
Ocean Data
and AI for
species
conservation
Capgemini

Abstract.

The problem and at the same time our motivation is the loss of species diversity in the ocean, which is often also referred to as ”invisible dying”. With our approach, we pursue the goal of making this dying visible and thus preventable. To make events in the ocean visible, we need to identify patterns depending on the ocean depth and then recognize deviations from these patterns. With the Norwegian institute Lofoten-Vesterålen, we are analyzing ocean data, to help detect anomalies. This should enable a better understanding of the ocean ecosystem. This should help to identify the consequences of human intervention in nature, such as dwindling fish stocks.

Introduction

Accurate observation of ecosystems enable detailed oceanographic research, allowing anomalies to be identified in enormous amounts of data with the help of artificial intelligence (AI).

The Lofoten-Vesterålen (LoVe) Ocean Observatory is located west of Hovden Vesterålen in the northern part of Norway. It is located in an ecological, geological, oceanographic and economic ”hotspot”. A network of submarine cables and seven sensor nodes covers a cross-section from the mainland to the deep sea. It includes a land-based station and seven sensor platforms, covering a gradient from sea level to a depth of 200m. The system continuously provides valuable online data on the marine environment in northern Norway, and has been active since 2013.

The system is both, a national research infrastructure, basic and applied research, as well as a test infrastructure, where industry partners can test new underwater sensors and technologies. The Lofoten-Vesterålen Ocean Observatory has collected over 100 terabytes of sensor data (temperatures, currents, echograms) over the years.

The team

Thomas Ramm

is a Software Engineer at Capgemini. He created the initial infrastructure and GitHub integration.

Majed Alaitwni

is a software developer. The focus of his bachelor thesis was interactive visualization for anomaly detection in ocean measurement data. He created the visualization.

Geir Pedersen

is a researcher at the LoVeOcean Ocean Observatory and supports the project on the Norwegian side.

Tom Hatton

is a Data Scientist, his master’s thesis explored the use of unsupervised AI models for anomaly detection in high-dimensional ocean measurement data. He continues to develop the AI model.

Sophie Bader

is a molecular biologist specializing in oceanic ecosystems, as well as a software developer. She assisted with the infrastructure.

Mustapha Mustapha

is a software developer at Capgemini. He has done planning work and helped design the original AI model.

Daniel Friedmann

is a software developer at Capgemini. He is an expert in containerization and Docker and provides content support for the project.

Nils Olav Handegaard

is a researcher at the LoVe Ocean Observatory. His research focuses on the application of new methodologies and data processing techniques to the fields of marine ecology and fisheries oceanography.

Eldar Sultanow is Enterprise Architect Director at Capgemini. His main focus is on modern software architectures, digitalization and enterprise architecture management. He developed the code with Thomas Ramm in the initial phase and is now supervisor of the project and oversees research work.

OceAIn was created as a team name to participate in Capgemini's Global Data Science challenge (GDSC) 2021. The goal of OceAIn was to develop an AI model that gains new insight into seasonal correlating patterns in ecosystems using the time series data which was collected by ocean sensors. This should help to build better models and understand the climate of our planet.

The AI model processes data of the cross section from the mainland to the deep sea. They are collected by four different sensors that measure (1) directional pulsating sounds in specific areas with a scientific echo sounder, (2) a so-called hydrophone, i.e. an underwater microphone that records sounds in the environment, (3) an Acoustic Doppler (ADCP) that detects the speed and direction of ocean currents using the Doppler effect, and finally, (4) point sensors that provide real-time physical, biological and chemical observations.

The Identification of repeating seasonal patterns and anomalies allows scientists to better monitor the marine environment. This involves widespread exploration of the anomalies and their influencing factors and drawing conclusions from the bigger picture, such as differences in fish populations, varying current patterns, or the influence of climate change.

While large volumes of raw data are difficult to process manually and the results are highly error-prone in the process, AI models allow filtering of this data for relevant events. In addition, AI enables continuous analyzation of incoming data, resulting in a stream of data to the researchers.

Architektur des Systems Even though the AI model is the core of OceAIn, there are other components that make up the platform. Our next goal is to make them work together according to the cloud-native architecture concept, which will create a future-proof and flexible data pipeline. The raw data provided by the institute will be collected in one step and cleaned and transformed in the next. These steps will take place in Docker containers, with each type of data (hydrophone, biomass detection, etc.) having its own container in every step. The collaboration of the containers is defined in

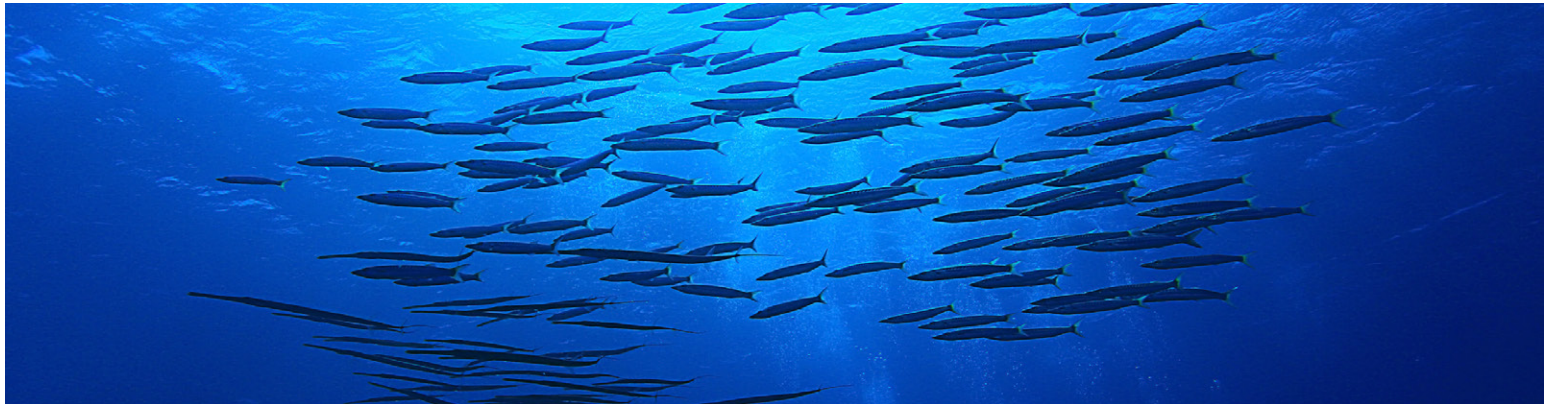
Apache Airflow, which works on the "Configuration As Code" principle. Airflow allows the definition of infrastructure using DAGs (Directed Acyclic Graph). It is also worth mentioning that Docker containers are actually managed by Kubernetes, which in turn is defined by Airflow. Finally, the transformed data is persisted in the form of CSV files. They are processed by the AI to detect anomalies, which are displayed with the data in the form of image files through an interactive web interface.

Implementation of the AI model OceAIn includes an AI model that detects anomalies in ocean data. The sensor data for the AI model are highly variable in the type of information, as well as in their duration. In addition, some types of anomalies are only detectable if the data points are considered interconnected from the beginning.

The initial idea was to focus on individual models that could handle different types of data and later combine the separate models. Due to the variety of data, this approach showed some disadvantages, such as increased complexity in aggregating the individual models.

Table 1. Results of the Baseline Models

MODEL	F1 SCORE	PRECISION	RECALL	MACRO AVG F1
always true	1.00	0.30	0.47	0.23
always false	0.00	0.00	0.00	0.41
uniform random	0.54	0.40	0.46	0.58
stratified	0.17	0.21	0.19	0.44



The idea that ultimately prevailed was to use a deep-learning neural network that analyzes all the data in their entirety for anomalies. At first, unsupervised models were supposed to be used, but this approach turned out not to be feasible.

The use of unsupervised AI models has the advantage that the training of those models does not rely on the presence of labeled training data (anomaly vs. normal).

However, those models are extremely vulnerable to data noise and corruption [1]. The underlying ocean measurement data are also subject to these characteristics, which are further exacerbated by their whole-scale nature and the high data dimensionality that accompanies them.

The contribution of labeled data by the researchers has enabled the use of supervised AI models. Basically, supervised AI models outperform their unsupervised counterparts in anomaly detection, as they are particularly capable of detecting application-specific anomalies [2].

To check the performance of the unsupervised models later, four baseline models were created first, the results of which can be seen in 1. These are a common tool in the evaluation of machine learning models and are naive solutions for a classification problem.

By comparing the results of a real model with those of the baseline models, conclusions about the correctness of the real models can be made. Of the baseline models used here, one always classifies false, one always classifies true, one splits the datasets in half between true and false, and the last stratifies the data based on their labels.

During the model development, two types of models were created. These are reconstructionbased models and predictive models. For the former, three different microarchitectures were created. The first uses one-dimensional convolutions in each of the hidden layers, where the amount of filters is determined by a hyperparameter. The second architecture uses so-called LSTM layers and the third is a fully connected autoencoder.

Table 2. Results of all reconstruction models

RECONSTRUCTION				
CLEAN DATA			FULL DATA	
dense	lstm	conv	lstm	conv
dense W32 HL400 40 dense W32 HL400 40 v2 dense W8 HL400 40 20 4	lstm W32 HL100 40 20 4 lstm W32 HL100 40 20 4 v2	dense W16 HL100 40 20 4 dense W32 HL100 40 20 4 dense W32 HL2000 1000 100 40 20 4 dense W32 HL400 40 dense W32 HL400 40 v3	lstm W32 HL100 40 20 4 lstm W32 HL100 40 20 4 v2 lstm W32 HL40 lstm W32 HL400 100 10 lstm W32 HL64 32 16 lstm W8 HL100 40 20 4	conv W16 HL1000 100 10 conv W16 HL1000 400 100 4 conv W16 HL100 40 20 4 conv W32 HL100 40 20 4 conv W64 HL100 40 20 4

predictive models also fall into three categories. These are also LSTM and fully connected hidden layers, as well as an architecture that uses convolution and max-pooling. The naming of the models follows a fixed pattern. First, it is specified which architectural scheme a model corresponds with. Then a ”W” and a number is given, which represents the window size. ”W32” thus describes a neural network with a window size of 32. This is followed by further numbers, which indicate the size of the hidden layers. Optionally, there is also a version number at the end, which indicates models that had delivered promising results in the first instance, which is why they are run several times. Since the models described above do not in themselves detect anomalies, there is another component that is responsible for exactly that. The results of these models were sobering. Hardly any of the models could exceed an F1 score of 0.5, which means that they were no better than the baseline models with random division of the values. In fact, there was only one model, ”lstm W32 HL128 128 128” which could (minimally) exceed this limit. All tested models can be seen in the tables 2 and 3, while 4 shows the average results of the different types of models.

There are several reasons for the comparatively poor results of unsupervised models. The data itself is rather unsuitable for an unsupervised model. Gaps, noise, and data corruption greatly degrade the results of unsupervised models.

Table 3. Results of all prediction models

RECONSTRUCTION				
CLEAN DATA			FULL DATA	
dense	lstm	conv	lstm	conv
dense W32 HL256*5 v2	lstm W32 HL128 128 128	conv W32 HL128 128 128 conv W32 HL128 128 128 v2 conv W32 HL32 32 32	lstm W32 HL128 128 128	conv W32 HL32 32 32

Table 4. Average results of all model types

MODEL	QUANTILE	F1 SCORE	PRECISION	RECALL	MACRO AVG F1
reconstruction dense clean data	0.73	0.43	0.34	0.73	0.41
reconstruction lstm clean dat	0.85	0.41	0.41	0.67	0.4
reconstruction conv full data	0.85	0.46	0.38	0.68	0.5
reconstruction dense full data	0.85	0.28	0.32	0.39	0.46
reconstruction lstm full data	0.85	0.4	0.33	0.62	0.46
forecast conv clean data	0.85	0.47	0.32	0.9	0.35
forecast dense clean data	0.73	0.48	0.32	0.95	0.33
forecast lstm clean data	0.73	0.48	0.32	0.98	0.3
forecast lstm full data	0.73	0.47	0.31	0.94	0.31
forecast conv full data	0.85	0.47	0.32	0.95	0.33

In addition, the datasets are highly dimensional, with a large number of sensors collecting information simultaneously. The more complex a dataset is, the harder it is to train AI models to produce correct results. Presumably, our models would produce far better results if the data was less complex. Another issue is the limited computational capacity we have. The compression ratio of the models is high because adding more layers would have required significantly more computation time, which was not feasible. Lastly, it should be mentioned that extensive hyperparameter optimization has not taken place yet. This could mean that the models themselves are actually better than assumed.

For all these reasons, it can be concluded that unsupervised models are unsuitable for anomaly detection in ocean data as it is generated by LoVe. Thus, OceAIn shall use a model that learns in a supervised manner. This model will later be continuously retrained based on data generated by the researchers during operation.

A snippet of code for the model is shown in listing 1. Here, the datasets that deviate significantly from the expected normal are detected. This is done by first iterating over all the datasets, while all datapoints that are detected as deviations are stored in the array anomalous_data_indices.

Listing 1. Code snippet of the anomaly detection module

```
# Detect all the samples which are anomalies.
anomalies = test_mae_loss > threshold
print("Number of anomaly samples: ", np.sum(anomalies))
print("Indices of anomaly samples: ", np.where(anomalies))

plt.plot(x_test[0])
plt.plot(x_test_pred[0], alpha=0.7)
plt.show()

anomalous_data_indices = []
for data_idx in range(TIME_STEPS - 1, len(X_test) - TIME_STEPS + 1):
    if np.all(anomalies[data_idx - TIME_STEPS + 1 : data_idx]):
        anomalous_data_indices.append(data_idx)
anomalous_data_indices

```

Visualization of the vast amounts of data

The visualization of the results generated by the AI models and the associated raw data is done in the form of a web application, whose structure can be seen in the form of a class diagram in Figure 1. The Visualizer class depends on the External Framework ThreeJS, and is responsible for generating the image files from the CSV files. The Visualizer is also extended with a separate component for each type of sensor data. The class "App", in turn, is responsible for initializing the sensor classes and processing user input.

On the server side, Python and Docker are used, while various packages are provided that contain the functionality for data synchronization and the creation of measurement data representations. On the client side, when executing the web documents sent by the server, control elements are created in the browser, which convert the user's interactions into corresponding requests. Javascript and the library Three.js are used for this purpose.

The visualization of the results generated by the AI models and the associated raw data is done in the form of a web application, whose structure can be seen in the form of a class diagram in Figure 1.

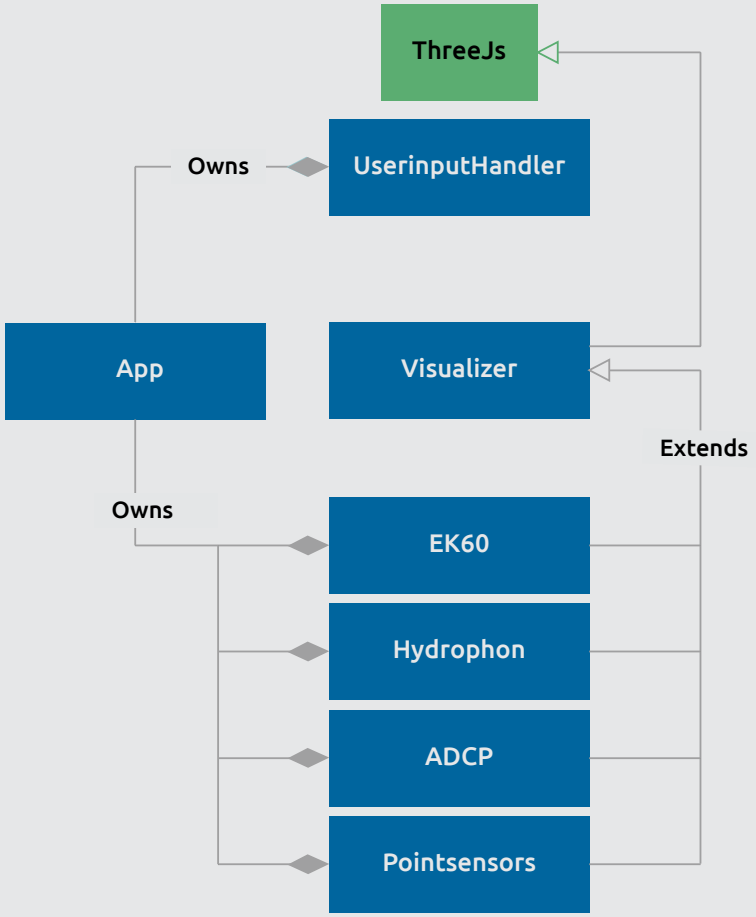


Figure 1. Overview of the main component of the client-side application

The actual visualization of the measurement data on the client side is done as following:

There are four canvas elements arranged as shown in figure 2. They contain the visualized data of different sensors, where each element can be moved on the x-axis, and thus in the temporal space. Movement on the y-axis, on the other hand, you move in a sensor specific space.

At the top left, the data from the EK60 sensors are evaluated. These sensors can detect how much biomass is present at a certain location. This is visible through the color gradient of the generated image. The brighter an area is, the more biomass was detected at the corresponding location. Vertical movement adjusts the depth of the captured data.

At the top right, acoustic signals recorded with hydrophones are displayed. Here, only a shift on the X-axis is possible.

At the bottom left, from top to bottom, flow velocity, strength and direction are displayed. See also 3. Just like the EK60 sensors, the vertical position determines the depth of the acquired data, while the brightness of the colors determines the speed or intensity of the currents.

Finally, at the bottom right are the point sensors. These capture different data, such as water composition, salinity, or the presence of certain chemicals, and display them as graphs.

On the server side, a conversion of the measurement data from CSV first into JSON and then into image formats (such as PNG, JPG and JEPG) takes place. In this process, only the truly important data is visualized. Thus, the unimportant part is ignored from the start, which means that only a fraction actually needs to be displayed. The conversion of the measurement data into image formats allows a reduction of the sent data to the client, so that large time intervals can be visualized and all measurement data information can be displayed.

On the client side, the representations created on the server side are retrieved and displayed by the client-side components. Figure 2 shows the implemented user interface, which displays a section of the measurement data of the EK60 sensors, hydrophones, ADCP sensors and point sensors for a period of three days. Each of the four areas can be controlled by a control bar. The X- and Y-positions of the mouse pointer in the visualization area are interpreted by the client without making any requests to the server, so that the X-position leads to the display of the time stamp for the measurement data displayed below, and the Y-position leads to the display of the Y-axis values of the measurement data. For example, the EK60 and ADCP display the depth values. The ADCP measurement data is also used to calculate the currents, which makes it possible to establish correlations between water currents and the distribution and movement of biomass from the EK60 measurement data. This is illustrated in figure 3.

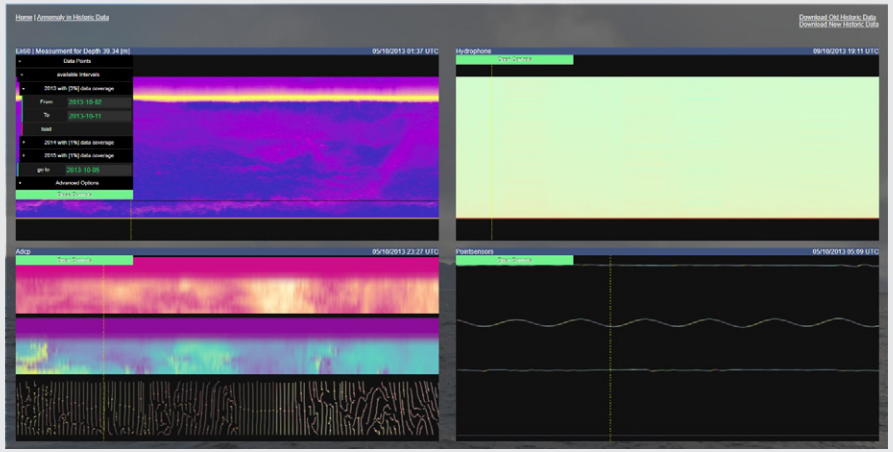


Figure 2. The GUI for visualizing a section of the measurement data of all sensors

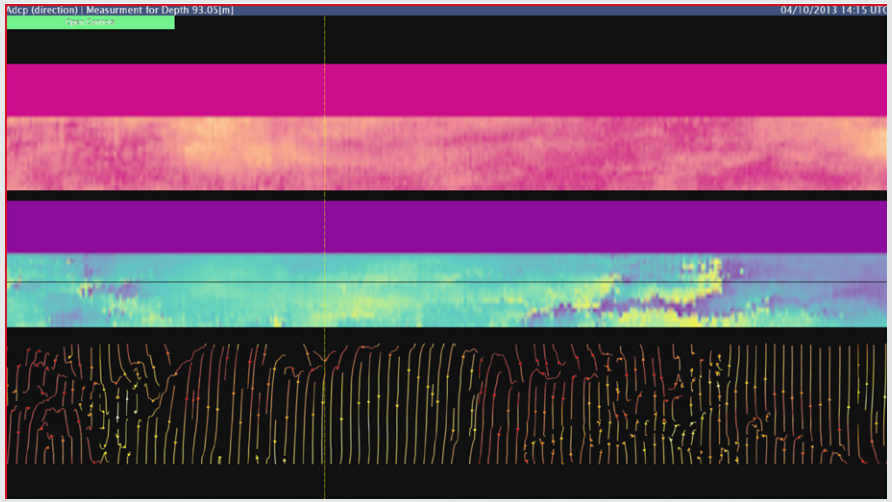
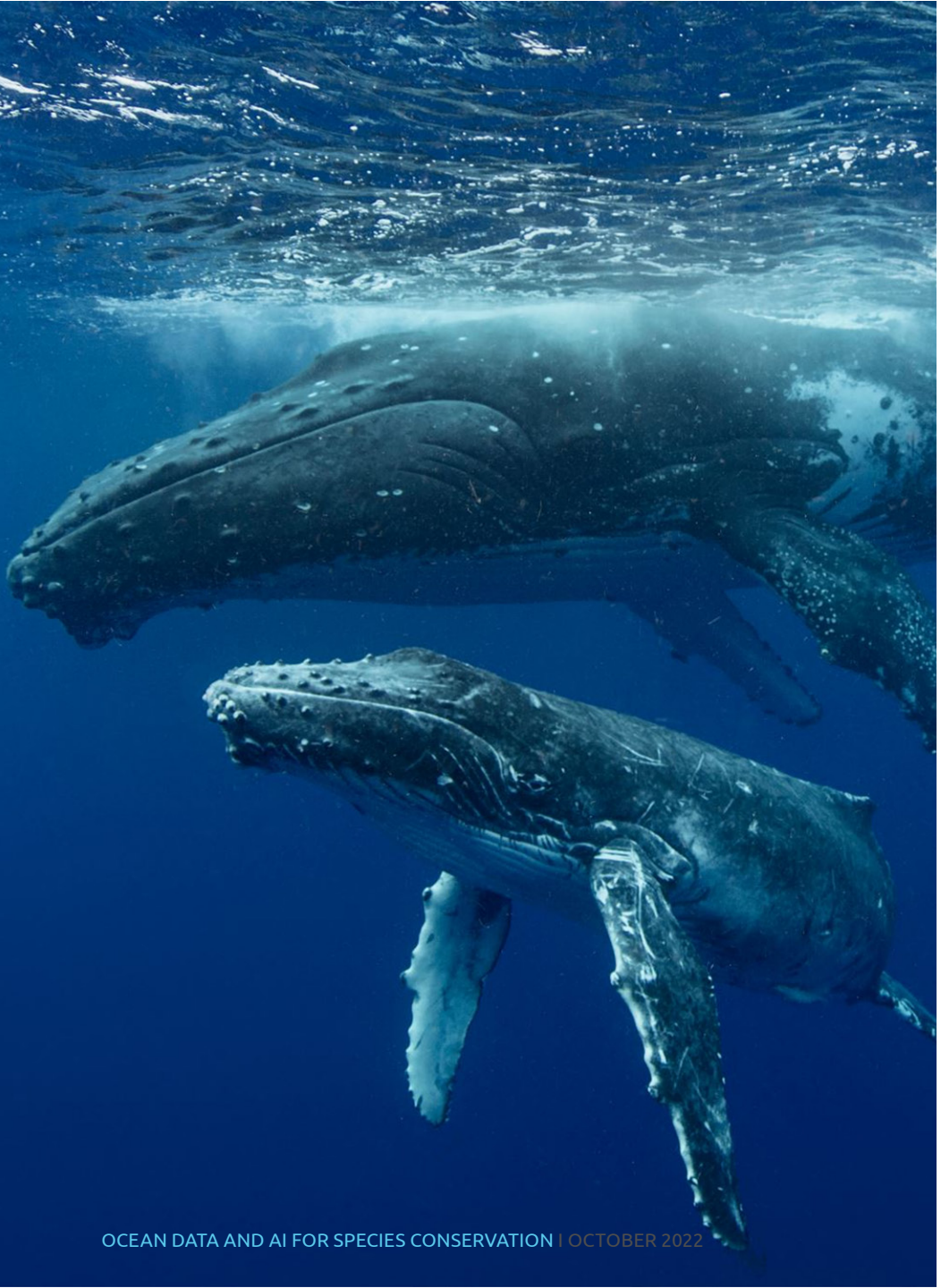


Figure 3. Visualization of ADCP measurement data in fullscreen mode.

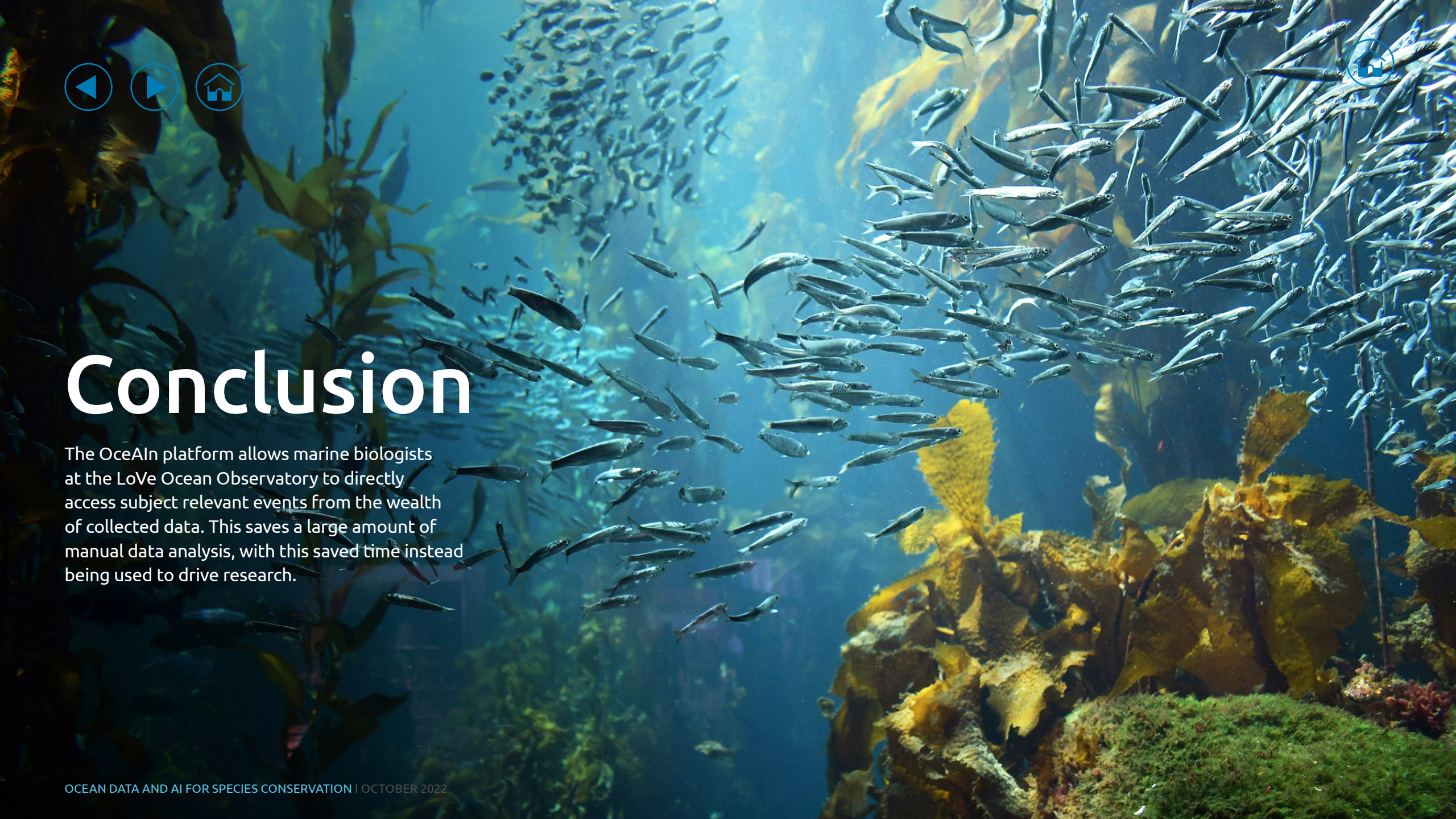


Listing 2 shows the function to˙streamplot, which is responsible for the calculation of the flow data. The variable plt represents the external Python library Matplotlib. In this function the ADCP datapoints are supplied as parameters. With the help of the Matplotlib function streamplot a flow diagram is generated, which shows the calculated ocean currents. The diagram is saved as an image file, from where it is later displayed on the web interface.

Listing 2. Code for the calculation of the currents for the visualization

```
def to_streamplot(X, Y, u, v,target_file_path,config):
    _,ax = get_fig_ax()
    ax.streamplot(X, Y, u, v,linewidth=config['linewidth'],
                  color=u,density=config['density'],
                  cmap=config['cmap'],
                  arrowstyle=config['arrowstyle'])
    plt.savefig(target_file_path,
    format=str(target_file_path.suffix).replace('.',''),
                bbox_inches='tight',pad_inches=0.0, transparent=True)
    return target_file_path

```



Conclusion

The OceAIn platform allows marine biologists at the LoVe Ocean Observatory to directly access subject relevant events from the wealth of collected data. This saves a large amount of manual data analysis, with this saved time instead being used to drive research.

Outlook

The OceAIn project shows how collaboration between marine biology and IT can help to protect species and increase the understanding of our environment. However, our work on this project is far from complete. There are several planned features on our roadmap. First, the entire structure, from data collection to processed results will be automated, using containers as described at the beginning of this article. Another goal is to enable continuous and automated retraining of the AI based on the manually marked anomalies. The visualization itself is also not yet complete. Here, for example, there are possibilities for extending the display of the data or the UI features. In addition, the completed website should soon be active continuously, so that it can be used by researchers at the LoVe Institute.

References

[1] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407, 2019.

[2] Charu C. Aggarwal. Time series and multidimensional streaming outlier detection. In Outlier Analysis, pages 273–310. Springer, 2017.

MAJED ALAITWNI

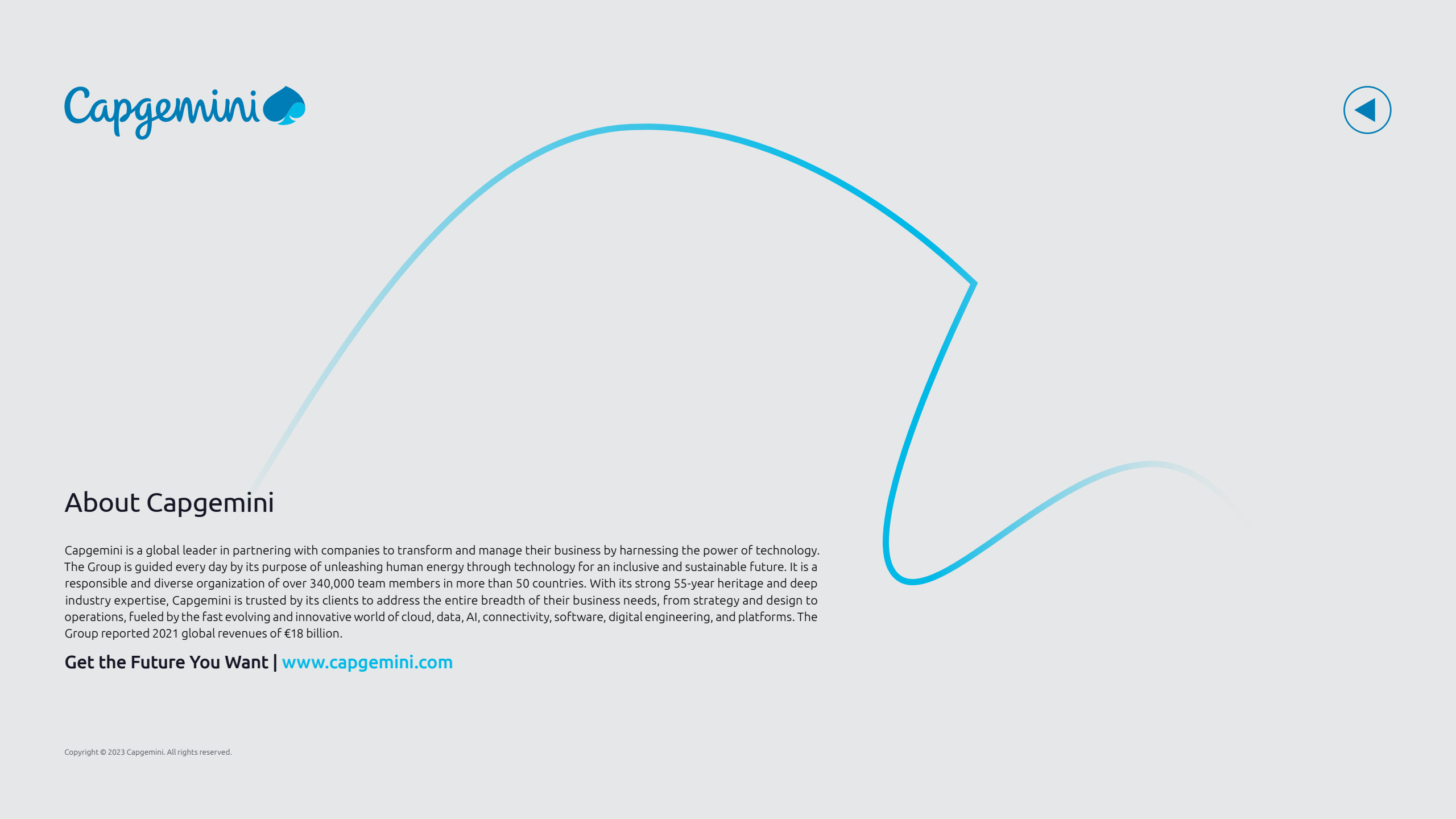
Capgemini, Leipziger Straße 12,
Hannover, Deutschland
Majed.Alaitwni@Capgemini.com

SHIRIN MIAN

Capgemini, Erzbergerstraße 117, 76133
Karlsruhe, Deutschland
Shirin.Mian@capgemini.com

ELDAR SULTANOW

Capgemini, Bahnhofstraße 30, 90402
Nürnberg, Deutschland
Eldar.Sultanow@capgemini.com



About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided every day by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of over 340,000 team members in more than 50 countries. With its strong 55-year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering, and platforms. The Group reported 2021 global revenues of €18 billion.

Get the Future You Want | www.capgemini.com