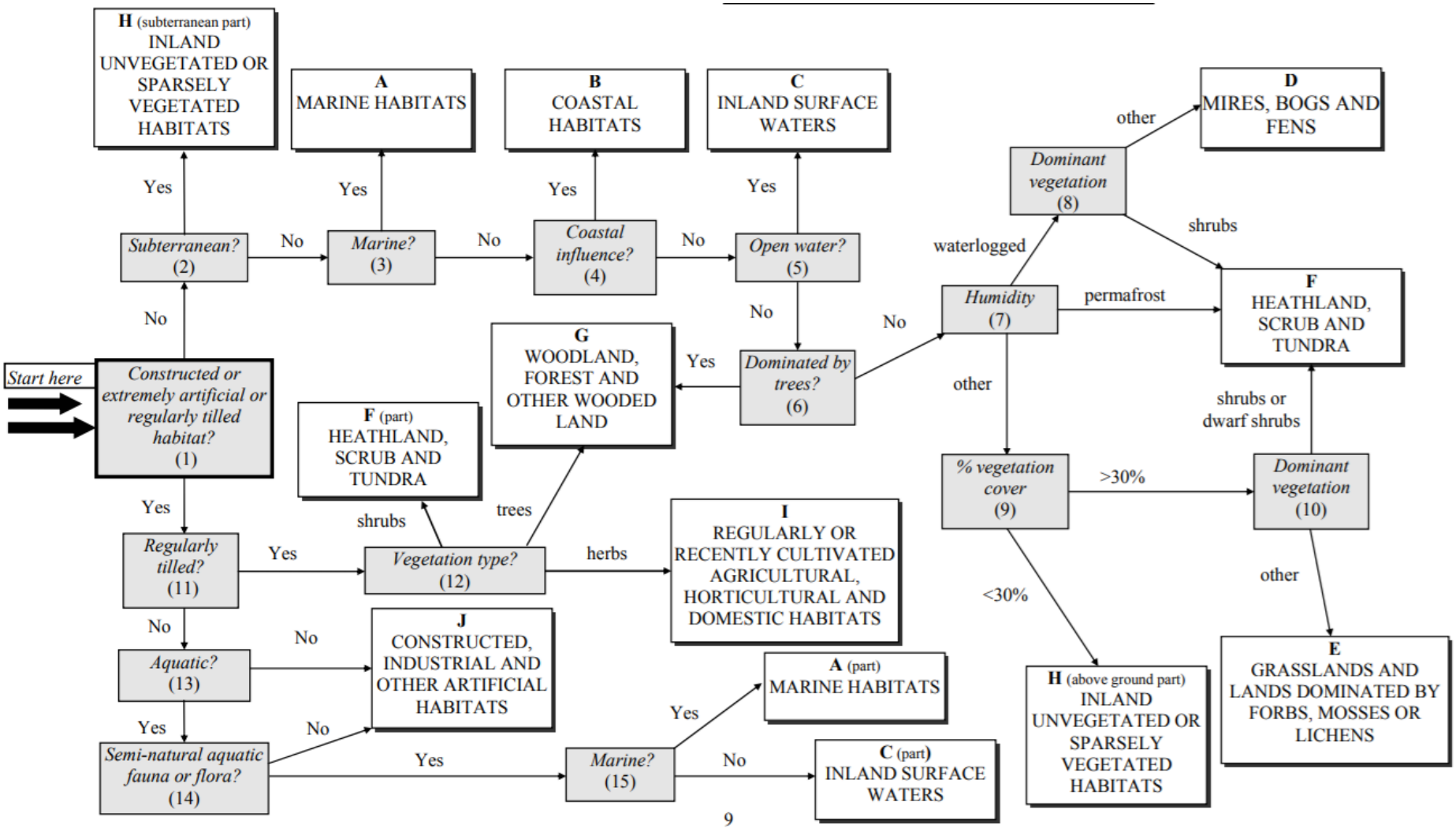
A6. Deep-sea bed

A7. Pelagic water column

A8. Ice-associated marine habitats

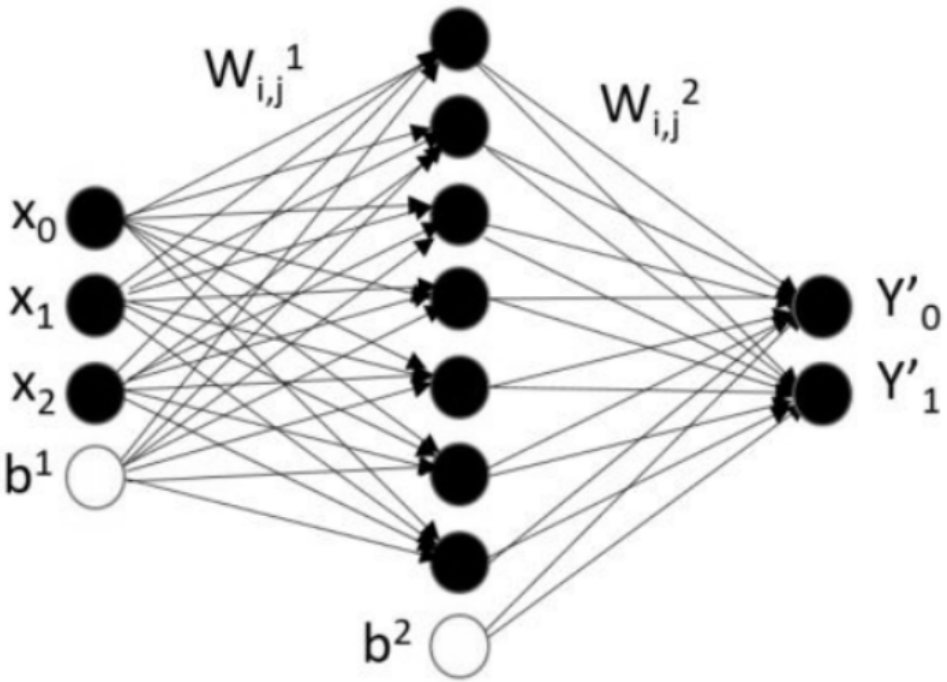
- EUNIS is a Hierarchical habitat classification system that gathers information from different sources that focus on the European continent and sea area.

EUNIS pipeline for level 1 classification

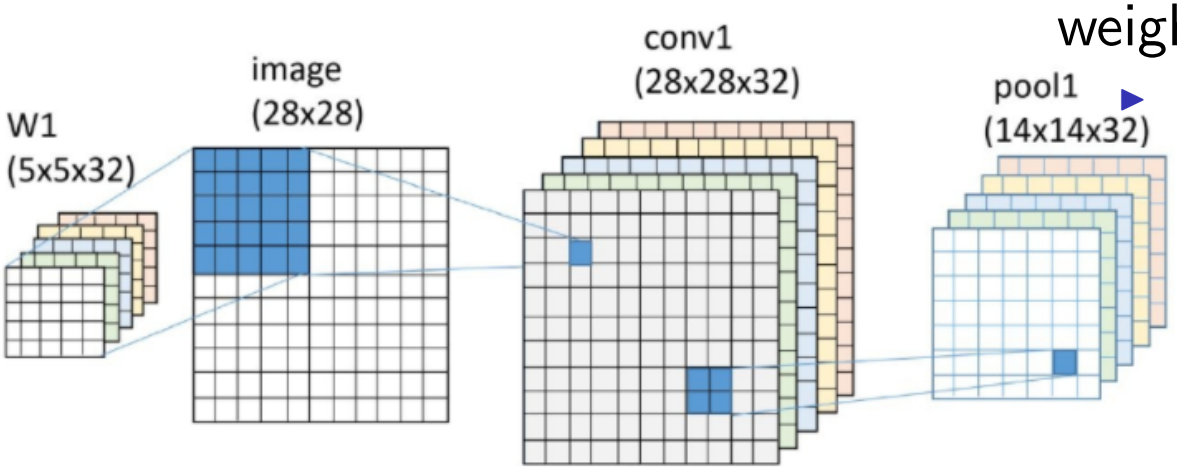


9

Machine Learning

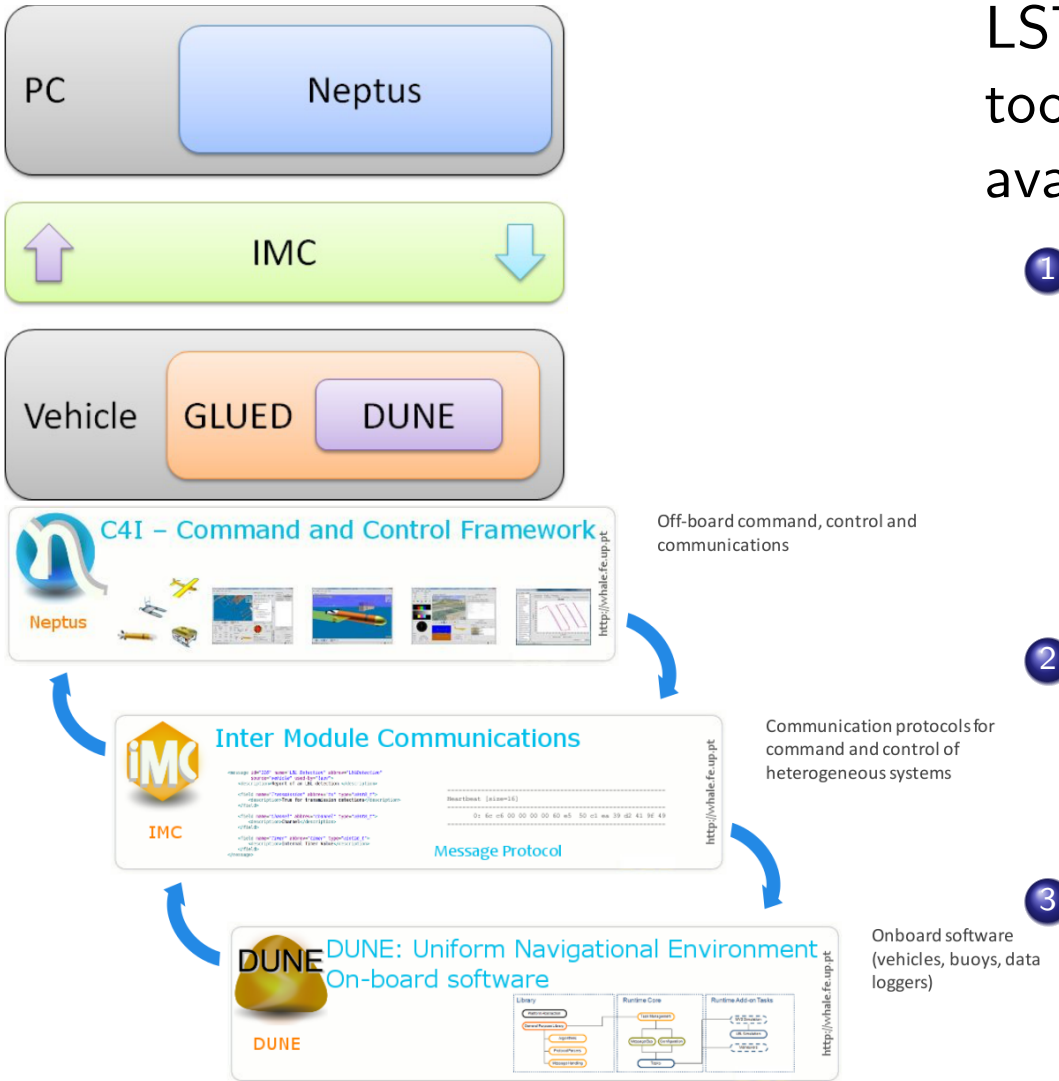


- The human brain has $\approx 10^{11}$ neurons where some are connected between themselves making complex decisions.
- A node, or neuron connects to another and can be on, sending data to the next layer, or off.
- Each node as its own linear regression model: input data, weights, bias and output.



- ★ By training we adjust the weight and bias to minimize our cost function.
- ★ CNNs are a type of NN specialized in detecting patterns.

LSTS software toolchain



LSTS has created a software toolchain for autonomous vehicles, available open-source at GitHub:

1. IMC: a message-based protocol for interoperability between nodes in a network such as vehicles or human operator consoles;
2. DUNE: an on-board software platform for autonomous vehicles;
3. Neptus, a command-and-control infrastructure that allows users to prepare, monitor, and review vehicle operations using GUI consoles.

Contributions

- Deep learning models for habitat mapping for 3 EUNIS sub-categories.
- Models trained with level 2 EUNIS categories using a dataset of 300 images per label (training, validation and test set).
  - 2 models with a simple architecture.
  - 1 model using Transfer learning.
  - 3 models derived using Google Auto ML.
- Onboard software platform for real-time image classification.
- Simulation results on Raspberry PI and Jetson Nano platforms.
- Previous work on OMARE project obtained a CNN with **85%** accuracy compared to this dissertation **93%**.

Dataset PNLN

Category	Images
A3 – Infralittoral rock and other hard substrata	**484**
A3.1 – Atlantic and Mediterranean high energy infralittoral rock	388
A3.11 – Kelp with cushion fauna and/or foliose red seaweeds	113
A3.7 – Features of infralittoral rock	96
A3.71 – Robust faunal cushions and crusts in surge gullies and caves	44
A4 – Circalittoral rock and other hard substrata	**1385**
A4.1 – Atlantic and Mediterranean high energy circalittoral rock	1207
A4.7 – Features of circalittoral rock	178
A4.71 – Communities of circalittoral caves and overhangs	178
A4.711 – Sponges, cup corals and anthozoans on shaded or overhanging circalittoral rock	20
A5 – Sublittoral sediment	**300**
A5.1 – Sublittoral coarse sediment	181
A5.13 – Infralittoral coarse sediment	2
A5.14 – Circalittoral coarse sediment	130
A5.2 – Sublittoral sand	85
A5.23 – Infralittoral fine sand	4
A5.25 – Circalittoral fine sand	81
A5.4 – Sublittoral mixed sediments	34
A5.44 – Circalittoral mixed sediments	34
Total	**2169**

Dataset

- Some categories have nearly no samples while others have much more comparing the same hierarchy level.
- Due to class imbalance and lack of samples overall we are forced to reduce the size of our data set and/or try data augmentation methods.

Category	**Total**	**Train**	**Validation**	**Test**
A3 – Infralittoral rock and other hard substrata	300	210	45	45
A4 – Circalittoral rock and other hard substrata	300	210	45	45
A5 – Sublittoral sediment	300	210	45	45
Total	900	630	135	135

- These allow for a model that covers categories at the same level and there is enough data to sample in all of them.

Developed models

```
model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=
    (64,64,3)),
tf.keras.layers.Dense(128,
    activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(3,
    activation='softmax')
])
```

Listing 1: Mnist

```
model = tf.keras.Sequential([
hub.KerasLayer( "https://tfhub.dev/
google/tf2-preview/mobilenet_v2/
feature_vector/4",
output_shape=[1280],
trainable=False,
input_shape=(224,224,3)),
tf.keras.layers.Dense(3,
    activation='softmax')])
```

Listing 2: MobileNet V2

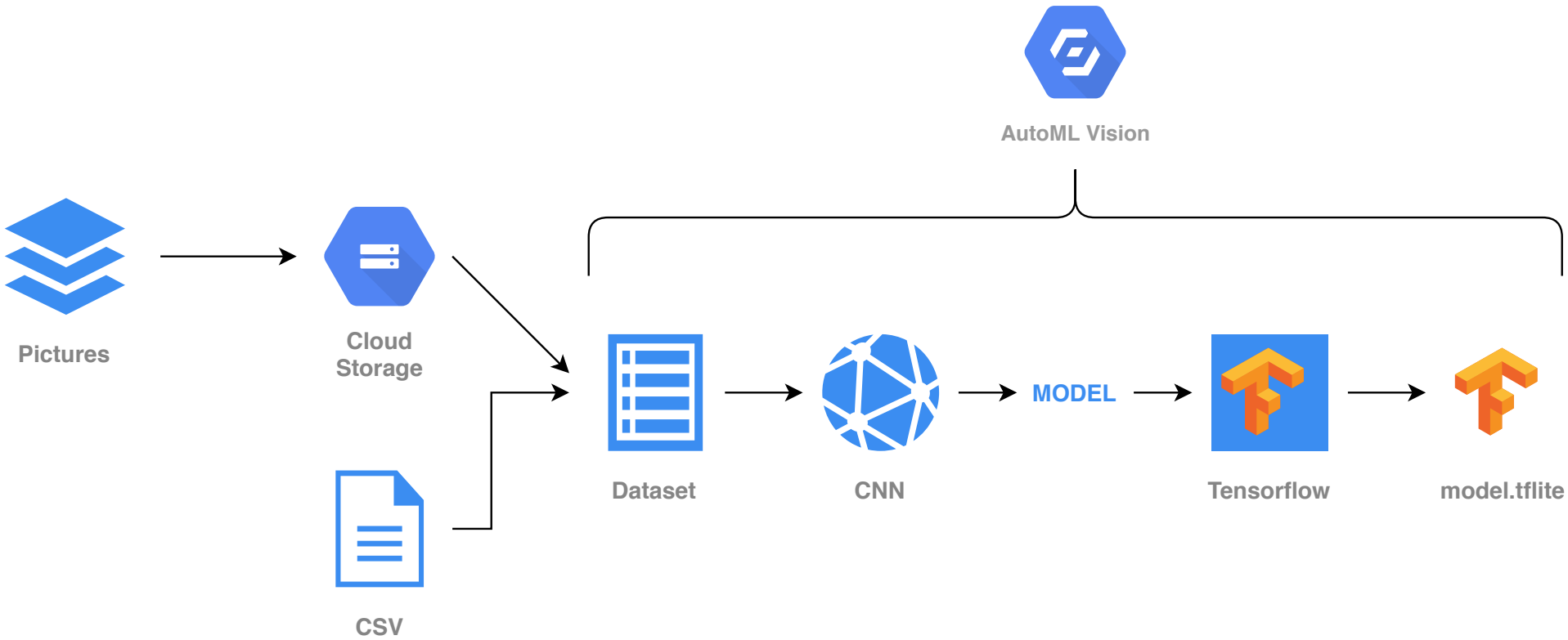
```
model = Sequential()
model.add(Conv2D(16, 3, padding= 'same',
    activation='relu', input_shape=(64,
    64, 3)))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, 3, padding= 'same',
    activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, 3, padding= 'same',
    activation= 'relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(512, activation= 'relu'))
model.add(Dropout(0.2))
model.add(Dense(3, activation= 'softmax'))
])
```

Listing 3: Udacity

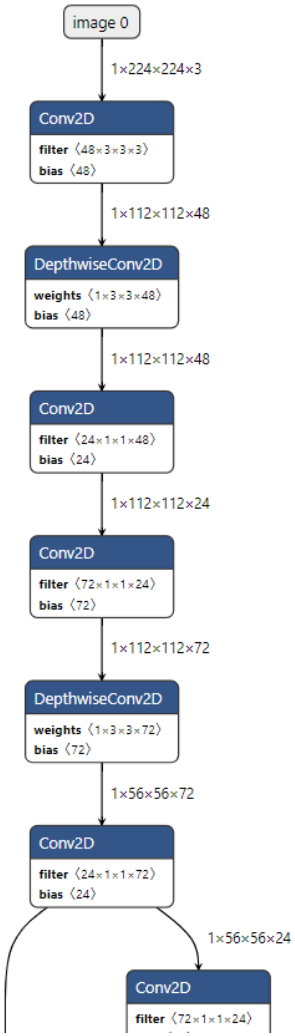
Developed models - Google Auto ML

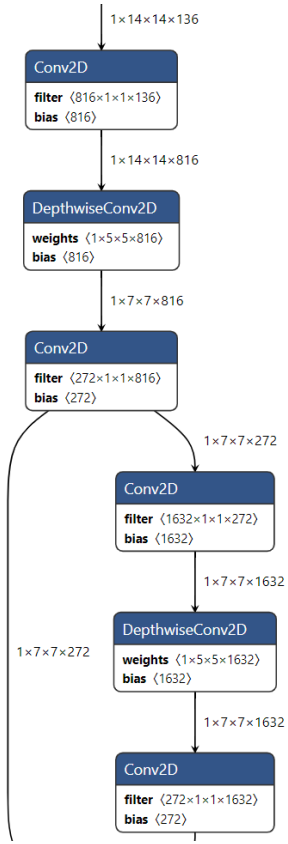
Derivation of 3 TFLite models using Google Auto ML:

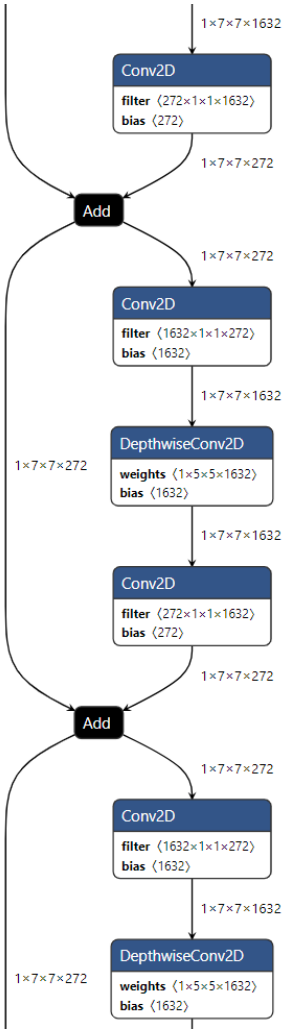
- autoML_fast.
- autoML_medium.
- autoML_slow.

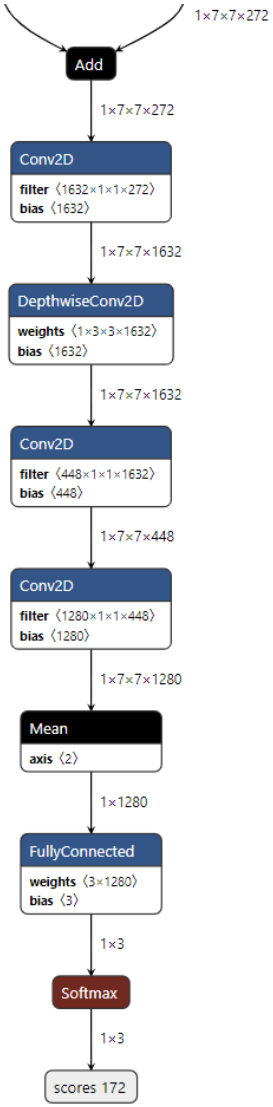


Model visualization

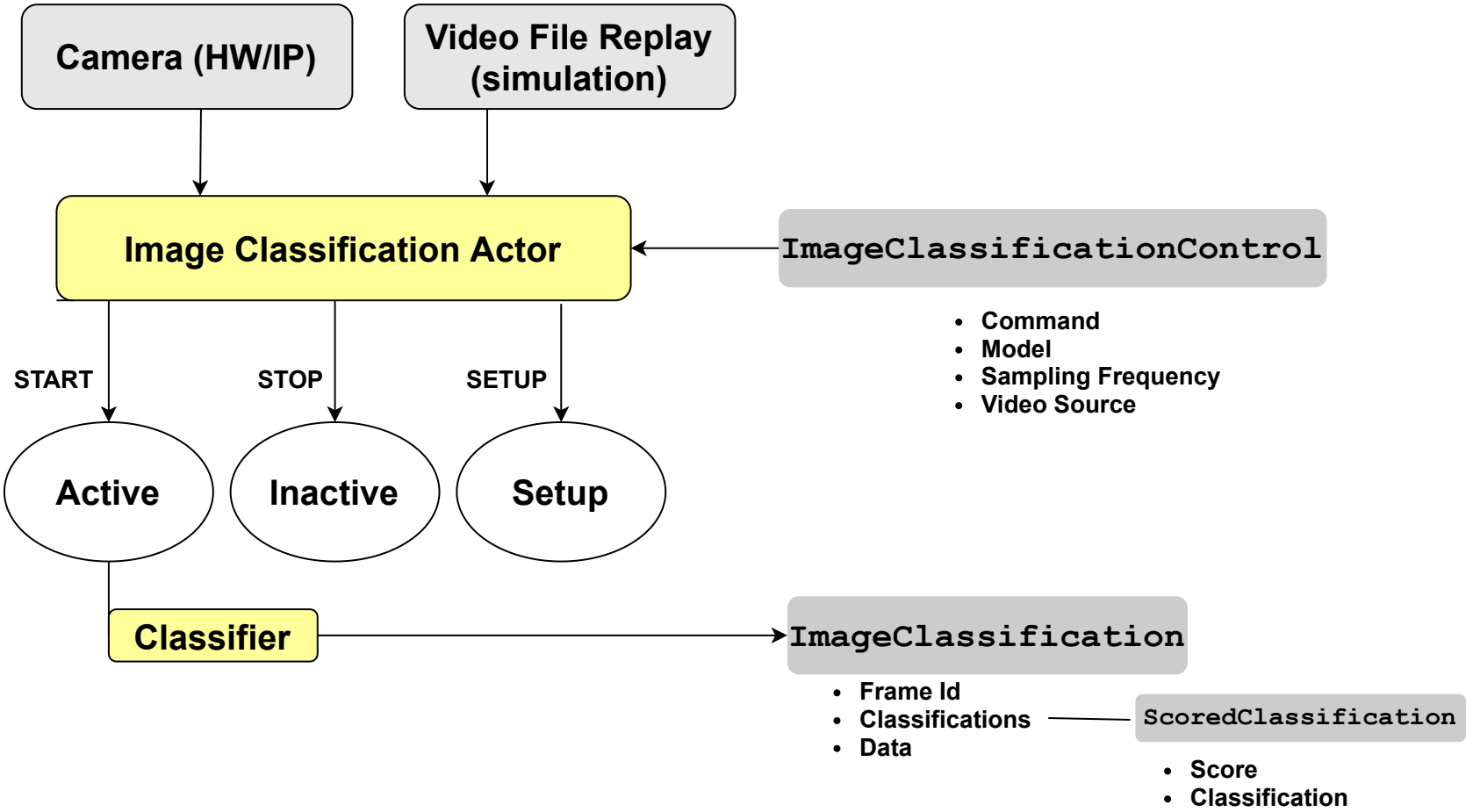








Architecture of the onboard software

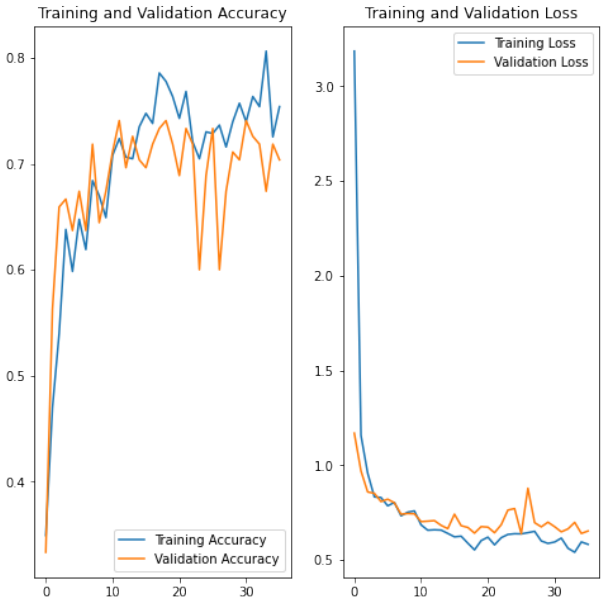


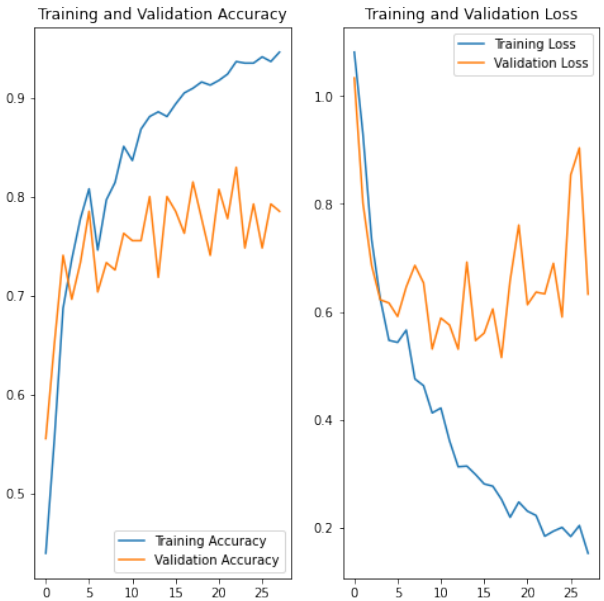
- OpenCV used for capturing imagery and Tensorflow Lite is used for machine learning as it is the most appropriate for embedded systems.
- The pyimc library takes the XML specification of IMC to generate the corresponding Python bindings for each type of message.

IMC messages used by the ICA

IMC message	Field Name	Field Type	Summary
ImageClassificationControl	command	uint8_t	Command flag (0: START, 1: STOP, 2: SETUP)
	model	plaintext	Identifier of model (required for SETUP command).
	video_source	plaintext	Identifier of video source path (required for SETUP command).
	sampling_freq	fp32_t	Image classification frequency in Hz (required for SETUP command)
ImageClassification	frame_id	uint32_t	Unique identifier for frame.
	classifications	message-list	Classifications, a list of messages of type ScoredClassification.
	data	rawdata	Original image frame, possibly compressed.
ScoredClassification	score	uint8_t	Classification score, from 0 to 100.
	classification	plaintext	Classification label.

CNN training results - Mnist, Udacity, MobileNet






```
history = model.fit(
      train_dataset,
      epochs=max_epochs,
      validation_data=validation_dataset,
      callbacks = [stop_early]
)
```

CNN architecture comparison

Model	Layers	Parameters (10^6)	File size (MB)	Accuracy (%)	Loss
Mnist	4	1.6	6.3	84	0.49
Udacity	11	2.1	8.5	86	0.36
Mobilenet	68	2.3	8.9	90	0.32
AutoML fast	66	0.5	0.6	93	0.28
AutoML medium	65	3.1	3.2	92	0.39
AutoML slow	65	5.8	5.9	93	0.24

- Higher model density achieves higher accuracy and lesser loss.
- Smaller models still achieve high accuracy.
- Google Auto ML models have a small size and high accuracy.

Hardware platforms

Platform	CPU	RAM	GPU	Operating system
Raspberry Pi 4	ARM Cortex-A72 Quad-Core 1.5 GHz	8	Broadcom VideoCore VI	Raspbian Linux 10 (buster)
Nvidia Jetson Nano	ARM Cortex-A57 Quad-Core 1.43 GHz	4	128 core Maxwell GPU 921 Mhz	Ubuntu Linux 18.04
HP laptop	Intel i5-1035G1 octa-core 1 GHz	8	Intel Corporation Device 8a56	Ubuntu Linux 20.04.3

Average prediction time (ms)

Model	Raspberry Pi	Jetson Nano	HP laptop
Mnist	2.47	6.36	0.74
Udacity	6.06	11.62	2.33
Mobilenet	94.54	107.02	23.39
AutoML fast	48.51	31.51	15.66
AutoML medium	127.73	86.54	43.92
AutoML slow	183.35	131.24	67.65

- Correlation between model complexity and performance.
- Simpler model Mnist runs the quickest with good accuracy.
- Worst performing model AutoML slow runs at 183.35 ms, 5 Hz.
- AUV cover 1-2 meters per second, and even the slowest model is enough to not lose information.

RAM and CPU usage

System + model	RAM (average GB)	CPU (idle)	CPU (1 Hz)	CPU (7 Hz)
Raspberry Pi - Mnist	1.9	33	49	98
Raspberry Pi - AutoML slow	1.8	33	64	100
Jetson Nano - Mnist	1.3	37	48	100
Jetson Nano - AutoML slow	1.3	37	59	100
Hp Laptop - Mnist	1.5	24	32	81
Hp Laptop - AutoML slow	1.5	24	37	113

- Only the simplest and the most complex models were considered.
- Not much variation is registered.
- Most of the examples run on 1 core.

Summary

- Software framework for automated image classification using machine learning models that can be deployed in autonomous vehicles.
- Several trained machine learning models for habit mapping of three EUNIS categories, based on dataset of underwater imagery.
- Models can be employed in embedded software platforms with good performance, both in terms of computational efficiency and classification accuracy.

Future Work

- Real-life scenarios in field tests.
- Handling the technical problems regarding the use of a GPU.
- The consideration of aerial, side-scan sonar imagery and other EUNIS categories given more dataset annotations.
- Automatic identification of man-made objects in the sea for pollution monitoring.
- Estimate values of a ROV that does not possess built-in sensors like the motion of the vehicle or water turbidity using the underwater camera.

Questions?