Pontifícia Universidade Católica
do Rio de Janeiro

**Evelyn Conceição Santos Batista**

# A framework for automated visual inspection of underwater pipelines

**Tese de Doutorado**

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica, do Departamento de Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica.

Advisor: Prof. Wouter Caarls

Rio de Janeiro
September 2023

Pontifícia Universidade Católica
do Rio de Janeiro

**Evelyn Conceição Santos Batista**

# A framework for automated visual inspection of underwater pipelines

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica. Approved by the Examination Committee:

**Prof. Wouter Caarls**
Advisor
Departamento de Engenharia Elétrica – PUC-Rio

**Prof. Leonardo Alfredo Forero Mendoza**
UERJ

**Prof. Raul Queiroz Feitosa**
Departamento de Engenharia Elétrica - PUC-Rio

**Prof. Milena Faria Pinto**
CEFET-RJ

**Dra. Bianca Zadrozny**
IBM

**Dr. Jose David Bermudez Castro**
McMaster University

Rio de Janeiro, September 25th, 2023

**Evelyn Conceição Santos Batista**

Master in Electrical Engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), specializing in the area of signal processing, automation and robotics. Graduated in Electrical Engineering from the University of the State of Rio de Janeiro (UERJ), specializing in the area of systems and computing.

Bibliographic data

Batista, Evelyn Conceição Santos

A framework for automated visual inspection of underwater pipelines / Evelyn Conceição Santos Batista; advisor: Wouter Caarls. – 2023.

90 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2023.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Aprendizado por Reforço Profundo. 3. Detecção de anomalia. 4. Planejamento de ponto de vista. 5. Classificação. 6. Robô Autônomo. 7. AUV. 8. ROV. 9. Framework. I. Caarls, Wouter. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

## Acknowledgments

First and foremost, I would like to express my gratitude to God for providing me with strength and guiding my path.

I extend my sincere appreciation to my advisor, Professor Wouter Caarls, for his teachings and encouragement throughout the completion of this work.

I am thankful to CNPq for the granted assistance.

Special thanks to the ICA and LCI laboratories for their provided support.

To my parents, especially my mother, who has consistently assisted and encouraged me throughout this course, I am profoundly grateful.

I want to express my thanks to all those who have assisted me in any way during this challenging journey, particularly Anna Carolina Leite, Leonardo Forero, Cristian Muñoz, Aline Lassance, Fábio Mourthé, and Guilherme Fadul, for their support, patience, and the numerous occasions we shared knowledge.

To everyone who has directly or indirectly contributed to my journey, I want to convey my heartfelt gratitude.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

# Abstract

Batista, Evelyn Conceição Santos; Caarls, Wouter (Advisor). **A framework for automated visual inspection of underwater pipelines**. Rio de Janeiro, 2023. 90p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

In aquatic environments, the traditional use of divers or manned underwater vehicles has been replaced by unmanned underwater vehicles (such as ROVs or AUVs). With advantages in terms of reducing safety risks, such as exposure to pressure, temperature or shortness of breath. In addition, they are able to access areas of extreme depth that were not possible for humans until then.

These unmanned vehicles are widely used for inspections, such as those required for the decommissioning of oil platforms. In this type of inspection, it is necessary to analyze the conditions of the soil, the pipeline and, especially, if an ecosystem was created close to the pipeline. Most of the works carried out for the automation of these vehicles use different types of sensors and GPS to perform the perception of the environment. Due to the complexity of the navigation environment, different control and automation algorithms have been tested in this area. The interest of this work is to make the automaton take decisions through the analysis of visual events. This research method provides the advantage of cost reduction for the project, given that cameras have a lower price compared to sensors or GPS devices.

The autonomous inspection task has several challenges: detecting the events, processing the images and making the decision to change the route in real time. It is a highly complex task and needs multiple algorithms working together to perform well. Artificial intelligence presents many algorithms to automate, such as those based on reinforcement learning, among others in the area of image detection and classification.

This doctoral thesis consists of a study to create an advanced autonomous inspection system. This system is capable of performing inspections only by analyzing images from the AUV camera, using deep reinforcement learning to optimize viewpoint planning, and novelty detection techniques. However, this framework can be adapted to many other inspection tasks.

In this study, complex realistic environments were used, in which the agent has the challenge of reaching the object of interest in the best possible

way so that it can classify the object. It is noteworthy, however, that the simulation environments utilized in this context exhibit a certain degree of simplicity, lacking features like marine currents or collision dynamics in their simulated scenarios.

At the conclusion of this project, a Visual Inspection of Pipelines (VIP) framework was developed and tested, showcasing excellent results and illustrating the feasibility of reducing inspection time through the optimization of viewpoint planning. This type of approach, in addition to adding knowledge to the autonomous robot, means that underwater inspections require little presence of a human being (human-in-the-loop), justifying the use of the techniques employed.

## Keywords

Deep Reinforcement Learning; Anomaly Detection; Viewpoint planning; Classification; Autonomous Robot; AUV; ROV; Framework.

# Resumo

Batista, Evelyn Conceição Santos; Caarls, Wouter. **Um framework para inspeção visual automatizada de dutos subaquáticos**. Rio de Janeiro, 2023. 90p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Em ambientes aquáticos, o uso tradicional de mergulhadores ou veículos subaquáticos tripulados foi substituído por veículos subaquáticos não tripulados (como ROVs ou AUVs). Com vantagens em termos de redução de riscos de segurança, como exposição à pressão, temperatura ou falta de ar. Além disso, conseguem acessar áreas de extrema profundidade que até então não eram possíveis para o ser humano.

Esses veículos não tripulados são amplamente utilizados para inspeções, como as necessárias para o descomissionamento de plataformas de petróleo. Neste tipo de fiscalização é necessário analisar as condições do solo, da tubulação e, principalmente, se foi criado um ecossistema próximo à tubulação. Grande parte dos trabalhos realizados para a automação desses veículos utilizam diferentes tipos de sensores e GPS para realizar a percepção do ambiente. Devido à complexidade do ambiente de navegação, diferentes algoritmos de controle e automação têm sido testados nesta área. O interesse deste trabalho é fazer com que o autômato tome decisões através da análise de eventos visuais. Este método de pesquisa traz a vantagem de redução de custos para o projeto, visto que as câmeras possuem um preço inferior em relação aos sensores ou dispositivos GPS.

A tarefa de inspeção autônoma tem vários desafios: detectar os eventos, processar as imagens e tomar a decisão de alterar a rota em tempo real. É uma tarefa altamente complexa e precisa de vários algoritmos trabalhando juntos para ter um bom desempenho. A inteligência artificial apresenta diversos algoritmos para automatizar, como os baseados em aprendizagem por reforço, entre outros na área de detecção e classificação de imagens.

Esta tese de doutorado consiste em um estudo para criação de um sistema avançado de inspeção autônoma. Este sistema é capaz de realizar inspeções apenas analisando imagens da câmera AUV, usando aprendizado de reforço profundo para otimizar o planejamento do ponto de vista e técnicas de detecção de novidades. Contudo, este quadro pode ser adaptado a muitas outras tarefas de inspecção.

Neste estudo foram utilizados ambientes realistas complexos, nos quais o agente tem o desafio de chegar da melhor forma possível ao objeto de interesse para que possa classificar o objeto. Vale ressaltar, entretanto, que os ambientes de simulação utilizados neste contexto apresentam certo grau de simplicidade, carecendo de recursos como correntes marítimas ou dinâmica de colisão em seus cenários simulados.

Ao final deste projeto, o Visual Inspection of Pipelines (VIP) framework foi desenvolvido e testado, apresentando excelentes resultados e ilustrando a viabilidade de redução do tempo de inspeção através da otimização do planejamento do ponto de vista. Esse tipo de abordagem, além de agregar conhecimento ao robô autônomo, faz com que as inspeções subaquáticas exijam pouca presença de ser humano (human-in-the-loop), justificando o uso das técnicas empregadas.

## Palavras-chave

Aprendizado por Reforço Profundo; Detecção de anomalia; Planejamento de ponto de vista; Classificação; Robô Autônomo; AUV; ROV; Framework.

# Table of contents

# List of figures

## List of tables

## List of Abreviations

AI – Artificial Intelligence
UUV – Unmanned Underwater Vehicle
ROV – Remotely Operated Vehicle
AUV – Autonomous Underwater Vehicle
DRQN – Deep Recurrent Q-Network
RL – Reinforcement Learning
MDP – Markov Decision Process
dAE – Dense Autoencoder
cAE – Convolutional Autoencoder
LSTM – Long Short-Term Memory
DRL – Deep Reinforcement Learning
CI – Confidence Interval
BBOX – Bounding Box
LD – Latent Dimension
VIP – Visual Inspection of Pipelines

# 1
# Introduction

Currently, the demand for services in the area of automated inspection (1) (2) (3) (4) (5) (6) (7) (8) (9) is evident. These autonomous intelligent inspection systems are being widely used to make inspections faster, more effective and safer as they do not require a human in hazardous environments.

In aquatic environments, the traditional use of divers or manned underwater vehicles has been replaced by unmanned underwater vehicles (such as ROVs or AUVs). While ROVs (Remotely Operated Underwater Vehicles) need an operator to control them remotely, AUVs (Autonomous Underwater Vehicles) have the ability to explore deep waters by following pre-programmed trajectories, while also having the capacity to make adaptive decisions during the mission, showcasing a remarkable degree of independence and self-sufficiency.

Inspections executed by ROVs are usually recorded for later analysis by specialists, because despite having the possibility of transmitting data in real time, ROVs have a more limited operation compared to AUVs, since they have a lower speed, mobility and space range.

Unmanned underwater vehicles (UUVs) have advantages in terms of reducing safety risks, such as exposure to pressure, temperature or shortness of breath. They also have the ability to access areas of extreme depth that were not possible for humans until then. Furthermore, the costs involved in inspections using UUVs are lower than manned underwater vehicles, exceeding them in the number of underwater operations.

These vehicles are of great importance in various industries and in environmental monitoring. In Brazil, where the primary source of oil extraction is from pre-salt reservoirs, these vehicles are indispensable to the oil industry, because they enable the inspection of equipment, marine life, and environmental impacts associated with offshore operations.

Some vehicles have a disadvantage, they have no autonomy to make decisions and often their routes are already planned (10) (11) (12). This is an impediment when the vehicle identifies events of interest and is unable to capture necessary images or focus at that point without the order of a human controller. This results in the repetition of an inspection or the loss of the event of interest. The cost involved in running these vehicles is quite high, so the industry is looking for solutions to this problem, in order to be able to make these decisions in real time.

Furthermore, most of the work presented for automating the perception

of the environment for automata is done through sensors of different types and GPS (13) (14) (15). Due to the complexity of the navigation environment, different control and automation algorithms have been tested in this area. In this work, the interest is that the automaton decision is taken through the analysis of visual events. Only a small part of the surveyed literature uses images to automate and make decisions about the routes (16) (17) (18), because of the complexity of the environments and the time it can take to process the images for decision-making.

Artificial intelligence (AI) in recent years has developed algorithms that allow autonomy. An example is the self-driving car, which is at a very advanced level of development and has allowed for an important advance in this area (19). These AI algorithms can be used in the area of underwater vehicles in order to give some degree of autonomy, so as to make real-time decisions, such as if the vehicle has observed events of interest and needs to change its route. This would make it possible to optimize the time and costs of these vehicles.

The autonomous inspection task has several challenges: detecting the events, processing the images and making the decision to change the route in real time. It is a highly complex task and needs multiple algorithms working together to perform well. Artificial intelligence presents several algorithms for autonomy, such as those based on reinforcement learning and several others in the area of image detection and classification.

One example is anomaly detection task, which is already well-known in various fields, such as maintenance and monitoring (20)(21), process automation (22), video surveillance (23) (24), defect detection (25), supply chain (26), time series (27) and others. However, as previously mentioned, the focus of this study lies in underwater inspections, which poses additional challenges since the underwater images usually have a very similar appearance due to low lighting and the uniformity of the ground. Sunlight decreases significantly as it penetrates the water, and most colors are absorbed at lower depths. As a result, underwater images often have a blue or greenish tone, can be turbid, and colors are generally less vibrant than on the surface (28), making it difficult to discern objects in the underwater environment.

Another vital aspect in AUV operations is viewpoint planning, a crucial task that enables AUVs to optimize their course and position in order to achieve optimal perspectives of targets of interest. By meticulously planning the viewpoints from which the AUV captures data, it can ensure the acquisition of high-quality images and information, facilitating accurate and detailed analysis, allowing the agent to perform inspection tasks faster and more accurately, saving valuable time and resources for the petroleum industry (29)

(30) (31) (32).

Viewpoint planning involves a combination of advanced algorithms and intelligent decision-making processes. AUVs utilize their onboard sensors, such as sonar, cameras, and other specialized equipment, to detect and identify potential targets. The data collected from these sensors is then analyzed to determine the optimal viewpoints from which the AUV should capture images or gather specific information (33). Careful viewpoint selection is critical for many computer vision and robotic tasks (34) (35) (36) (37) (38).

## 1.1 Problem Definition

One of the most important inspections is carried out for the decommissioning of oil platforms. Abandonment or decommissioning occurs when the platform, after a certain period, reaches its final stage of production. This can happen when the production of oil and gas is disadvantageous, resulting in the closure of activities, cleaning and removal of structures and environmental recovery of the site, which are operations with a high cost and high potential for generating environmental impacts.

Brazil is the ninth largest oil producer in the world and its main reserves are offshore, which is why inspection of equipment and the environment by AUV is essential. In the case of offshore platforms, some ecosystems are created on top of or close to underwater structures. When this occurs, the structures will be partially removed, as the ANP (National Agency of Petroleum, Natural Gas and Biofuels) does not allow the removal in case there is marine life around it. Therefore, inspections to assess the presence of marine life in the decommissioning process of offshore platforms become highly cautious procedures. It is necessary to conduct a careful assessment of the area, identifying and documenting the species present, as well as evaluating the potential impact of removing the structures on these marine communities. All of this requires advanced techniques, such as the use of Autonomous Underwater Vehicles (AUVs), capable of collecting precise and detailed data in complex underwater environments.

Currently, decommissioning inspections are recorded to enable later analysis by experts, such as biologists, in order to identify potential impacts on wildlife and flora during equipment removal. However, these videos often have a duration of over 48 hours, leading to significant fatigue for the experts who analyze them. This prolonged viewing period can, in some cases, result in errors due to human fatigue. Given the sensitive nature of this operation, it is imperative that such errors be minimized or avoided.

## 1.2 Proposed Solution: An Novel Autonomous Inspection Framework

The use of a camera instead of GPS and sensors for AUV (Autonomous Underwater Vehicle) control has been the subject of recent research due to its significant cost reduction. A camera has a considerably lower cost compared to other devices. Studies, as exemplified by (39) (40) (41) (42) (43), investigate the use of images captured by cameras or sonars, employing detection or segmentation methods to guide and plan the trajectory of the AUV. Thus, this study also delves into the field of computer vision as a means of controlling the AUV.

This work consists of an autonomous inspection framework that eliminates the need for experts to monitor videos for extended periods. However, it is crucial to emphasize that human interventions are still required in specific operations, such as readjusting the SVM model after an inspection, or in the event that the AUV does not properly identify the pipeline. This characterizes a human-in-the-loop approach, where human involvement plays an essential role in optimizing and correcting the system.

This system is also capable of automatically capturing the most relevant angles of the objects under analysis, significantly reducing the likelihood of needing to do a new inspection due to the loss of some important detail. Furthermore, upon completing the inspection, the system automatically generates a detailed report, providing valuable support to the experts involved in the process.

This VIP framework (Figure 1.1) is capable of performing underwater inspections just by analyzing images from the AUV camera, using deep learning, reinforcement learning and novelty detection. However, this framework can be adapted to many other inspection tasks. The VIP framework consists of steps that allow an AUV an autonomy capable of detecting objects of interest and tracing optimized routes that make it possible to record images of these objects for their classification.

The Figure 1.1 presents the simplified framework that will be presented in this study. As can be seen, the automaton first performs a step, which is the equivalent of following the duct to be inspected, then it checks if the image obtained by the automaton camera contains any anomalies. In case an anomaly is found, the automaton will have to approach the object in the best way, in order to capture the best images of the anomaly. Finally, if the inspected object is considered unknown, the SVM model will be retrained with the collected images from it, and return to the saved current position. At the end of the inspection, a report is generated, containing relevant information about the

objects found, such as their coordinates, for example.

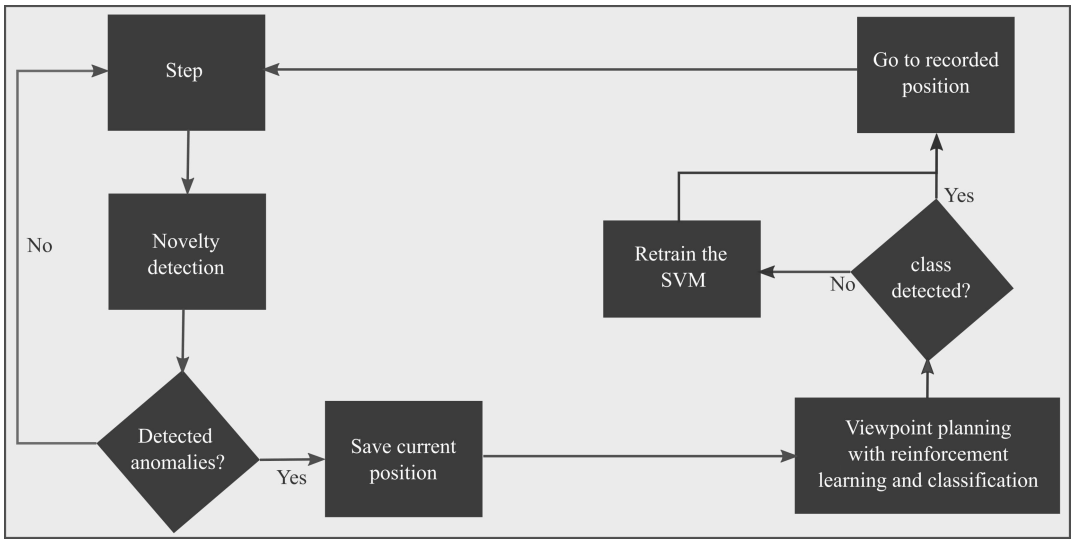


Figure 1.1: Simplified Visual Inspection of Pipelines (VIP) framework diagram

This study contributes to the creation of a VIP framework responsible for controlling an AUV during a pipeline inspection while simultaneously searching for anomalies and learning to classify them using only images from the autonomous camera. To achieve this goal, the study incorporates a novelty detection model, allowing the system to quickly and accurately identify novelties in real-time. Additionally, the study also integrates a dedicated model to reduce inspection time in viewpoint planning, resulting in time savings for the experts involved in the process. This innovative approach provides a more efficient inspection and more accurate detection of anomalies in underwater pipelines.

It is important to emphasize that in this work, the focus is on Artificial Intelligence models, rather than control algorithms, which are addressed in a relatively simplified manner in this study.

## 1.3 Thesis Organization

This thesis will be divided into chapters as follows:

Chapter 2: Theoretical background – introduces concepts about Deep Learning, Deep Reinforcement Learning, Novelty detection, Viewpoint planning and other relevant subjects to this study.

Chapter 3: Methodology - explains the system proposed for this thesis.

Chapter 4: Experiments - introduces the environments used in the tests and explains each experiment performed.

Chapter 5: Results - reveals the results of tests performed in the previous chapter.

Chapter 6: Conclusions - presents conclusions drawn after observing the results of the experiments and shows where the solutions of this work will be applied in future work.

# 2
# Theoretical background

## 2.1
## Deep learning

In this work neural network models and deep learning are used in some learning tasks, in the following sections the models and techniques based on neural networks used in this project will be introduced. Neural networks are models inspired by biological brains. Its components simulate the interactions between axons and dendrites, which emit and receive nerve impulses, respectively. Neural networks have shown good results when used in very complex problems, such as: segmentation of objects in images or videos, description of scenes or semantic extraction of words.

### 2.1.1
### Convolutional Neural Network

As stated by Mohammad Mustafa Taye in (44), convolutional neural networks (CNNs) are one of the main types of neural networks used for image recognition and classification. CNNs serve various purposes, including the identification of objects, image manipulation, computer vision tasks, and facial recognition. These networks take image data as input, and their distinctive capability lies in autonomously acquiring a hierarchy of features, subsequently employed for classification, eliminating the need for manual feature engineering.

A convolution is a mathematical operation between two functions $f$ and $h$, which produces a third function seen as a modified version of $f$ as a function of $h$. In this work, $f$ is delimited as a sequence of vectors and $h$ is a linear filter that makes each new element of the output a weighted sum of the elements in the context of each element processed in the sequence. Thus, in the input sequence (or in the previous convolution outputs) a set of linear filters with the same weights for the entire sequence is applied, in the case of convolutional networks, learned by back-propagation. This achieves certain properties in the output sequence, one of the most important being translation invariance.

When there is an input $f$ and a filter $h$, each element $(i, j)$ of the output $g$ of a convolution is defined as $g(i, j) = \sum_{k,l} f(ik, jl)h(k, l)$, where $k$ and $l$ are the height and width of the filter respectively. Figure 2.1 shows an example of this operation.

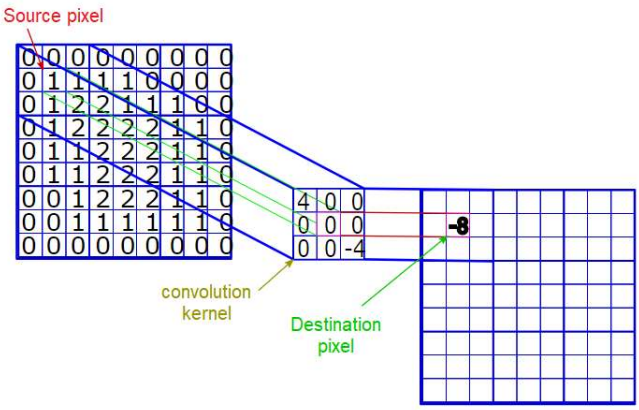


Figure 2.1: 2D convolution

Regarding the first advantage, the interaction parameters between each input and output are shared, causing there to be more interactions than parameters. As a result, memory requirements decrease and correlations between neighboring pixels are learned.

CNNs offer several significant scientific advantages in the field of computer vision and image processing. One key advantage is their ability to automatically learn hierarchical representations from raw data, such as images. CNNs employ a series of convolutional layers that apply filters to extract features like edges, textures, and shapes, progressively learning more abstract features in deeper layers. This mimics the human visual system's hierarchical processing, allowing CNNs to capture complex patterns efficiently.

### 2.1.2 Fully Connected Layers (FC)

FC layers work like a traditional neural network and contain approximately 90% of the network parameters (45). The 1D output vector of the network is usually of predefined length. For example, for an image classification task, it will be of length equal to the number of categories (classes) that are taken.

As this type of network handles a large number of parameters, it is very expensive to train them, since they require a large computational load to train them. For this reason, many authors defend reducing the connections between the neurons of these layers using some kind of method, as in the case of GoogleLeNet (46), reducing their number, or even eliminating them (47).

The most complex part when working with this type of architecture is training. If it is decided to use a supervised strategy, both in feedforward and backpropagation, computers with high computational capacity will be required.

### 2.1.3 Loss function

In machine learning, the acquired knowledge is evaluated through loss functions. If the predictions deviate too much from the actual results, the loss function will return a very large number. The optimization function (48) uses this loss to adjust the model's parameters during the training process, reducing the error and improving the model's performance.

In general, loss functions can be classified into two main categories depending on the type of learning task we are dealing with: Regression Losses and Classification Losses. In this work, more emphasis will be given to classification losses, which are introduced below.

#### 2.1.3.1 MSE

The Mean Squared Error (MSE) is a widely used metric to quantify the dissimilarity between two sets of values. The MSE is calculated as the average of the squared differences between the two sets (2-1).

$$\text{MSE}(y, \hat{y}) = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N} \tag{2-1}$$

Where:

- $N$ is the total number of data points or observations
- $y_i$ represents the actual (ground truth) values
- $\hat{y}_i$ represents the predicted values.

A lower MSE value indicates better similarity between the two sets of values, with a value of 0 indicating perfect similarity.

#### 2.1.3.2 SSIM

The Structural Similarity Index (SSIM) is a more comprehensive metric that not only considers pixel-wise differences but also takes into account structural information, luminance, and contrast. SSIM measures the structural similarity between two images and is defined as 2-2.

$$\text{SSIM}(y, \hat{y}) = \frac{(2\mu_y\mu_{\hat{y}} + C_1)(2\sigma_{y\hat{y}} + C_2)}{(\mu_y^2 + \mu_{\hat{y}}^2 + C_1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + C_2)} \tag{2-2}$$

Where:

- $\mu_y$ and $\mu_{\hat{y}}$ are the means of the sets of values $y$ and $\hat{y}$ respectively.

- $\sigma_y$ and $\sigma_{\hat{y}}$ are the standard deviations of values in sets $y$ and $\hat{y}$ respectively.
- $\sigma_{y\hat{y}}$ is the covariance between the two sets of values.
- $C_1$ and $C_2$ are constants added for stability, typically $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$, where $L$ is the dynamic range of values, and $k_1$ and $k_2$ are constants.

A higher SSIM value indicates better similarity between the two images, with 1 indicating perfect similarity.

### 2.1.4 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) is an algorithm based on the work of David Rumelhart in 1986 (49). RNNs are known for their ability to process sequential data, recognize patterns, and predict the end result. Therefore, video analysis, image subtitling, natural language processing and music analysis depend on the capabilities of recurrent neural networks. Unlike other artificial neural networks, which assume independence between input data, RNNs actively capture their sequential and temporal dependencies.

The RNN operates in a recurrent manner, where at each time step, it takes an input data point and combines it with the internal memory/state from the previous time step. This combination is passed through a set of mathematical operations, which include weights and activation functions, to produce an output and update the internal state.

The key idea is that the internal state of the RNN serves as a kind of memory, storing information about previous time steps. This allows the RNN to capture patterns and dependencies in sequential data, making it suitable for tasks like natural language processing, speech recognition, and time series forecasting.

#### 2.1.4.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) used in machine learning and deep learning. LSTMs are designed to process and make predictions based on sequences of data, such as time series or natural language text.

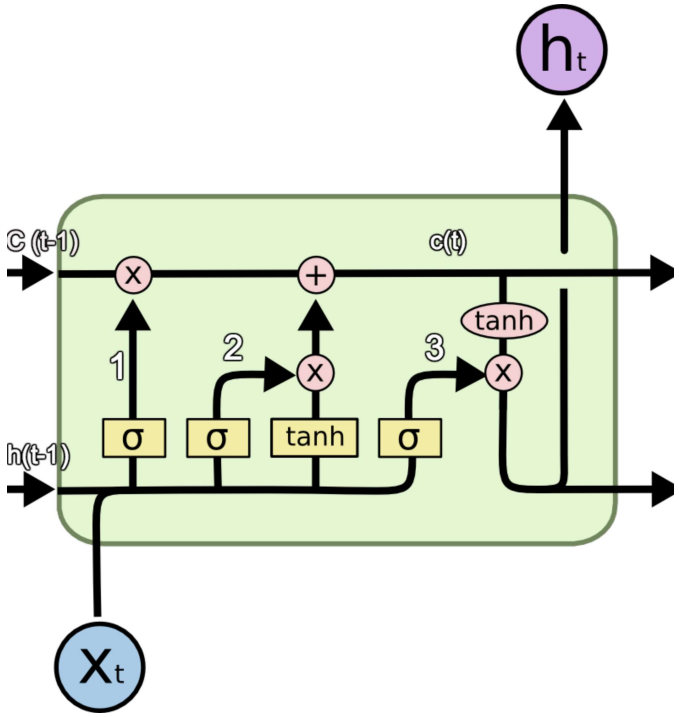


Figure 2.2: LSTM cell

LSTMs work by maintaining a memory cell that can store information over long sequences. This memory cell (Figure 2.2) has some key components: an input gate, a forget gate, and an output gate. These gates control the flow of information in and out of the cell. They are described below:

- 1 - Forget Gate: The forget gate decides which information from the previous memory cell state should be forgotten or remembered;
- 2 - Input Gate: The input gate decides which information from the current input should be stored in the memory cell;
- Update Memory Cell: Using the input gate, the LSTM updates the memory cell by adding new information to it, and using the forget gate it decides which old information to discard;
- 3 - Output Gate: The output gate decides what information from the updated memory cell should be used to generate the output at the current time step. It takes the current input and the previous hidden state as input and calculates the output value.

LSTMs offer several advantages in sequential data processing. Their ability to capture and retain long-term dependencies makes them well-suited for tasks where context from distant past inputs is critical, such as natural language processing and speech recognition. Their versatility allows them to handle various types of sequential data, from text to time series, making

them a valuable choice for a wide range of machine learning applications. Furthermore, LSTMs are robust in managing sequences of varying lengths, providing flexibility and reliability in modeling sequential data.

## 2.2 Novelty detection

Novelty detection or anomaly detection is the detection of rare items, events, or observations that arouse suspicion because they are different from most data (50). By establishing a baseline of normal behavior, novelty detection algorithms can help detect deviations or irregularities that might indicate fraud, faults, or unusual events. Designing a novelty detector presents an immensely intricate challenge, primarily because of the inherent unpredictability of novelties and their unavailability for training purposes. These factors underscore the fundamentally unsupervised nature of the problem at hand.

Novelty detection requires something that records an event, something like a memory or, more recently, represented in deep autoencoders. In general, the novelty methods build some model of a training set that is selected to contain no examples (or very few) of the important (i.e., novel) class (51) (52).

There are several applications in the anomaly detection area, such as video surveillance (53) (54) (55) (56), video inspections (57) (58) (59) (60), medical images (61) (62) (63) (64) (65) and fraud detection (66) (67) (68) (69) (70). In this study, as well as in some of the previously mentioned references that address video inspections and fraud detection, we will employ the autoencoder model.

### 2.2.1 Autoencoder

In the case of this study, an autoencoder was used. An autoencoder is a neural network architecture class that aims to learn how to compress/reduce a data (step known as an encoder) and then learn to reconstruct the data from the version that was previously reduced (step known as a decoder). It is expected that the reconstructed data will suffer some (preferably minimal) information loss during the process (measured by the reconstruction loss). There are some types of autoencoder (Figure 2.3), in this work we use two of them: Dense Autoencoder (dAE) and Convolutional Autoencoder (cAE).

- **dAE:** It is an autoencoder that contains dense and convolutional layers. Due to the presence of dense layers, the size of the representation generated by the encoder depends solely on the number of neurons used in the last dense layer of the encoder;

– **cAE:** It is an autoencoder network consisting only of convolutional layers, which results in the size of the representation generated by the encoder being equivalent to the $width * height * dimension$ of the output from the last convolution.

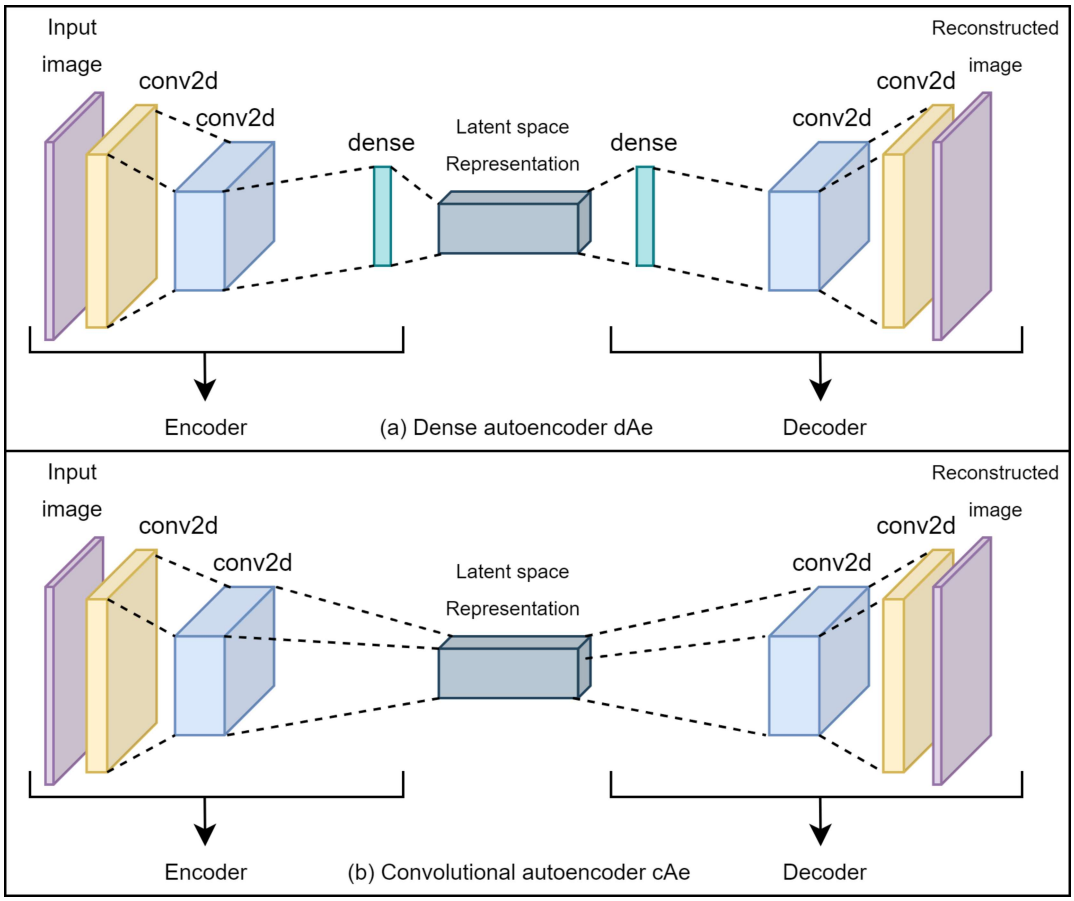


Figure 2.3: An overview of different autoencoder frameworks: Dense autoencoder (a) and Convolutional autoencoder (b).

## 2.3 Reinforcement learning

Reinforcement learning (RL) (71) is an area within machine learning dedicated to the development of algorithms that allow an agent to achieve a goal, maximizing an accumulated reward value. Some examples of RL algorithms include Q-Learning (72), which estimates action quality in discrete action spaces; Deep Q-Networks (DQN) (73), which handles high-dimensional state spaces using deep neural networks; and Deep Deterministic Policy Gradients (DDPG) (74) for continuous action spaces.

There are a few things to define in reinforcement learning that are fundamental to the subject. First, there is a concept of reward. This is what differentiates reinforcement learning algorithms from other types of machine

learning algorithms. An agent will try to maximize not only its immediate reward, but future rewards as well. Reinforcement learning algorithms have often found new ways to accomplish this.

In RL, an agent can be an instance of different types, for example a robot or simply a system. There is a large number of practical applications for these types of problems, where the main advantage is that it does not require an expert to find the solution to the problem, but simply the problem must be properly formulated, specifying the rewards or penalties for the agent to learn it.

The purpose of an agent during reinforcement learning is to find a strategy that leads to choosing the best action and obtaining the highest cumulative reward expected in any state. An agent is considered to have learned an optimal strategy, usually called optimal policy, when it is able to accumulate the highest possible reward for the specified task. Often, finding a policy close to optimal is enough to solve the task properly.

### 2.3.1 Markov decision process (MDPs)

Markov Decision Processes (MDPs) are the most used models to determine the reinforcement learning problems. MDPs can be considered as a type of sequential decision process, where the Markov properties are fulfilled. (75).

The MDP is defined through the tuple $< S, A, f, R >$ , following a notation similar to that presented by Sutton and Barto (76), where $S$ is a set of states, where $s_t \in S$ the state in which the agent is at the moment $t$, $A$ is the set of actions that the agent can perform when it is in the state $s_t$, where $a_t \in A(s_t)$ a action that the agent processes at the instant $t$ when it was in the state $s_t$, $f$ is a transition function and finally, $R$ is a reward function.

The agent is tasked with finding the policy $\pi : S \times A \rightarrow [0, 1]$, where $\pi_t(s, a)$ indicates the probability that the agent selects the action $a$ in the $s$ state, in order to maximize the reinforcement parameter.

#### 2.3.1.1 Markov property

A stochastic process is said to have the Markov property if the conditional probability of the next state of the process, $s_{t+1}$, depends only on the state in which the agent was positioned, $s_t$, and of the action performed in this state in $a_t$.

A finite MDP can be defined by its set of actions, states and state transition functions and reinforcements, so according to the Markov property,

we can obtain the state transition function, as shown in the equation 2-3:

$$f(s, a, s') = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{2-3}$$

where $Pr$ is the probability distribution over the next state of the system given the current state and the action taken by the agent.

It is verified that $\forall s \in S, \forall a \in A, \sum_{s_i \in S} f(s, a, s_i) = 1$, since the function $f$ determines a distribution of probabilities.

In the equation 2-4, the reward function is displayed, in which $E$ defines an expectation about the amount to be received.

$$R(s, a) = E\{r_{t+1} | s_t = s, a_t = a\} \tag{2-4}$$

### 2.3.2 Policies

The policy $\pi$ is used for the agent to choose the action to perform in each state. When an agent follows a policy $\pi$, it means that, at any instant $k$ and state $s_k$, the action performed will be given by $a_k = \pi(s_k)$.

If the policies perform a mapping between states and distributions in the action space, $\pi : S \times A \rightarrow [0, 1]$, they will be stochastic, respecting the following conditions for each state $s \in S$ (equations 2-5 and 2-6).

$$\pi(s, a) \geq 0 \tag{2-5}$$

$$\sum_{a \in A} \pi(s, a) = 1 \tag{2-6}$$

The policy is part of the agent, and in general, the goal of RL algorithms is to find an optimal policy (77). An example of how a policy works can be given as follows: assuming an MDP where an initial state distribution, $I : S \rightarrow [0, 1]$, defines the probability that the system starts in the $states_0$, the policy $\pi$ is then used, so the action performed by the agent will be $a_0 = \pi(s_0)$ and as a consequence of this action, according to the transition function $f$, a transition to the state $s_1 = f(s_0, a_0)$ and then the agent will receive a reward $r_0 = R(s, a, s_0)$.

The process described above will be repeated, generating the trajectory $s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, ....$ In the case of an episodic problem, the trajectory will end in a terminal state $s_{term} \in S_{term}$, where $S_{term} \subseteq S$, and then a new sequence will start with a state determined by $I$. If the problem is continuous, the trajectory can be extended indefinitely.

### 2.3.3 Value function and Bellman equations

Reinforcement learning algorithms mostly aim to try to obtain the $\pi$ action policy by approximating the value functions. Such functions evaluate the agent's benefit of being in a state or the agent's benefit of performing an action from a state. Therefore, there are two types of value function, the *state-action value function* $Q$ and the *state-value function* $V$.

The function $Q$ of a policy $\pi$ , $Q^\pi : S \times A \to \mathbb{R}$ is equal to the return obtained when, from a state $s$ and the action $a$, follows the policy $\pi$ (equation 2-7)

$$Q^\pi(s,a) = \sum_{k=0}^{\infty} \gamma^k R(s_k, a_k, s_k 0) \tag{2-7}$$

Another expression is the optimal policy. It is one that in each state selects the action with the highest value of the optimal $Q$ function, maximizing the return obtained (equation 2-8)

$$\pi^*(s) = argmax_a Q^*(s,a) \tag{2-8}$$

where $argmax_a g(a)$ for any function $g(.)$ returns the value of $a$ that gives the maximum of the function $g(a)$.

For any given function $Q$, a policy $\pi$ satisfying the equation 2-9

$$\pi(s) = argmax_a Q(s,a) \tag{2-9}$$

is considered greedy about $Q$. Therefore, a possible way to find the optimal policy is to first calculate $Q^*$ and then apply $\pi^*(s) = argmax_a Q^*(s,a)$ to get a greedy policy towards $Q^*$.

Through the Bellman optimality equation, the optimal value functions $Q^*$ can also be characterized recursively. The aforementioned equation determines that the optimal value of performing an action $a$ in a state $s$ is equal to the sum of the immediate reward obtained plus the value of $Q^*$, which was obtained by the optimal action in the subsequent state (equation 2-10)

$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} Q^*(f(s,a), a_0) \tag{2-10}$$

### 2.3.4 Deep Q-Network (DQN)

Q-learning (72) is a reinforcement learning algorithm used to find the optimal action-selection policy for a Markov decision process. The Q-learning update rule is used to iteratively update the Q-values, which represent the

expected cumulative rewards of taking a particular action in a particular state. The update rule is as follows (equation 2-11):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left( R + \gamma \cdot \max_{a_0} Q(s_0, a_0) - Q(s,a) \right) \tag{2-11}$$

In this equation:

- $Q(s,a)$ represents the Q-value for taking action $a$ in state $s$.
- $\alpha$ is the learning rate, which controls the step size of the updates. It's a value between 0 and 1.
- $R$ is the immediate reward received after taking action $a$ in state $s$.
- $\gamma$ is the discount factor, which represents how much importance is given to future rewards. It's also a value between 0 and 1.
- $s_0$ is the next state that the agent transitions to after taking action $a$ in state $s$.
- $a_0$ represents the action that maximizes the Q-value in the next state $s_0$.

This method makes it possible to represent, in a very precise way, the value of the long-term rewards that can be obtained in each state for each of the available actions. However, this representation also implies a very important problem: increasing the complexity of the state also increases the size of the Q table, but exponentially.

The objective is to create a model that can handle a large $Q$ table while fitting within memory constraints. This approach involves using a neural network to approximate the $Q$ function, a technique within Deep Reinforcement Learning (DRL) which combines principles from reinforcement learning and deep learning. A notable application of this method was in the 2013 project "Playing Atari with Deep Reinforcement Learning" by the DeepMind team (73), where they employed Deep Q-Networks (DQN).

DQN uses experience buffer to improve the stability and efficiency of training. This method consists of storing tuples of the form $< s, a, r, s_0 >$ in replay memory (a memory that stores past transitions), where $s$ is the state current, $a$ action taken, $r$ reward obtained and $s_0$ next state; and during the training of the network, a set of tuples selected randomly among all stored ones. That way, the network could train using samples in a different order in which they occurred.

The algorithm in Algorithm 1 operates as follows:

**Algorithm 1** Deep Q-Learning algorithm with experience replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
Create $\phi$ function to resize and normalize images
**for** episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocess sequence $\phi_1 = \phi(s_1)$
  **for** $t = 1$, T **do**
    With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t = argmax_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in simulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, x_{t+1}$ and Preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions
    $B = \{(\phi_j, a_j, r_j, \phi'_j)...(\phi_{j+\tau}, a_{j+\tau}, r_{j+\tau}, \phi'_{j+\tau})\}_{j=1}^{\text{batch size}=\tau} \subseteq \mathcal{D}$
    **for** each sequence $\in B$ **do**
      Set $y_j = r_j$ for terminal state
      Set $y_j = r_j + \gamma max_{a'} Q(\phi'_j, a'_j; \theta)$ for non-terminal state
      Calculate $\mathcal{L}_j = (y_j - Q(\phi_j, a_j; \theta))^2$
      Perform a gradient descent step on $\mathcal{L}_j$
    **end for**
  **end for**
**end for**

### 2.3.5 Deep Recurrent Q-Network (DRQN)

One of the biggest problems faced by a DQN (section 2.3.4) is the partial knowledge of the environment (78). A DQN works with a Markov decision process (MDP) (79), which is a good formalism, but in the field that concerns this project, it is not possible to implement this pattern. In the environments considered here, there is missing information for optimal decision-making. For example, the AUV when looking at a single image, may not be able to make the best decision.

The proposed solution to this problem is to give the agent the ability to temporally integrate the observations (78). The intuition behind this is: if information from a single moment is not enough to make a good decision, then acquiring enough temporal information is probably the best option.

Within the context of Reinforcement Learning, there are different ways of achieving temporal integration. One of the solutions is proposed in (80),

where instead of giving the neural network a single input image, an external buffer is created in which the 4 previous images that were used are kept.

Some articles (73) (80) suggest that better performance can be obtained if more than the last 4 frames are used, but the experience buffer used consumes much more memory for the groups of 4 or more images, and also, it requires more weights, leading to slower training and possible overfitting. To deal with these issues more robustly, the use of recurrent neural networks (RNN) is proposed (Algorithm 2).

An RNN has the particularity that the connections between its nodes form a directed graph based on a temporal sequence. This allows it to display dynamic temporal behavior and learn temporal dependencies. Unlike other networks, an RNN can use its internal state (memory) to process input sequences. The characteristic recurrent block of an RNN can be integrated into the DQN architecture. By doing this, the resulting agent can receive unique images of the environment and the network will change the output depending on the temporal pattern of observations it receives. This is achieved by maintaining a hidden state that is computed every cycle. The recurring block can be provided with the hidden state, acting as an enhancement that informs the network what happened before. Agents of this type are called Deep Recurrent Q-Networks (DRQN) (78). In this study, the RNN chosen to be used was the LSTM described in 2.1.4.1.

**Algorithm 2** Deep Recurrent Q-learning algorithm with experience replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
Initialize action-value function parameters $\theta_0$ with random weights
Set target action-value function parameters $\theta^- \leftarrow \theta_0$
Create $\phi$ function to preprocess images with SVM
**for** episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocess sequence $\phi_1 = \phi(s_1)$
  **for** $t = 1$, T **do**
    With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t = max_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in simulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, x_{t+1}$ and Preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    **if** i % freeze interval == 0 **then**
      $\theta^- \leftarrow \theta$
    **end if**
    Sample random minibatch of sequences
    $B = \{(\phi_j, a_j, r_j, \phi'_j)...(\phi_{j+\tau}, a_{j+\tau}, r_{j+\tau}, \phi'_{j+\tau})\}_{j=1}^{\text{batch size}=\tau} \subseteq \mathcal{D}$
    **for** each sequence $\in B$ **do**
      $h_{j-1} \leftarrow 0$
      **for** $k = j...j + \tau$ **do**
        Update the hidden state $h_k = Q(\phi_k, a_k, h_{k-1}; \theta_i)$
      **end for**
      Set $y_j = r_j$ for terminal state
      Set $y_j = r_j + \gamma max_{a'} Q(\phi'_j, a'_j, h_j; \theta)$ for non-terminal state
      Calculate $\mathcal{L}_j = (y_j - Q(\phi_j, a_j, h_{j-1}; \theta))^2$
      Perform a gradient descent step on $\mathcal{L}_j$
    **end for**
  **end for**
**end for**

DRQN has been successfully applied in various domains. In video games, it has been used to train agents that achieve state-of-the-art performance in games like Atari 2600 (78). In robotics, DRQN has been employed for tasks such as robot navigation (81) and grasping objects (82). It has also been used in financial applications, such as stock trading (83) (84), where the sequential nature of financial data makes DRQN a suitable choice for modeling and decision-making.

## 2.4 Viewpoint planning

In the oil industry context, viewpoint planning in Autonomous Undersea Vehicles (AUVs) plays a crucial role in monitoring and detecting information during underwater inspections. Viewpoint planning allows AUVs to optimize their course and position to obtain optimal perspectives of targets of interest, such as subsea pipelines, drilling rigs, or underwater structures. By carefully planning the viewpoints, the AUV can capture high-quality images and data, ensuring accurate and detailed analysis. In addition, viewpoint planning also increases the efficiency of operations, allowing the AUV to perform inspection tasks faster and more accurately, saving valuable time and resources for the petroleum industry (85) (32) (31) (30).

Robot navigation is an extremely important technology for performing various tasks. Navigation is generally performed while robots move around and estimate their positions and orientations using information from various sensors and/or cameras.

For tasks containing 3D objects, creating a plan from multiple viewpoints at runtime is essential when there is no prior information about them or their environments. The goal of viewpoint planning is to get better images of the object and acquire knowledge about the invisible parts of it. At the same time, the agent is positioned in the best way, obtaining the best focus, line of sight, and collision avoidance.

In the case of recognition and inspection of 3D objects (86) (87) (88), a good image facilitates this task. For this, several views of the object are usually required. Typically, rather simplistic methods or assumptions (e.g., steps with fixed angles) are made when selecting viewpoints. The benefits include improved quality and efficiency if a priori or online views can be determined.

For a better view planning, some studies use Reinforcement Learning algorithms (88), others only image classification (89) or object detection (86). In this research, was trained an SVM to classify objects and use its output as a feedback mechanism during the deep reinforcement learning process.

## 2.5 Support vector machine (SVM)

Support vector machine (SVM) (90) is a popular supervised learning algorithm for classification and regression analysis. SVM works by creating a hyperplane that separates the data into different classes. The goal is to find the hyperplane that maximizes the margin, the distance between the hyperplane

and the nearest data points from each class. SVM is effective in handling high-dimensional data. It can handle non-linear data by using kernel functions to transform it into a higher-dimensional space where they can be linearly separable. SVM is widely used in diverse areas, including image recognition, text classification, and bioinformatics.

The SVM output for classification is a decision boundary that maximizes the margin between two classes in a feature space. It identifies a hyperplane that best separates the data points into distinct categories, with the goal of minimizing classification error. The output consists of the equation of this hyperplane, for example, for the linear kernel we have: $w * x + b = 0$, where $w$ is a weight vector, $x$ is the input data, and $b$ is a bias term. To classify new data points, SVM assigns them to one of the two classes based on which side of the hyperplane they fall. This decision boundary is determined by the support vectors, which are the data points closest to the hyperplane, ensuring robustness and generalization of the classification.

There are several advantages in using SVM as a supervised learning algorithm, as demonstrated by Pin Wang et al. in their comparative study (91): it is well-known that traditional machine learning achieves better results on small sample datasets, while deep learning structures achieve higher recognition accuracy on large datasets. Therefore, in this study, which does not have a large dataset, the SVM approach proves to be advantageous.

To train an SVM, you can use the "one-vs-one" and "one-vs-rest" strategies. The SVM's "one-vs-one" strategy trains a binary classifier for each unique pair of classes in order to solve multiclass classification issues. All binary classifiers' votes are tallied during classification, and the class with the most votes is selected as the output class. The "one-vs-rest" approach, on the other hand, entails training a binary classifier for each class, treating the first one as the positive class and the rest as the negative class. The example is categorized by classifying it into the one whose binary classifier yields the highest confidence score. Both strategies are frequently used in SVM to address multiclass classification issues, and each has pros and cons of its own. The "one-vs-rest" approach was used in this work.

## 2.6 Image tracking

In recent years, image tracking technology has gained a prominent role in various fields. Image tracking refers to the ability to locate and track specific objects or patterns in an image or sequence of images in real time. This technology has revolutionized the way we interact with the digital world and

is opening up new possibilities in a variety of applications.

Image tracking is based on computer vision and image processing techniques. Its systems use advanced algorithms to identify specific reference points or patterns in an image. These reference points can be unique features such as edges, colors, textures, or even pre-defined markers. Based on these points, the system is able to track the position and movement of the object of interest as the camera captures new images.

In this study, we employed the Median Flow tracker (92). This algorithm is renowned for its speed and effectiveness in scenarios where the moving object maintains a relatively constant appearance and lighting variations are not excessive — characteristics that align with the conditions present in our application. Here, tracking is exclusively utilized when the object is in close proximity to the agent, mitigating any concerns regarding significant lighting changes. Furthermore, given the agent's slow movement, changes in the object from one frame to another are minimized, contributing to even more stable results.

## 2.7 Proportional control

Proportional control is a fundamental and widely used technique in the field of control engineering for dynamic systems. Playing a crucial role in the stability and accuracy of these systems, it allows for behavior adjustment according to specific needs. This work explores the concept of proportional control, highlighting the benefits it offers.

Proportional control is a control strategy that adjusts the action proportionally to the error between the desired value and the actual value of the system. The larger the error, the greater the applied correction. This proportion is determined by the proportional gain.

There are several advantages associated with proportional control. Firstly, it contributes to the system's stability since the control action is directly proportional to the error. This avoids excessive oscillations and nonlinear responses, promoting more stable behavior.

Furthermore, proportional control enables greater precision, allowing for a quick and accurate response of the system to external variations. By adjusting the control action according to the error, it ensures efficient adaptation to real-time conditions.

Another significant advantage is the simplicity of proportional control compared to other control techniques. Its implementation and understanding are relatively straightforward, requiring fewer adjustments and being less prone

to instability issues. This facilitates its application in a variety of systems and makes it more accessible.

In summary, proportional control is a valuable strategy for controlling dynamic systems, offering stability, precision, and simplicity. Its widespread use highlights its effectiveness and importance in the field of control engineering.

# 3
# Methodology

The number of inspections carried out by autonomous robots has been increasing due to the development of technology. As a result, there is an increase in studies focused on the area of autonomous inspections, whether underwater or not.

Some of these studies only use sensors like LiDAR (15) or other sensors (5) to control the automata during the inspection. Other studies use, in addition to sensors, cameras installed in robots (93) (94). Some just follow pre-selected routes, reporting only if there is a problem along the route (95). And some studies prefer to focus only on image processing, not controlling the automaton (96).

Unlike the studies mentioned, this work focuses on the creation of a framework that allows an inspection using mostly images captured by the camera installed in the automaton.

The complexity of this work is in executing the inspection and controlling the robot only from images. For this to be possible, three different models will be trained: an autoencoder network to find anomalies, and a classification model and deep recurrent Q network to perform viewpoint planning in order to go to the found anomaly and inspect it, capturing the best images.

Details of the developed framework can be found in the next sections.

## 3.1
## Proposed framework

In order to find objects other than the pipeline, rocks or soil on the seabed, the following Visual Inspection of Pipelines (VIP) framework was created to automate the inspection process by an AUV.

While the AUV follows the pipeline using images and proportional control (3.1.2), the system checks for any anomalies through an autoencoder network (Figure 3.3 - 1). If an anomaly is detected, the system needs to save the current position (Figure 3.3 - 2) and navigate towards the object for a more detailed observation, utilizing viewpoint planning (Figure 3.3 - 3).

However, the process of selecting viewpoints can be slow and consume many computational resources, limiting the practical applicability of this technique. A solution to speed up the viewpoint planning process is the use of Deep Recurrent Q-Networks, an extension of conventional neural networks with an architecture that allows them to process sequences of inputs, such

as sequences of images. This feature makes DRQNs suitable for sequence processing tasks like viewpoint planning.

The main focus of viewpoint planning in this work is to determine whether the object is known or unknown to the model, that is, whether the agent should continue searching for better viewpoints or if it has already reached a conclusion: known or unknown object, and can stop moving. By default, the agent should move towards the object of interest and circle it using a tracking algorithm and a proportional control, capturing images at each step (Figure 3.1) and ultimately classifying the object. However, this path can be shortened through the use of DRQN.

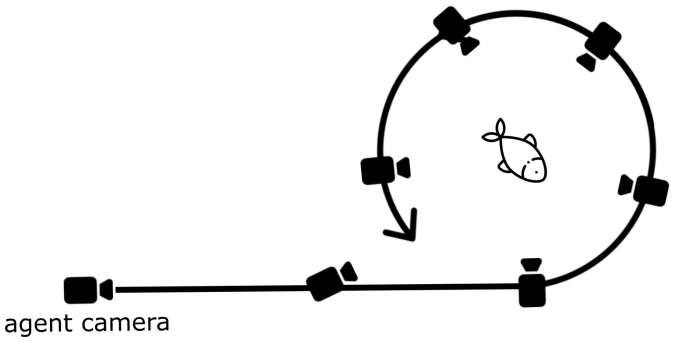


Figure 3.1: Proposed path to analyze an object

In order to achieve this, an SVM model (One-vs-Rest) is trained (Figure 3.3 - 4) using the images captured by the agent during the inspection process. This enables the SVM to learn information about the objects. This information is then passed from the SVM to the DRQN model through entropy and confidence metrics (Figure 3.2). In other words, the DRQN leverages information from the SVM to classify whether the agent should take another step (if the action is "continue") or stop searching for better images (use known and unknown consistently). The difference between entropy at $t-1$ and $t$ and also between confidence at $t-1$ and $t$ is calculated for a better data analysis over time, and also used as input to the DRQN. That is, the DRQN entries from the SVM are: confidence, confidence difference, entropy, entropy difference and the image number. The SVM output was chosen to be used in the DRQN (instead of using images directly) due to the fact that the SVM simplifies the input of the DRQN. This results in faster training of the DRQN, as well as allows for more efficient retraining since only the SVM needs to be readjusted.

This retraining of the SVM is necessary since the agent may initiate the inspection without prior knowledge of the anomalous objects it will encounter along the way. Thus, the system needs to acquire knowledge about these

objects, and this learning is accomplished through the SVM, which is retrained whenever a new object is detected to better understand it.

At the end of the inspection, a report is generated, containing information about all objects, including their coordinates, the path to the captured images, and an indication of whether the object is known to the system or not. This allows experts to analyze the results effectively and easily access inspection-related information.

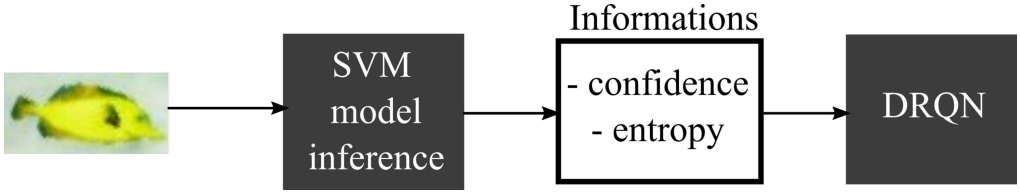


Figure 3.2: SVM and DRQN inputs pipeline

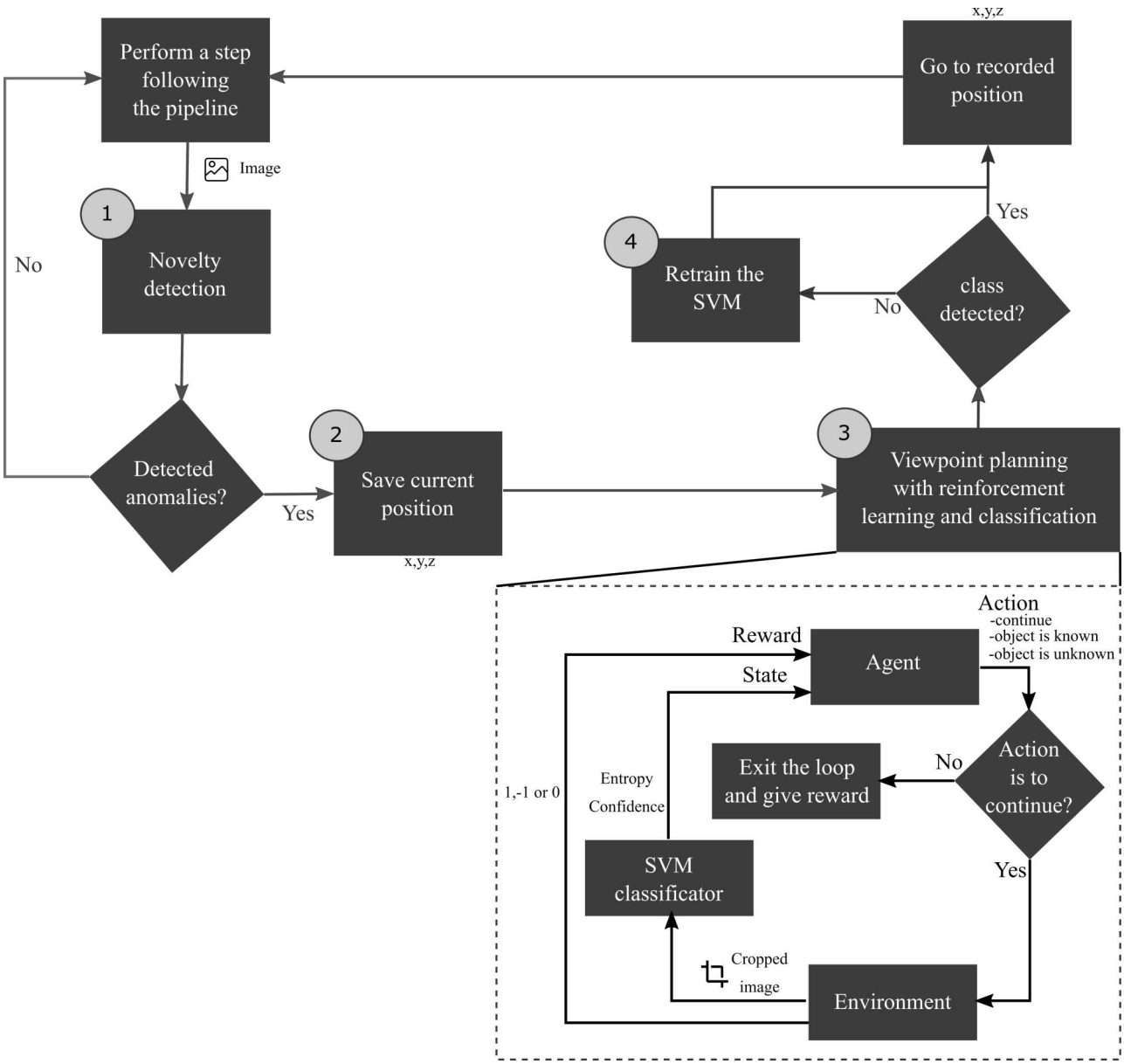


Figure 3.3: Proposed Visual Inspection of Pipelines (VIP) framework diagram

In the next sections, the following items will be explained in detail:

- Simulated environments
- Following the pipeline
- Novelty detection (Figure 3.3 - 1);
- Viewpoint planning (Figure 3.3 - 3).
- Microservices architecture

### 3.1.1 Simulated environments

The simulator chosen was AirSim (97), a simulator for drones, cars and more, built on Unreal Engine (98), a well-known game creation platform. AirSim has become the best choice compared to using Gazebo, as the images are more realistic and as its simulation speed is higher, as it does not incorporate simulation of elements such as gravity and other physical forces.

As said, the simulation scenarios were built using Unreal Engine. On this platform initially one environment was created, (Figure 3.4), with a notion of depth, aquatic background, animals, objects and ducts. The objective is: follow the pipeline, find objects/animals, inspect and ascertain whether the object is known or unknown.

It is notable that in the simulations performed, the agent is a drone. However, as the main objective of this work is the analysis of images combined with the control of the agent, the type of agent is not important, but the environment. Therefore, in this work, the drone will be called AUV to contextualize the study.

Using this simulator, we have the capability to acquire images from the agent's cameras for further processing and navigation guidance.

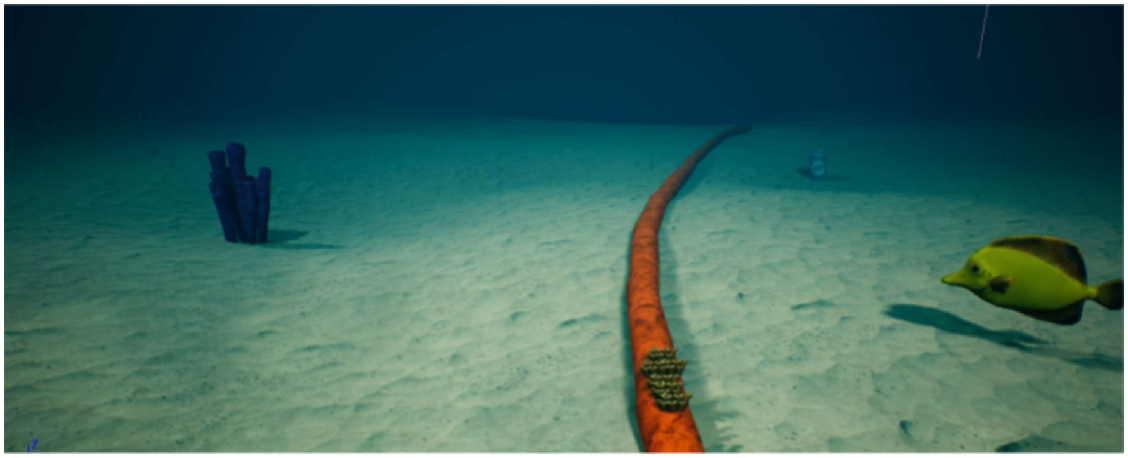

Figure 3.4: Simulated environment

### 3.1.2 Following the pipeline

To ensure the agent follows the pipeline correctly, it is essential that the agent's camera is positioned preferably above the duct. However, in certain sections, the duct may have curves, requiring the agent to be prepared to adjust its route. Therefore, a proportional control based on the angle formed between the duct and the x-axis has been implemented (as illustrated in Figure

3.5). The ideal value for this angle is 90 degrees. By calculating the difference between the current angle and 90 degrees, it is possible to perform proportional control appropriately.

However, since obtaining this information is solely based on images, certain processing steps are necessary to extract the relevant data. The first step involves binarizing the image, which entails selecting an appropriate threshold depending on the specific characteristics of the image at hand. This process separates the pixels of the pipeline from the rest of the image's pixels.

Once we have the binarized image, we can identify four points: two upper extremes and two lower extremes (Figure 3.5, points A, B, C, and D). By connecting these points, we obtain a polygon of similar size to the pipeline. By drawing a line through the center of this polygon, we can calculate the angle ($\alpha$) formed between this line and the x-axis. This angle will be used as a parameter for proportional control, as the ideal value is 90 degrees, as mentioned earlier.

For agent control, the previously calculated angle is used, as well as the proportional gain of the x-axis, which is calculated in equation 3-1, to perform lateral displacement. This way, the agent can move at a constant speed by rotating around its own axis, as well as moving left and right, always trying to keep its position on top of the pipeline.

$$px = M_x - width/2 \tag{3-1}$$

Where $M_x$ is the Centroid of X.

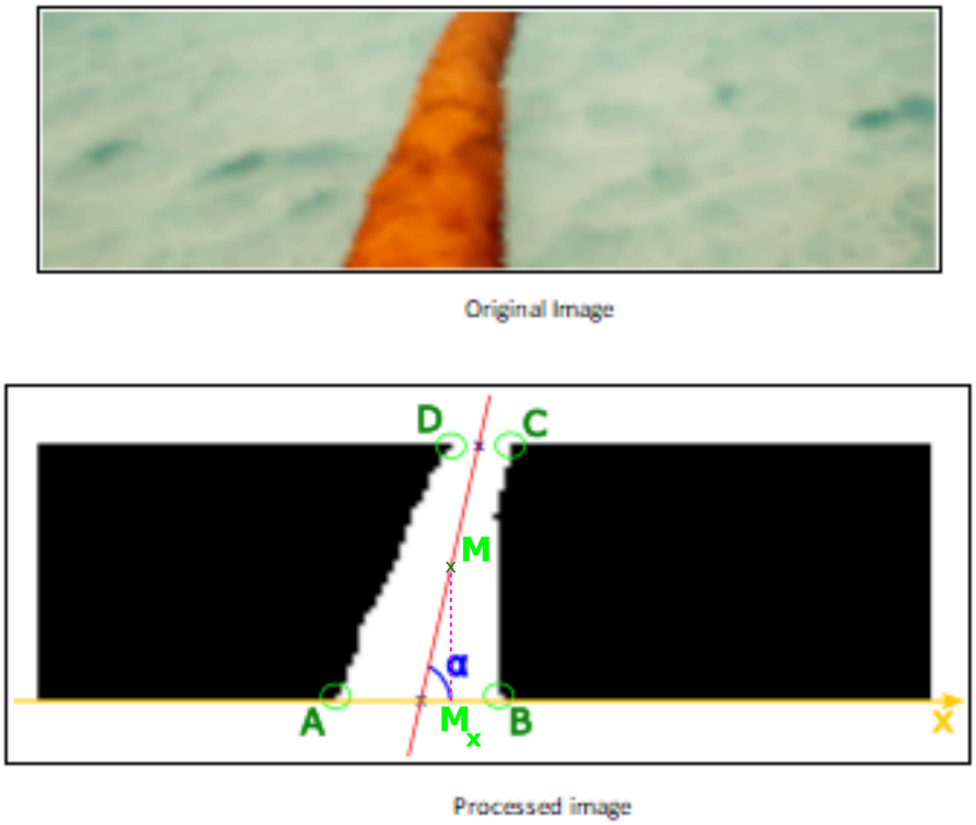


Figure 3.5: Original and processed image to enable the calculation of the agent's angle adjustment for proportional control.

### 3.1.3 Novelty detection

The choice of using autoencoders (99) was motivated by the fact that most works use these networks to detect novelties. This study tested Dense Autoencoder and Convolutional Autoencoder to recreate the input image and then subtract this result from the original image, obtaining the anomaly. These methods will be presented below.

An autoencoder network is trained in order to detect novelties/anomalies. The proposal is to train several networks with different depths and latent space sizes with a dataset that contains only soil, rocks and pipeline images. After training, the network can be used for inference. As the autoencoder only learned characteristics from the training dataset images, if there is something different in the input image, the network decoder will not be able to recreate the different object (anomaly) in the image. Therefore, given that the output image has the same dimensions as the input, it is possible to subtract these images and generate a mask (Figure 3.6), which represents the pixels of the image anomaly.

Additionally, we have improved the quality of the generated mask by eliminating any potential noise that may arise during this process. This is achieved through tests aimed at selecting the most appropriate threshold for mask creation.

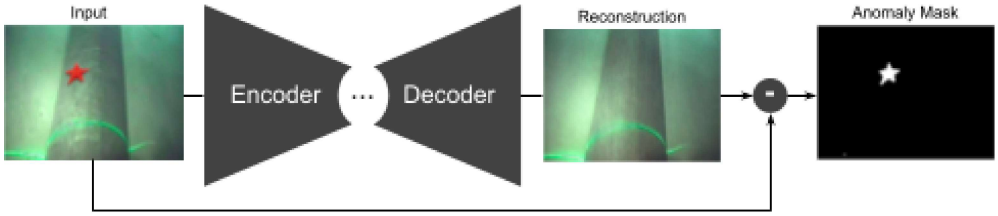


Figure 3.6: By subtracting the reconstructed image from the input image, it's possible to effectively reveal the anomalous regions present in the image.

### 3.1.4 Viewpoint planning

This section covers the presentation of the viewpoint planning system's development, addressing everything from the essential elements used in training to details about the system's functioning in a production environment.

The purpose of this system is the selection of optimal perspectives for obtaining high-quality images, as well as determining whether the object is known or not by the DRQN model. As mentioned earlier, the use of DRQN 2.3.5 has the potential to significantly reduce the number of captured images needed to support a decision, since it has memory.

The DRQN algorithm is composed of the following elements:

**Actions:** 0 (keep looking for new images), 1 (defines that it is a new object) and 2 (defines that it is not a new object);

**States:** Set of 4 consecutive vectors of [confidence, confidence difference, entropy, entropy difference, image number] from the chosen sequence, determined by the environment explained in section 3.1.4.1;

**Rewards:** 1 in case of success, if the object is known or not, and -1 in case of error. And the reward for each step the agent will take, will be defined later, which is a fundamental step for the success of the learning process. The proper selection of each step will allow the agent to explore the environment more efficiently and maximize the reward obtained, contributing to a better overall performance of the chosen model.

**Output:** the approximate $Q$ table for the given state.

Afterwards, it will be explained how the DRQN training will be implemented and how it will work in production.

#### 3.1.4.1 Python library to run the DRQN environment

Commonly, in algorithms like DQN or DRQN in Python, it is usual to use libraries such as Gym (100) or PyGame (101) to perform the calculation of rewards, next states, and final states. However, for this specific study, it became essential to create a custom library focused on calculating these elements. The need arose from the lack of a specialized library that employed SVM and sequences of images to calculate the next state and associated rewards.

This library is responsible for calculating the reward, changing the agent's state, and determining the next actions based on the model. Additionally, since the input to the DRQN consists of information obtained from the SVM, this library is also responsible for training an SVM with random data sequences from the dataset for each epoch and providing the trained SVM model to the DRQN. Furthermore, it randomly selects and provides the image sequence that will be analyzed by the model.

This library is available in <https://github.com/evysb/VIP-framework>.

#### 3.1.4.2 Following and analyzing the object

In addition to following the pipeline route, the Autonomous Underwater Vehicle (AUV) needs to have the capability to approach any detected anomalies. This will enable the AUV to circumnavigate the object of interest in order to conduct a detailed analysis. To perform this operation efficiently, a proportional control and tracking algorithm is employed.

When the system detects an anomaly using the Novelty Detection model, a mask is generated, as described in 3.1.3. This mask is used to create a bounding box (bbox) that will be the input for the tracking algorithm known as Median Flow (92), which provides the agent with a reference base of the object, producing as output the coordinates of the bounding box that encloses the object.

To control the agent, proportional control is used, similar to the one presented earlier in 3.1.2. This method guides the agent along a predefined path, encompassing the approach phase to the object and circumnavigation around it, enabling the acquisition of images from multiple angles.

The agent moves towards the object until it is in the center of the image. This movement is performed by attempting to balance the distances between the edges of the bbox and the boundaries of the image, as illustrated in Figure 3.7, where $d_{w1}$ and $d_{h1}$ must be equal to $d_{w2}$ and $d_{h2}$ respectively, or have

a difference of less than 1.2. Then it approaches the object until the area delimited by the bbox corresponds to 70% of the size of the original image, or as far as the simulator allows.

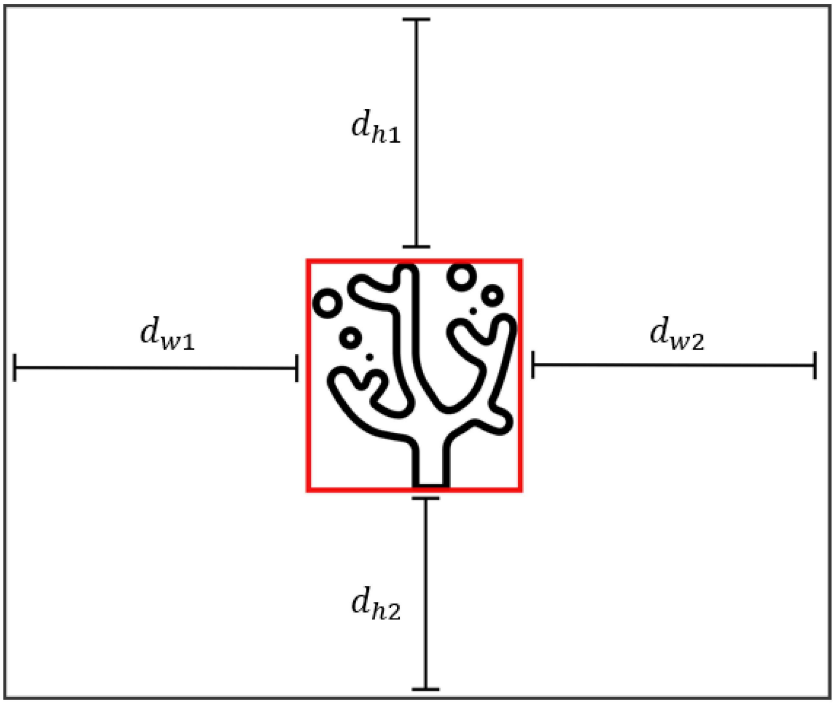


Figure 3.7: Object centering control in the image

As for circumnavigating the object, the agent maintains a constant speed, continuing its movement until the current distance is less than 0.7 the distance from the point where it started the circumnavigation, and it has already taken at least 5 steps.

#### 3.1.4.3 Training viewpoint planning

DRQN training will be performed as follows: the definitions of state, action, and reward will remain consistent with the DQN model. The main change now is the need to incorporate a sequence of information as input state. This sequence is obtained through the environment library (3.1.4.1), which randomly selects a series of images captured sequentially from a randomly chosen object.

Furthermore, the library also performs the random selection of an SVM model. This occurs because, during operation in a production environment, it is not possible to anticipate whether the examined object has been previously trained by the SVM. To address this uncertainty and strive for the generalization of DRQN training, the model was trained using a dataset that contains

both the presence and absence of the represented object in the sequence (as illustrated in Figure 3.8). This approach allows for simulating the system's behavior concerning known objects (when the SVM has been trained with the object image) and unknown objects (when the SVM has not been trained with the object image).

It is essential to emphasize the importance of these random choices, as they aim to obtain a generalized model capable of operating efficiently in various environments and with different classifiers, without the need to waste time on simulations that are not part of the viewpoint planning strategy in question.

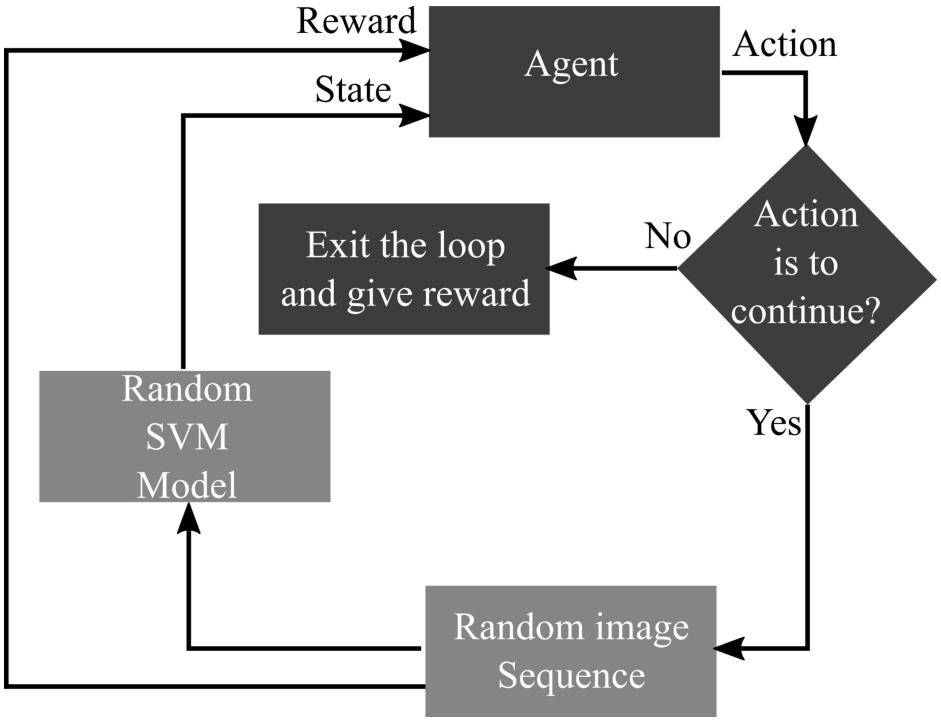


Figure 3.8: Training diagram of DRQN

#### 3.1.4.4 Viewpoint planning in production

With the model already trained, it will be possible to perform only the inference stage of the network (Figure 3.9). The difference in this stage is that the DRQN will use only the simulation environment (no longer preconceived image sequences) and an SVM model, which should be retrained (using the latest images seen by the agent for that object) if the last action taken by the DRQN results in: "1 - a new object".

Conversely, when the system receives an image similar to a previously seen object, it will be possible to classify it as "2 - not a new object".

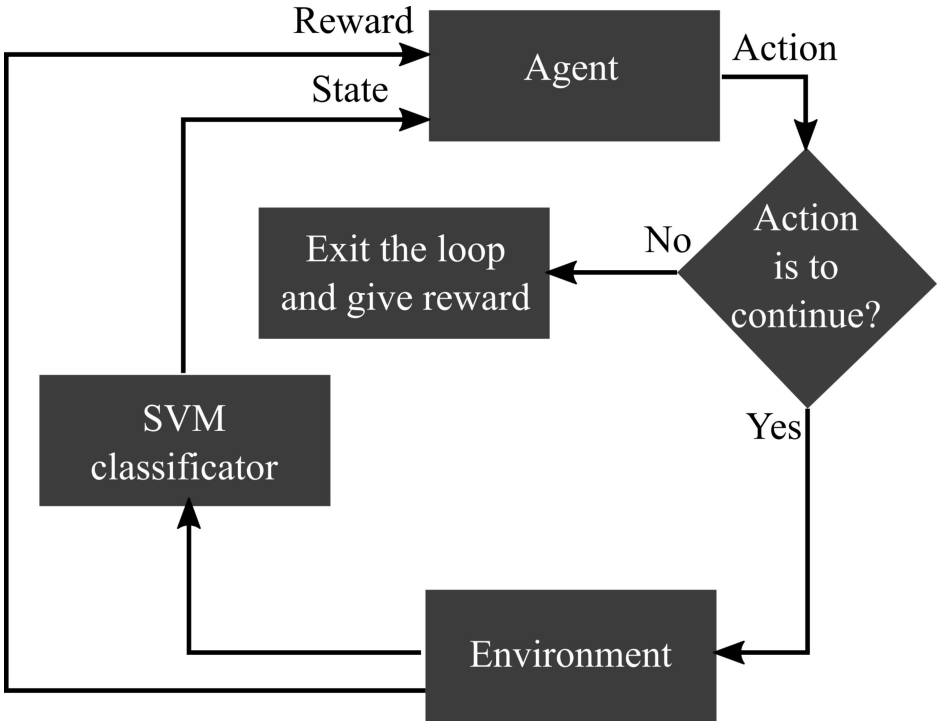


Figure 3.9: Inference diagram of DRQN

In Figure 3.9, we observe the process wherein the agent selects and executes an action. Based on the chosen action, the agent progresses by moving towards the target object (assisted by the mask generated by the anomaly detector, a tracking algorithm, and proportional control). At this juncture, the agent captures an image of the object. This image is then subjected to SVM processing. The resulting information, including entropy and confidence scores, is employed to feed the DRQN agent. This plays a pivotal role in computing the subsequent state and determining the associated reward. Thus, at time $t + 1$, the agent is poised to calculate and take the next action.

In Figure 3.10, it is possible to observe the input images of the SVM, the information provided by it to the DRQN (entropy and confidence), and the output of the DRQN agent (actions). Furthermore, it is an example of the optimization that the DRQN offers, in this case, with only 3 out of 12 images, which make it possible to conclude that the object is new.

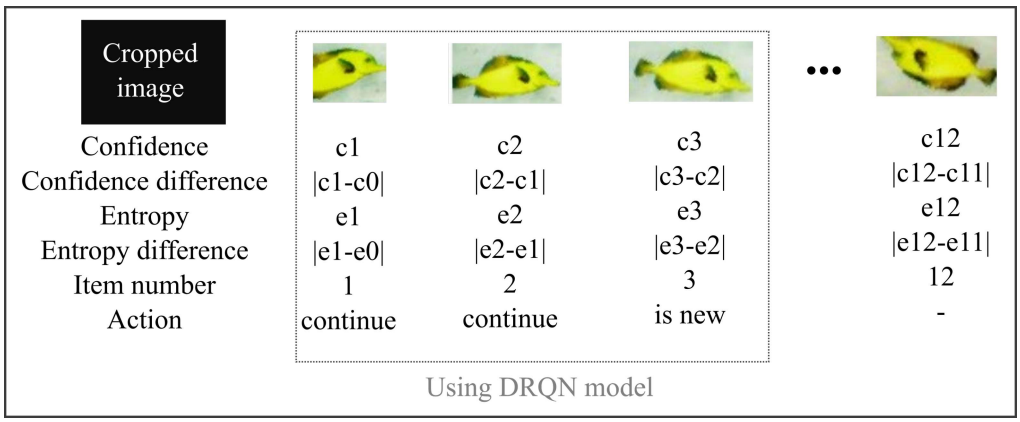


Figure 3.10: DRQN model input schema

### 3.1.5 Microservices architecture

To enable the utilization of multiple GPUs, given the need to use a simulator, load 2 models, and (re)train the SVM simultaneously, individual microservices were created, each utilizing one GPU, which respond to a main microservice.

The adoption of microservices allows the development of services using technologies that are more suitable for their specific functionalities. Each service is responsible for a specific functionality. This promotes decoupling between services, making it easier to evolve and maintain the system as a whole. Additionally, the individual scalability of each service allows for better utilization of available resources, avoiding waste and maximizing efficiency.

An example of this is the microservice responsible for the environment simulator (Figure 3.11 - 4), which focuses exclusively on executing the Unreal Engine. It awaits requests containing actions to be performed and, in turn, returns the images and coordinates of the current state.

The novelty detection microservice (Figure 3.11 - 2) is responsible for identifying unexpected events or patterns, while the viewpoint planning microservice (Figure 3.11 - 3) handles the inference of the DRQN network and, when necessary, performs the (re)training of the SVM. One interesting aspect is that, thanks to the utilization of microservices, it is possible to perform SVM training simultaneously while the agent follows the normal inspection flow. In case the agent encounters a new object to be inspected during training, there will only be a pause to await the completion of the training.

In the main microservice (Figure 3.11 - 1), on the other hand, the agent's control codes are centralized, being responsible for sending requests to the other microservices.

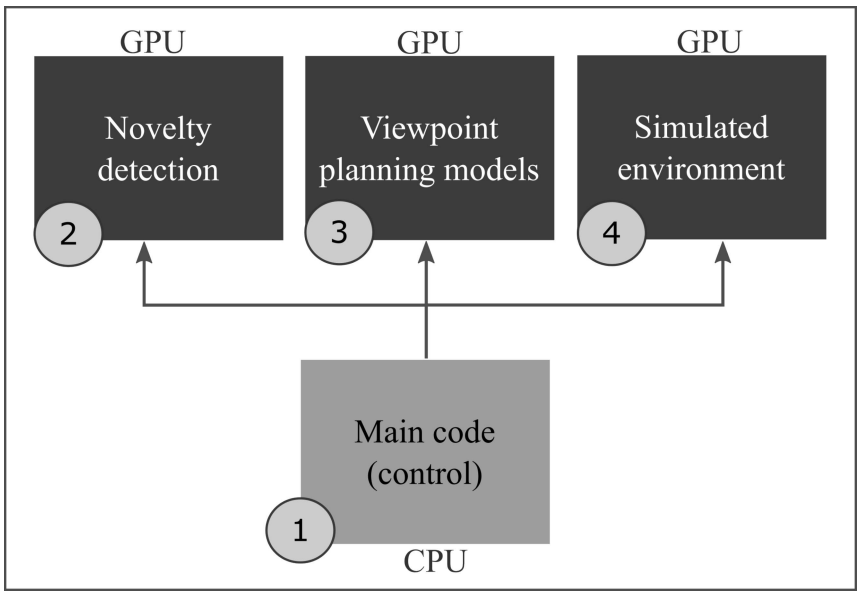


Figure 3.11: Microservices framework diagram

Furthermore, the Main code is responsible for saving all captured images for the analysis of each object into separate folders (corresponding to each object), and generating a report (Figure 3.12) that includes the coordinates where the object was found, as well as the path of the folder where the images are stored, and whether the object was considered known or not. This enables the specialist to read and analyze the results in a more efficient and accurate manner after the inspection.

| Objects | Coordinates | Folder Path | Is new |
|---|---|---|---|
| obj 1 | coord x1, y1, z1 | /path/to/folder/1 | TRUE |
| obj 2 | coord x2, y2, z2 | /path/to/folder/2 | TRUE |
| obj 3 | coord x3, y3, z3 | /path/to/folder/3 | FALSE |
| obj n | coord x4, y4, z4 | /path/to/folder/n | TRUE |

Figure 3.12: Report example

Based on the data analyzed by an expert, it is possible to train a new SVM model in order to enable the autonomous system to initiate the next inspection with prior knowledge, using the best available model.

# 4
# Experiments

## 4.1
## Novelty detection

Following this study's objective, many images were created to test and train the autoencoders, in order to test the robustness of them. For this, two datasets were created, one generated from images captured from the AirSim (102) simulator, which we call Synthetic dataset, and another dataset created from videos of ROV inspections, which we call Real dataset (Figure 4.1).

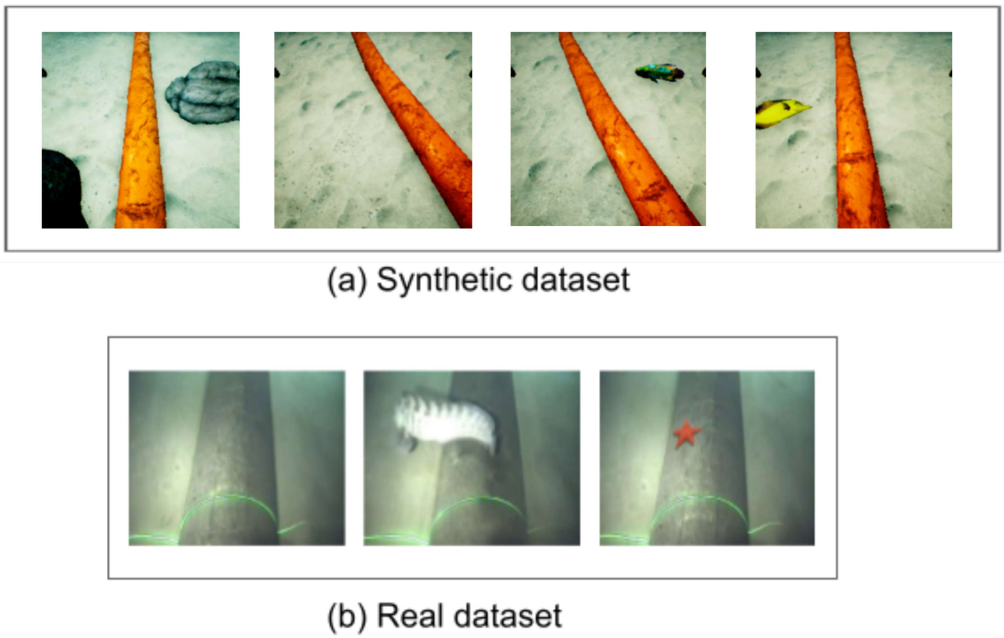


Figure 4.1: Dataset images with synthetic (a) and real (b) image.

- Synthetic dataset: composed of 933 color images of 224x224 (50% anomaly, 50% normal);
- Real dataset: composed of 1475 color images of 640x480 (23% anomaly, 77% normal).

These two datasets were created to test the robustness of the model, both in simulated environments and in real life. For training, they were divided into training (only images without anomalies) validation and testing, 70%, 20% and 10%, respectively.

### 4.1.1 The architecture of the network chosen

To choose the best architecture for the model, several autoencoders with different depths were trained for the two datasets. Furthermore, the convolutional autoencoder (cAE) networks were based on (103) and the latent layer size of dense autoencoder (dAE) network was modified from 4 to 1064, in other words, for each proposed dAE model, there are 9 different architectures changing only the latent layer. To evaluate the inference of each model, the Dice similarity coefficient (104) (Equation 4-1) was used.

$$\text{Dice Score} = \frac{2|X \cap Y|}{|X| + |Y|} \tag{4-1}$$

Where $X$ and $Y$ represent two sets for which we want to measure the overlap. $|X|$ and $|Y|$ denote the cardinalities (i.e., the number of elements) of the sets X and Y, respectively. The intersection of the two sets, denoted as $X \cap Y$, represents the elements that are common to both sets, such as non-anomalies.

For the decision of the best architecture to be used in this work, the results in the synthetic dataset will be decisive, because the images in it are closer to what the automaton will see in the simulations. From it, the threshold of the anomaly masks will also be chosen (Section 4.1.3).

The following encoder architectures (Table 4.1 and Table 4.2) were used (the decoder is the opposite of the encoder, using *Conv2DTranspose*). All of them used *Conv2D* and *Conv2DTranspose* (functions from the Tensorflow library (105)) with stride 2 and $3 \times 3$ filters.

Table 4.1: Architecture table of trained models with Synthetic dataset. Where the latent layer size of dense autoencoder (dAE) network was modified from 4 to 1064.

| | dAE Model 1 Synthetic dataset | dAE Model 2 Synthetic dataset | cAE Model 1 Synthetic dataset | cAE Model 2 Synthetic dataset | cAE Model 3 Synthetic dataset |
|---|---|---|---|---|---|
| Encoder filters | 32, 64 | 16, 32, 32, 64, 64 | 16, 8, 8, 8 | 16, 8, 8, 8, 8 | 16, 8, 8, 8, 4 |

Table 4.2: Architecture table of trained models with Real dataset. Where the latent layer size of dense autoencoder (dAE) network was modified from 4 to 1064.

| | dAE Model 1 Real dataset | dAE Model 2 Real dataset | cAE Model 1 Real dataset | cAE Model 2 Real dataset | cAE Model 3 Real dataset |
|---|---|---|---|---|---|
| Encoder filters | 32, 64 | 16, 16, 32, 32, 64, 64, 64 | 16, 8, 8, 8 | 16, 8, 8, 8, 8, 8, 8 | 16, 8, 8, 8, 8, 4, 4 |

The hyperparameters used in the training of the both datasets are shown in Table 4.3.

Table 4.3: Hyperparameters of autoencoder training

| Hyperparameters | Values |
|---|---|
| Max epochs | 200 |
| Batch size | 4 |
| Learning rate | 0.0001 |
| Optimizer | Adam |
| Early stopping | min_delta=0.0001, patience=20 |

### 4.1.2 Training summary

This section aimed to make a neural network recognize an anomaly in a series of images. For this purpose, a neural network was trained. The network input, described above, requires at least one image per batch. The pre-processing only requires resizing each image to $224 \times 224$ (if Synthetic dataset is used) or to $640 \times 480$ (in case of Real dataset) and a normalization. In certain cases, due to the limitations of the neural network's training format, it becomes unfeasible to use images with dimensions of $640 \times 480$. Consequently, it becomes necessary to resize the images to $640 \times 384$.

All models were trained just using images without anomalies. Moreover, the images (with and without anomalies) were semantically segmented by me, so that it was possible to have masks (ground truth) to calculate the Dice, in order to choose the best trained model.

### 4.1.3 Choosing the image mask threshold

After training the model, it is tested. The test consists of using an image (with or without anomaly) as input to the model. As said in 3.1.3, the model will output an image of the same dimensions as the input, this output will be subtracted from the input image generating a mask (Figure 3.6). To better tune this mask, a threshold is chosen. So that the choice of this threshold is not random, tests were performed with thresholds 0, 5, 10, 20, 30, 40, 50, 60, 70, 80, and 90 using images not used in training. The threshold that maximizes the Dice score on the validation set was chosen.

### 4.1.4 Choosing the loss function

The loss function, also known as the error function or cost function, is a fundamental part of deep learning models. It is responsible for quantifying the difference between the model's predictions and the actual values of the data set. The objective of the model is to minimize this error function by adjusting its parameters during the training process. There are several types of error functions, each suitable for different types of problems, such as classification or regression. Choosing the right error function can have a significant impact on the effectiveness of the model and how quickly it converges to an optimal solution.

In this study, MSE 2.1.3.1, SSIM 2.1.3.2 and SSIM+MSE (using normalization to prevent metric overlap) were tested as loss function, in order to obtain the best possible models.

### 4.1.5 Evaluation of results using Dice similarity coefficient

With the threshold chosen appropriately for each model, images (with and without anomalies in the same proportion) not used in training and in choosing the threshold are used for the test. A similar process to the one explained above is carried out. Images are the inputs of the trained model and its output is subtracted from the input image, generating a mask. Then the Dice score is calculated. The model with the highest Dice score is considered the best.

### 4.1.6 t-SNE graphics creation

In order to know how much the representations created by the autoencoder networks can distinguish the images with anomalies from the images without, the t-SNE algorithm (106) (t-Distributed Stochastic Neighbor Embedding, a high dimension data visualization algorithm) is used. For this, the same data from the test stage are used, they are input into the model and the encoder output is graphed using the t-SNE algorithm.

## 4.2 Viewpoint planning

The inputs of the DRQN model are information such as entropy and confidence originated from the SVM. Thus, we can divide the processes as in the Figure 3.2.

For the proper training of the DRQN, pre-trained SVM models need to exist, as said before, so we will divide the sections into: SVM training, DRQN training, and finally the baseline to compare with the results of the DRQN models.

### 4.2.1 Datasets to training models

In this section, we introduce the datasets used to train the fundamental models for Viewpoint planning, which are:

#### 4.2.1.1 SVM Dataset

A dataset consisting of multiple image sequences was created for training the model. Since the agent's aim is to reach the object of interest and perform viewpoint planning to find a better image of it, multiple image sequences can be saved for model training, as each step corresponds to obtaining an image.

To create this dataset, the agent needed to approach various objects multiple times, ensuring that the dataset is generalizable enough. After creating the dataset, the use of the simulator is not necessary, which reduces the consumption of computational resources.

As the aim of this study is solely focused on the objects appearing in the image, the images were inferred by an object detector, and subsequently, the objects of interest were cropped (Figure 4.2). These cropped images are used for SVM model training.

The image sequences were taken from an underwater environment created in Unreal (98), they are RGB images of size 64 x 64. These sequences contain 1 to 20 images per object and 25 objects (e.g. fish, can, jar ...), and each object has at least 20 sequences of images, which gives us a dataset containing a minimum of 500 sequences.

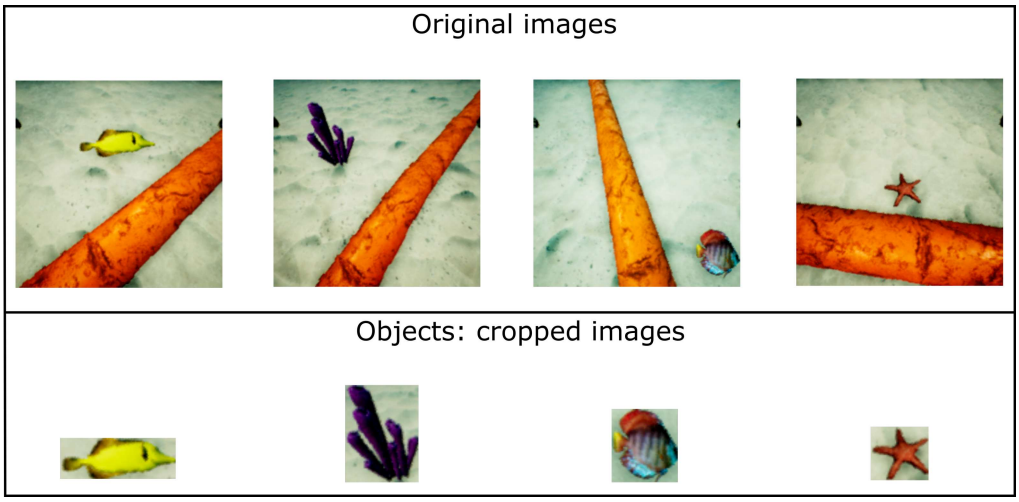


Figure 4.2: Object image dataset for SVM training

#### 4.2.1.2 Baseline and DRQN dataset

The Baseline and DRQN share the same dataset. This dataset consists of metrics generated from the output of the SVM model, with the help of the library created in 3.1.4.1. It is relevant to highlight that this library has the capability to randomly employ various trained SVM models, along with sequences of random images, which contributes to an increased capacity for generalization in model training.

Each entry in the dataset is composed of:

- Entropy;
- Difference between entropy $t-1$ and $t$;
- Confidence;
- Difference between confidence $t-1$ and $t$;
- Image number.

And entropy and confidence are calculated in the following way:

- Entropy: Since the output of the SVM consists of a decision score that evaluates the distance of a data point from the decision hyperplane, the

Platt calibration technique (107) is employed to transform these decision scores into probability estimates. With these probabilities in hand, the entropy (Equation 4-2) is calculated.

$$H(X) = -\sum_{x} p(x) log p(x) \tag{4-2}$$

– Confidence: The confidence of an SVM model is the measurement of the distance between the data points and the separation hyperplane identified by the SVM. The greater the distance between a data point and the hyperplane, the higher the confidence associated with that point, making its classification more secure.

### 4.2.2 SVM training

To use the outputs of the models by DRQN, it is necessary to train several SVM models with images of random classes. This training occurs on demand, being performed in real-time by the code of the environment (3.1.4.1).

The SVM was chosen to be used in this work due to its relatively fast training time and efficiency in processing large amounts of data.

Testing different configurations to determine which model will produce the best results is always important. Several settings can be adjusted in SVM, such as the kernel used, the regularization parameter, and the kernel coefficients. Adjusting settings is crucial to obtain the best possible results from the SVM model and ensure that it is properly trained for the specific task being performed.

To do this, the GridSearchCV function from the scikit-learn library (108) was employed to identify the optimal configuration of the SVM when using the SVM dataset 4.2.1.1.

In order to assess whether the accuracies of SVM models decrease significantly as the number of classes increases, we incorporated 25 new objects into the existing dataset. These sequences of images are shorter than the existing ones, containing approximately 6 images per object. Conducting this type of test is of utmost importance because in a practical inspection scenario, it is not possible to accurately determine how many objects may be encountered along the path, which would require retraining the SVM.

Using the dataset of 50 classes (objects) and the obtained configuration, several SVM models were trained. This process was repeated 10 times, with the classes added in random order in each iteration, in order to calculate the average accuracy of the models.

### 4.2.3 Baseline

In order to verify the efficiency of the DRQN, a simple network without LSTM (Table 4.4) was trained with identical inputs to the DRQN and with the output of "is new" or "is not new". Two training sessions were conducted: the first using only the initial images of the sequences, and the second using only the final images of the sequences. This allows us to measure how much better the use of DRQN can be.

Table 4.4: Neural network architecture used in baseline

| Layer | Output space dimensionality |
|---|---|
| Input | 5 |
| Dense | 32 |
| Dense | 16 |
| Dense | 2 |

### 4.2.4 DRQN training

In order to achieve the best possible DRQN model, multiple hyperparameter configurations were tested, including variations in the values of gamma and reward. They are important because they influence, for example, the number of steps the agent takes.

The gamma value can range from 0 to 1; the smaller the gamma, the more it relies on immediate rewards. Since the immediate reward for stopping (when the action is "known" or "unknown") is 1, and negative in case of an error or the action being "continue", the agent tends to take fewer steps because it follows a greedy policy, always seeking the highest reward. However, to have a high probability of correct classification, the agent must first take some steps that incur a negative reward in order to gather sufficient information. Therefore, finding an appropriate balance between reward values and gamma is of utmost importance. This will enable the agent to make precise decisions, taking fewer steps, and maximizing its effectiveness in the given task.

Training configurations that remained constant were:

**Episodes:** 1200 episodes - 200 of them are random, which means that the agent randomly chooses the actions to be carried out, to be able to do exploration and exploitation;

**Neural network used:** Described in the Table 4.5.

Table 4.5: Neural network architecture used in DRQN model

| Layer | Output space dimensionality |
|---|---|
| Input | 5 |
| LSTM | 32 |
| Dense | 16 |
| Dense | 3 |

## 4.3 Framework

After completing the training of all models and ensuring the proper functioning of the microservices, the integrated execution of the framework finally becomes viable. These series of experiments were conducted with the purpose of assessing the system's robustness, as well as identifying its advantages and disadvantages.

Tests were conducted in order to compare the advantage of using DRQN in viewpoint planning. The first experiment involved the agent traversing the entire pipeline and inspecting objects without utilizing DRQN, while the second experiment incorporated the application of DRQN. The inspection times for each object were recorded. To conduct these tests, five environments containing distinct objects were utilized. Additionally, several SVM models were applied to simulate situations in which the agent could be familiar or unfamiliar with the objects.

To test the entire VIP framework, three Nvidia V100 GPUs were used, with one allocated to each microservice: novelty detection, viewpoint planning, and environment simulator.

Furthermore, analyses were conducted to identify potential errors that may arise during the system's utilization, as well as its expected behavior in such situations.

# 5
# Results

## 5.1
## Novelty Detection

The results of the training are reported below. The experiments were separated as follows:

- Best image mask threshold for each network architecture;
- Best loss function;
- Best model for Synthetic dataset;
- Best model for Real dataset.

### 5.1.1
### Best image mask threshold for each network architecture

In this experiment, the objective is to choose a threshold that maximizes the Dice score, in order not to use a random threshold.

The threshold choice is highly important, as increasing the threshold could lead to the exclusion of crucial pixels for forming the mask that identifies the anomalous object. Unfortunately, this action would result in the loss of important information. To avoid this type of loss, we calculate the Dice score for each threshold in order to find the threshold that maximizes the Dice score.

The results of both the Synthetic (Table 5.1) and Real (Table 5.2) datasets are found in the following tables.

Table 5.1: Threshold for models trained with the Synthetic dataset. (*) Mode of thresholds that maximize the dice score.

| | dAE Model 1 Synthetic dataset | dAE Model 2 Synthetic dataset | cAE Model 1 Synthetic dataset | cAE Model 2 Synthetic dataset | cAE Model 3 Synthetic dataset |
|---|---|---|---|---|---|
| Threshold | 70(*) | 80(*) | 30 | 40 | 60 |

Table 5.2: Threshold for models trained with the Real dataset. (*) Mode of thresholds that maximize the dice score.

| | dAE Model 1 Real dataset | dAE Model 2 Real dataset | cAE Model 1 Real dataset | cAE Model 2 Real dataset | cAE Model 3 Real dataset |
|---|---|---|---|---|---|
| Threshold | 30(*) | 30(*) | 20 | 20 | 20 |

Analyzing the results, it is evident that the synthetic dataset showed a higher threshold compared to the real dataset. This can be attributed to the presence of less complex and more similar images in the synthetic set, resulting in masks with reduced levels of noise.

### 5.1.2 Best loss function

For all architectures created, the training was done using 3 variations of loss function: MSE, SSIM and SSIM+MSE, in order to discover the best one for the proposed task.

For both the dAE and cAE architectures, the loss SSIM+MSE presented the best results (Table 5.3), followed by the results of the loss being only SSIM and MSE.

Table 5.3: Comparison of the mean Dice score of models trained using MSE, SSIM and SSIM+MSE.

| | dEA mean dice score | cEA mean dice score |
|---|---|---|
| MSE | 0.4427 | 0.3712 |
| SSIM | 0.6314 | 0.5132 |
| SSIM+MSE | 0.8125 | 0.6510 |

Thus, we can conclude that when comparing the generated images with the original ones, it is more important to consider information about luminance, contrast and structure (calculated by SSIM) than just the difference in pixel values (calculated by MSE). However, the use of all information makes the result more accurate.

### 5.1.3 Best model for Synthetic dataset

After choosing the threshold for each model, they were tested. For the Synthetic dataset, the results on the test set, with hyperparameters chosen using the validation set were as follows (Table 5.4):

Table 5.4: Dice-Score (mean and std. deviation of images) for trained models with Synthetic dataset. Where LD is latent dimension.

| | dAE Model 1 (LD = 256) Threshold=70 | dAE Model 2 (LD = 512) Threshold=80 | cAE Model 1 Threshold=30 | cAE Model 2 Threshold=40 | cAE Model 3 Threshold=60 |
|---|---|---|---|---|---|
| Dice ($\mu \pm \sigma$) | 0,90612 ± 0,12102 | 0,89964 ± 0,10237 | 0,82241 ± 0,15553 | 0,66944 ± 0,23307 | 0,55494 ± 0,19888 |

Aiming to experiment with different networks of different depths, dAE and cAE networks were trained, and for dAE the latent dimensions were varied from 4 to 1024 resulting in 9 trained models for each dAE architecture variant, and from these variations the 3 loss functions were used. In other words, 27 models were trained for dAE models and 9 for cAE models.

For "dAE Model 1 Synthetic dataset" the highest score resulted from the use of the latent dimension of 256 and using the threshold selected in the previous step (Figure 5.1), while for "dAE Model 2 Synthetic dataset" the latent dimension that maximizes the score using the selected threshold is 512.

The best results came from the dAE models, using the SSIM+MSE loss. The best of these models, as seen in table 5, is the dAE Model 1.

It can be concluded that these dense autoencoders work well for the images in the dataset used, with few complexities. And the cAE models achieved moderate results due to the occurrence of noise in some cases, which generated inadequate masks.

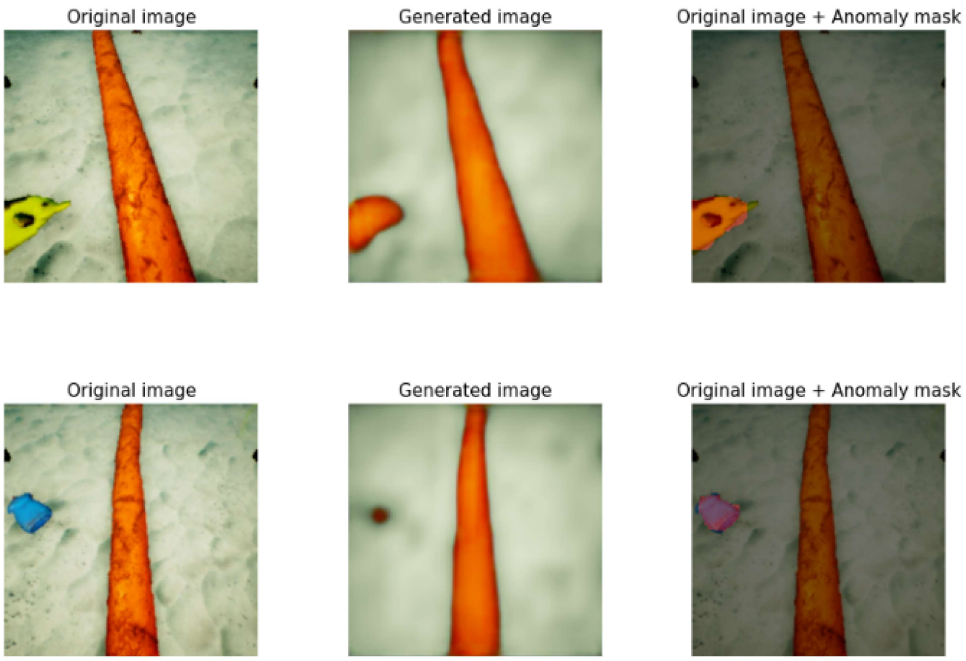


Figure 5.1: Images resulting from "dAE Model 1 Synthetic dataset" inference

The t-SNE was used in order to observe if the embeddings of data with anomalies can be separated from those without anomalies. The best possible result would be if the two classes were divided into well-defined clusters.

The t-SNE result, generated based on the best trained model and its embeddings (Figure 5.2), clearly illustrates the segregation between the representations of images with anomalies and those without. This highlights the effectiveness of the model in successfully performing its function of distinguishing between anomalous and non-anomalous images.

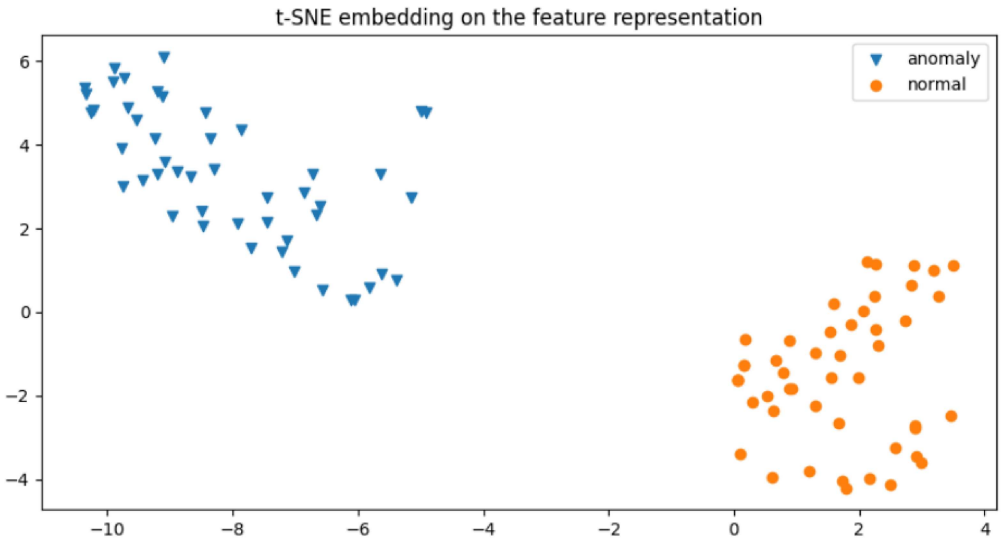


Figure 5.2: t-SNE result of model "dAE Model 1 Synthetic dataset" encoder representation

### 5.1.4 Best model for Real dataset

Using the real image dataset, experiments similar to the Synthetic dataset were performed, but some networks were deeper (Table 4.2), given the dimensions of the dataset images (640x480).

The model results for the Real dataset on the test set, with hyperparameters chosen using the validation set were as follows (Table 5.5):

Table 5.5: Dice-Score (mean and std. deviation of images) for trained models with Real dataset. Where LD is latent dimension.

| | dAE Model 1 (LD = 64) Threshold=30 | dAE Model 2 (LD = 256) Threshold=30 | cAE Model 1 Threshold=20 | cAE Model 2 Threshold=20 | cAE Model 3 Threshold=20 |
|---|---|---|---|---|---|
| Dice ($\mu \pm \sigma$) | 0,76289 ± 0,24197 | 0,79379 ± 0,27118 | 0,54545 ± 0,14271 | 0,56118 ± 0,15140 | 0,53711 ± 0,15029 |

For this dataset, the architecture using dAE with many convolutional layers (dAE Model 2) and with a latent dimension of 512 obtained the best result (Figure 5.3), despite the difference between dAE Model 1 and dAE Model 2 be statistically insignificant.

In this dataset, the worst results were obtained in the cAE models, the generated images were not very different from the original ones, however had a lot of noise, which did not occur in dAE models, which filtered out the existing noise due to the low latent space dimensionality.

In some of the results, some areas of the images were more segmented than necessary, but this can be advantageous, even if it may lead to an increase in the number of false positives - that is, regions that are mistakenly identified as belonging to the object of interest - it is preferable to have false positives above false negatives. This is because a false negative indicates that a region that should have been segmented as part of the object was mistakenly left out, causing the area not to be observed during the inspection. This can lead to inaccurate results and misinterpretations of the image. On the other hand, a false positive can be easily eliminated during the post-processing step.

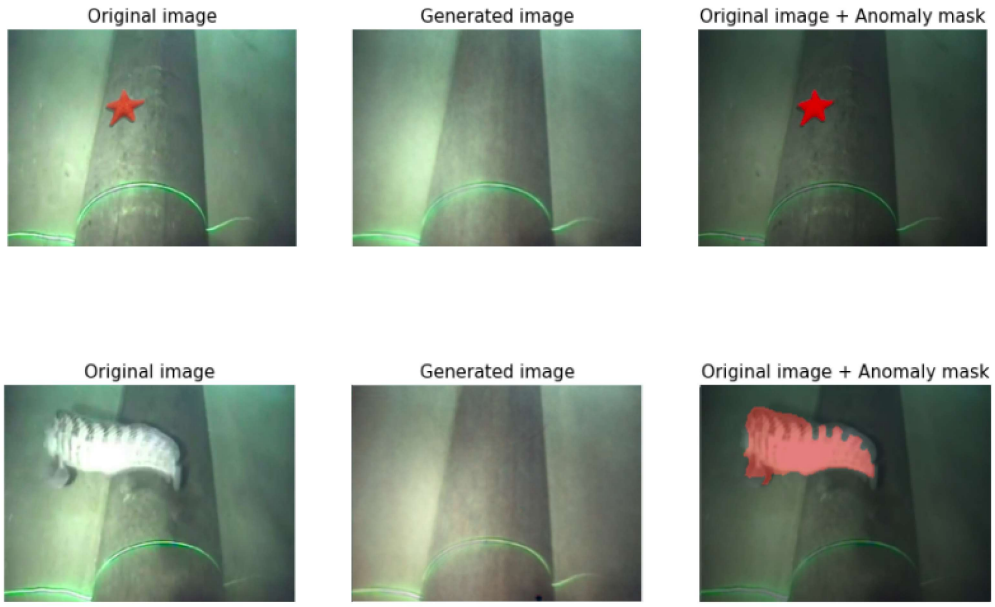


Figure 5.3: Images resulting from "dAE Model 1 Real dataset" inference

Once again, to analyze the results, a graph was created using the embeddings of the most effective model, through the t-SNE technique (as illustrated in Figure 5.4). It can be inferred that even facing a dataset containing larger and more complex images, the model once again demonstrates its remarkable ability in distinguishing between images with and without anomalies. This performance is clearly visible in the segregation of the two classes of embeddings, with only a few distinct representations in proximity.

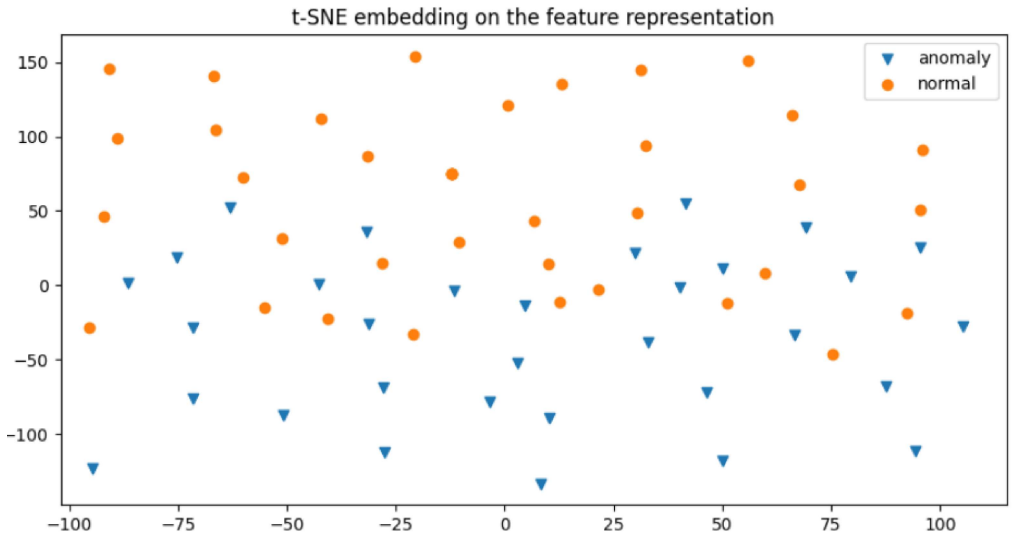


Figure 5.4: t-SNE result of model "dAE Model 1 Real dataset" encoder representation

## 5.2 Viewpoint planning

The results of the training are reported below. The experiments were separated as follows:

- SVM model;
- Baseline model;
- DRQN model.

### 5.2.1 SVM model

As previously mentioned, the best configuration for the SVM model was first found. Which obtained using the as the best model the configuration:

Kernel: RBF, C: 10 and Gamma: 0.001

Using this configuration, multiple SVM models were experimented with, having been trained on as many as 50 classes. This process was repeated 10 times, with the classes being added in random sequences each time. Subsequently, an average accuracy score was computed for each quantity of classes. (Figure 5.5). The results were promising, showing that even after increasing the classes, the accuracy remains high.

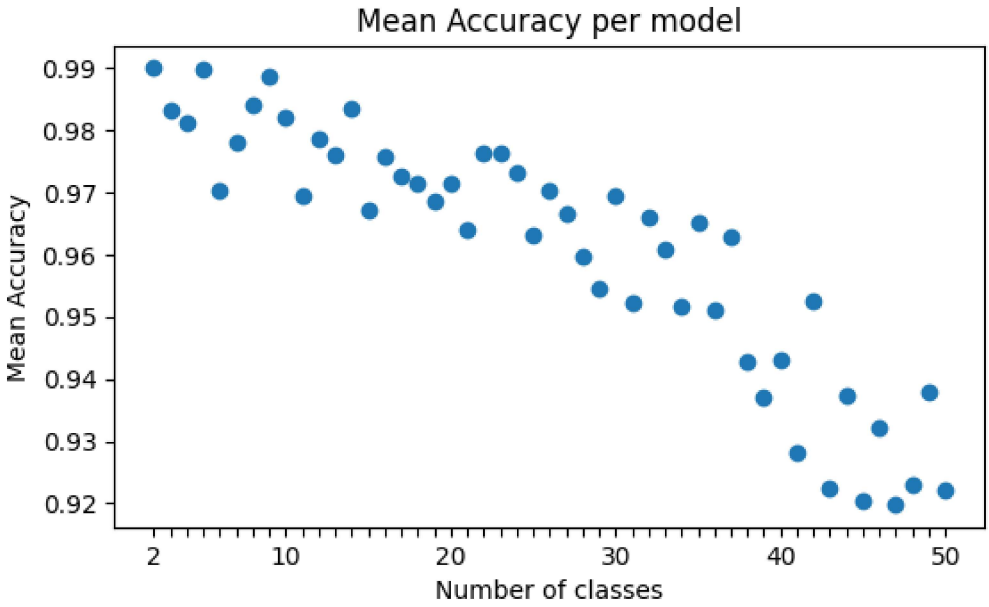


Figure 5.5: Accuracy per SVM model

Since the output of the SVM is used as input for the DRQN model, and considering that the object to be investigated is not always known by the

SVM, relying solely on accuracy/probability does not yield satisfactory results for determining whether the object is known or not. Therefore, we use entropy and confidence score, which in SVM is calculated from the signed distance of that sample to the hyperplane.

An example between confidence and probability estimate for an object that is known by the model and another that is not known:

**Known object:** Probability: 0.8160, Confidence: 16.2

**Unknown object:** Probability: 0.8982, Confidence: 1.7

These results show that we could erroneously say that the model classified the unknown object well if we did not take confidence into account. As detailed in section 4.2.1.2, when dealing with an unknown object for the SVM model, even if the probability is high, it is possible to observe that the distance between the sample and the hyperplane is small, which results in a confidence reduced in the result.

### 5.2.2 Baseline model

A simple neural network was trained, consisting of two fully connected layers (Table 4.4), having the classes "is new" (0) and "is not new" (1) as output and the same input of the DRQN model.

As previously stated, two training sessions were carried out (10 times each), the results were as follows:

- Mean test accuracy of the trained model using only the first images of the sequences: 76.7%
- Mean test accuracy of the trained model using only the last images of the sequences: 86.3%

These results are good; however, it is expected that with the use of DRQN, the presented accuracy will be higher, in addition to requiring fewer steps to make decisions on whether an object is new or not.

### 5.2.3 DRQN model

Several DRQN models were trained using the previously mentioned dataset to find the best result. In order to optimize the performance of the model, different configurations of received rewards and gamma values were tested (10 times each). The results of these experiments will be presented below to evaluate the effectiveness of the DRQN model against the baseline and provide insights into the impact of these settings on model performance.

As mentioned previously, the training process utilized 1200 episodes per training, requiring approximately 3 hours when executed on an Nvidia V100 GPU.

– **Choice of gamma**

The proper choice of the gamma value is a crucial aspect of agent training (as explained in 4.2.4) since this parameter directly impacts the model's performance. Several values were used to evaluate the effects of different gamma values, including 0.99, 0.80, 0.70, 0.60, 0.50, 0.40, 0.30, 0, 20, 0.10, and 0.0. The results obtained with these values are represented in Figure 5.6, allowing the evaluation of the relationship between the gamma value and the agent's error rate.

It was observed that the lower the gamma value, the lower the number of steps performed by the agent. This is because the immediate reward is prioritized over the future reward, which can reduce the overall performance of the model. For example, when gamma is 0.0, the agent's average steps is 1, and when it is 1.0, the agent takes as many steps as possible. In other words, a wrong choice of gamma can result in a higher error rate and a longer time for model convergence.

The result that showed the most favorable trade-off between accuracy and the number of steps was obtained with a gamma value of 0.99. This value will be used in the following tests to evaluate the reward for each step.

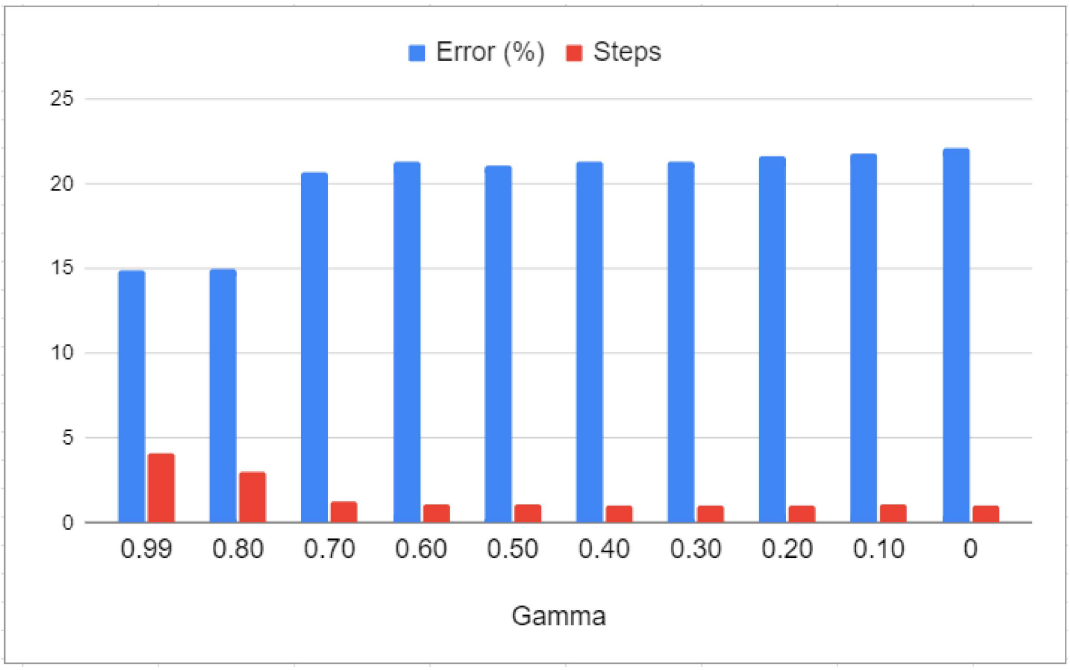


Figure 5.6: Comparing mean error and gamma results

– **Choice of Reward**

Tests were performed with different reward values to select the best reward strategy for the reinforcement learning agent. The maximum reward was set to 1 when the agent takes the correct action, while the reward was set to -1 when the agent took the wrong action. In addition, rewards were tested for when the agent decides to take another step because he is still not sure whether to choose the "is new" or "is not new" actions, with values of 0, -0.001, -0.01, -0.05, and -0.1. These tests were performed to evaluate which reward configuration maximizes the agent's performance in the learning environment.

The results of the tests carried out with different reward values are represented in the graph of Figure 5.7. The analysis of these results allowed us to conclude that the reward configuration that presented the best performance had a value of 0, contributing to reduce the agent's error in the learning environment at the expense of taking many steps. This conclusion indicates the importance of choosing the appropriate reward strategy in optimizing the performance of the reinforcement learning agent.

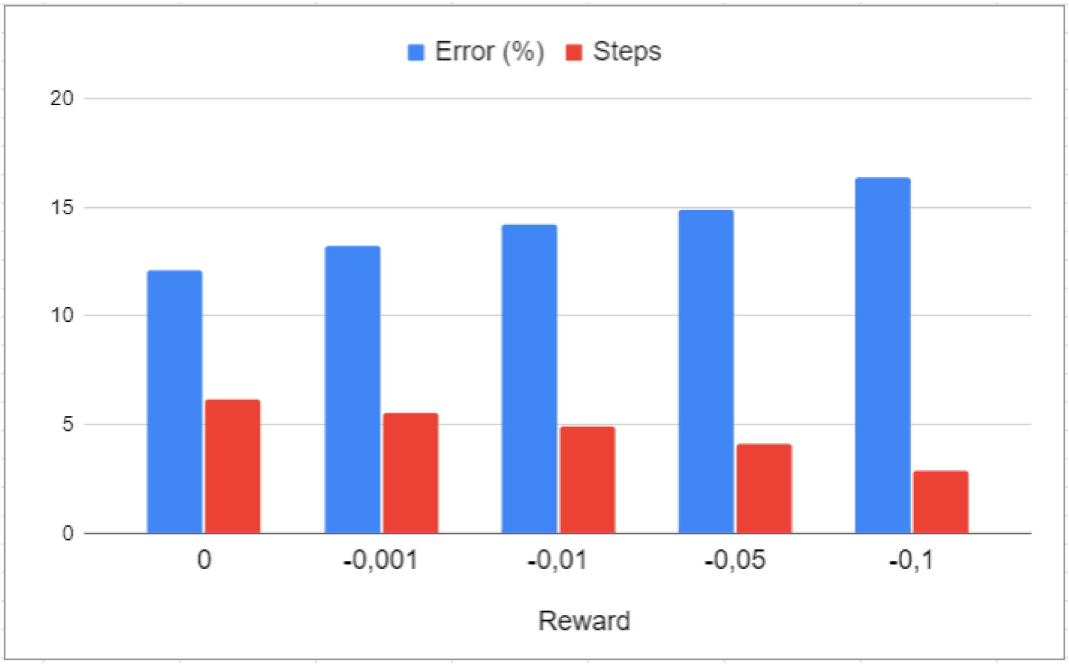


Figure 5.7: Comparing mean error and rewards results

In summary, in Figure 5.8 and Table 5.6, it is possible to compare the results of the baseline with the trained DRQN models, which were excellent. The results showed good accuracy and a reduced number of steps.

When calculating the confidence interval (CI) (109) of the accuracies, it can be noticed that the difference in accuracy between the best DRQN model ($\gamma$=0.99 and reward per step=0) and the best baseline model (which only uses the last images in the sequence) is significant, demonstrating the effectiveness of the DRQN model in progressively acquiring knowledge over time. Additionally, it is worth noting that the number of steps has decreased significantly. This indicates successful optimization of viewpoint planning while simultaneously increasing accuracy.

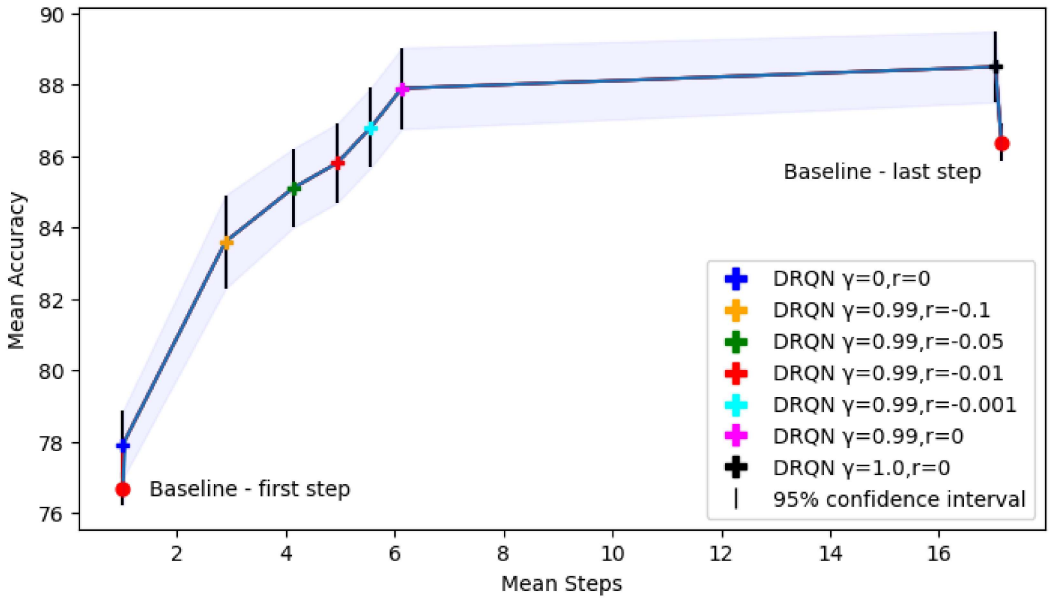


Figure 5.8: Comparison of Baseline and DRQN results. (r = reward per step)

Table 5.6: Comparison of Baseline and DRQN results.

| Models | Mean Steps | Mean Accuracy ($mean\ acc \pm CI$) |
|---|---|---|
| Baseline - First Step | 1 | 76.6±0.46 |
| DRQN $\gamma$= 0<br>reward per step= 0 | 1 | 77.9±0.95 |
| DRQN $\gamma$=0.99<br>reward per step= -0.1 | 2.89 | 83.6±1.31 |
| DRQN $\gamma$=0.99<br>reward per step= -0.05 | 4.13 | 85.1±1.11 |
| DRQN $\gamma$=0.99<br>reward per step= -0.01 | 4.94 | 85.8±1.11 |
| DRQN $\gamma$=0.99<br>reward per step= -0.001 | 5.56 | 86.8±1.12 |
| DRQN $\gamma$=0.99<br>reward per step= 0 | 6.13 | 87.9±1.14 |
| DRQN $\gamma$=1.0<br>reward per step= 0 | 17.04 | 88.5±0.99 |
| Baseline - Last Step | 17.15 | 86.3±0.52 |

## 5.3 Framework

Several tests were conducted in order to prove the efficiency and correct usage of the VIP framework, evaluating different scenarios and potential events that could occur. These include:

- Performance comparison of the VIP framework with and without the DRQN model;
- Analysis of SVM outputs in the VIP framework with and without the DRQN model;
- How the System Behaves in Case of Errors: Approaches and Solutions;
- Exploring the implementation of the VIP framework in real scenarios.

### 5.3.1 Performance comparison of the VIP framework with and without the DRQN model

Tests were conducted to compare the performance of viewpoint planning with and without the use of DRQN. Next, we present the results (Table 5.7), which include the mean time for each type of object (known and unknown) identified by the system as anomalies. These tests were conducted in five distinct environments, each containing different objects considered anomalies.

Table 5.7: Comparison of mean times (with a 95% confidence interval - CI) of inspection of each object with and without DRQN

| Types of objects | Mean time(s) without DRQN ($mean\ time \pm CI$) | Mean time(s) with DRQN ($mean\ time \pm CI$) |
|---|---|---|
| Unknown | 35.18±0.99 | 21.36±1.51 |
| Known | 35.46±0.94 | 3.02±0.40 |

When analyzing the results, it becomes evident that the inspection time per object is significantly reduced with the use of DRQN, especially when the object in question is known. The main difference lies in the fact that, even when the object of interest is already known by the system, the agent always completes a full revolution around the object, whereas with the use of DRQN, the agent does not require many steps to analyze the object, as it utilizes the SVM model trained with images of previously encountered objects as input. In other words, if the object belongs to the same class as one previously inspected, the analysis process will be much faster.

It is important to highlight that if an object is considered unknown, the SVM will be retrained by the Viewpoint Planning microservice (Section 3.1.5). This process typically takes about 4~30 seconds. If a new anomaly is found and needs to be inspected during the retraining, the system will wait for the completion of this training to be able to inspect the new object with the updated model. This waiting time was not taken into account in the results presented.

### 5.3.2 Analysis of SVM outputs in the VIP framework with and without the DRQN model

For a more comprehensive understanding of the impact of SVM outputs on the decision-making of DRQN within the framework, a series of tests were conducted.

Ten SVM models were trained, covering a variety of classes. Among these, ten objects were chosen for analysis purposes. These models were used to make inferences about the sequences of the selected objects.

The outputs generated by the SVM were evaluated in two contexts: first, up to the point at which the DRQN determines the "known" or "unknown" actions; second, considering no intervention from the DRQN, by analyzing the SVM outputs up to the last frame.

The entropies were calculated for all the images. The one with the lowest entropy was selected. Then, the SVM classifier was used to try to classify the image in order to calculate the accuracy.

Continuing with this logic, it was decided to also select the images that showed the greatest distance from the hyperplane defined by the SVM, and the last images in the sequence. This enabled a comparative analysis of accuracies among different approaches.

Comparing the results (Table 5.8), it can be observed that the accuracies consistently remain high at all times. However, when analyzing the last images without the use of DRQN, a significant variation is noticed. This is due to the presence of some images that are considerably different from the initial ones, leading to challenges in the classification task.

Furthermore, when examining the use of DRQN, we observed that the accuracy was slightly lower in the analysis of images with lower entropy and those farther from the hyperplane. However, this difference was not large, and considering that the use of DRQN results in a reduction in the number of required steps, it is still advantageous to use it.

Additionally, it is possible to conclude that the inputs to the DRQN (the SVM outputs) exhibited high quality, as the SVM displays metrics that are quite favorable.

Table 5.8: SVM mean accuracy results (with a 95% confidence interval - CI)

| Tests | Mean Accuracy, analyzing the minimum entropy image ($mean\ acc \pm CI$) | Mean Accuracy, analyzing the farthest image of the hyperplane ($mean\ acc \pm CI$) | Mean Accuracy, analyzing the last image ($mean\ acc \pm CI$) |
|---|---|---|---|
| With DRQN | 91.6±0.95 | 92.8±1.01 | 90.4±0.56 |
| Without DRQN | 94.1±1.33 | 96.5±1.21 | 86.4±2.08 |

### 5.3.3 How the system behaves in case of errors: Approaches and Solutions

Occasionally, the system may encounter errors. In this section, we will discuss the system's behavior in such situations and provide guidance on how to handle these errors. The errors that will be subject to analysis include:

- Novelty detection error;
- Viewpoint planning makes a misguided decision.

#### 5.3.3.1 Novelty detection error

The anomaly detector is an essential component of the system, responsible for determining whether the agent should interrupt the task of following the pipeline and examine the object in question. If the detector fails to detect any anomalies, no image of the object is recorded, and no entry is included in the report. This error can have a significant impact on the final outcome of the report.

Some tests were conducted in the simulated underwater environment, in which the agent was only supposed to determine whether there was or wasn't an anomaly in the images captured by the camera by performing inference with the trained autoencoder. In total, there were 112 frames, with half containing some anomaly and half without.

The obtained results are encouraging, with a false positive rate of only 6% and a false negative rate of 2%. It is preferable to have more false positives than false negatives, as it ensures that what truly matters is not being overlooked. The inclusion of less relevant information in the reports allows experts to analyze them later and determine the relevance of the data.

Analyzing some images (Figure 5.9) of objects that have been misclassified by the network, we observed that some of them are very small or have low contrast, which may have led them to be mistaken for the ground itself by the neural network.

Figure 5.9: Some images with objects that have been misclassified by the autoencoder

#### 5.3.3.2 Viewpoint planning makes a misguided decision

Viewpoint planning may make some mistakes, one of which is misclassifying a known object as unknown in its output. This could be due to an error propagated from the SVM (already calculated in 5.3.2) to the DRQN or solely an error from the DRQN (already calculated in 5.2.3). The following errors were verified:

- **Viewpoint planning is classified as unknown, when in fact it is known.**

  Several tests were conducted in order to measure the severity of these errors. If they occur, the SVM will be retrained with the current images, considering them as a new class. However, this approach may lead to less accurate SVM results.

  The tests were conducted as follows: 16 classes of images were randomly selected, and models were incrementally trained by adding one class at a time. Then, the same images were presented to the models again during training, but they were labeled as different classes from the previous ones, simulating possible errors in the system. In order to obtain more accurate results, the test was repeated 10 times, and the average accuracy was calculated.

  To achieve the results shown in Figure 5.10, when the images of the classes began to repeat, such as in the case of class 1 and class 18, which contained the same images, the accuracy evaluation during testing considered these classifications as correct when the SVM assigned either class 1 or 18 to these images.

These results indicate that the SVM model's performance slightly decreases as it is trained with images incorrectly labeled as new classes. This suggests that when the viewpoint planning fails to correctly classify an object as unknown, there is an increased tendency to make mistakes in the future.

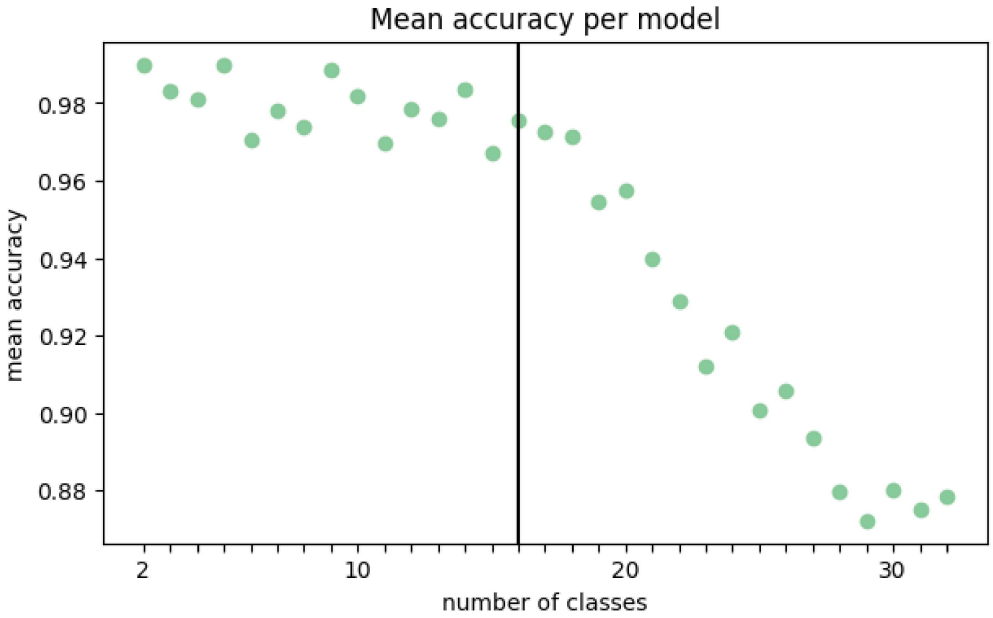


Figure 5.10: Accuracy per SVM model, repeating classes

To mitigate this issue, it would be beneficial, as future work, to implement an additional verification step, possibly by comparing the images from each folder created for the report with the new images, using cosine similarity (110). This would ensure that during the retraining process, the images truly belong to the same class.

– **Viewpoint planning misclassifies as known when it is actually unknown**

In the event that the situation arises where the object is not recognized, but the system incorrectly deems it as recognized, there will be no retraining of the SVM model. However, the captured images of the object will be stored appropriately for inclusion in the report, enabling the specialist to analyze them later.

For the next inspection, all the captured images for the report will be carefully analyzed and used as input for training a new SVM model. This way, when the new inspection begins, the SVM will be updated with prior knowledge of the objects to be encountered along the way.

### 5.3.4 Exploring the implementation of the VIP framework in real scenarios

With the aim of adapting the framework developed to operate in a simulated environment to a real-world setting, it becomes necessary to implement some significant changes. For instance, in the simulated environment, the agent moves at a fixed height relative to the duct. However, in a real-world environment, this height may be influenced by maritime currents, requiring a more sophisticated control than simple proportional control. Furthermore, while simulations do not account for collisions, in a real scenario, it is crucial to incorporate guidelines to avoid them, preventing damage to equipment.

In the real-world environment, the Autonomous Underwater Vehicle (AUV) will have more degrees of freedom to be controlled, necessitating a more robust control system, such as the implementation of a PID controller (Proportional, Integral, and Derivative), for example.

The quality of images will also be compromised, with lower brightness and sharpness, which may impact the effectiveness of the novelty detector. In this regard, the application of image pre-processing can be a viable strategy to mitigate the lack of focus.

Considering the high cost of equipment, it is essential to incorporate fault recovery routines. This ensures that the AUV does not become lost in the depths of the sea, preserving the integrity of the investments made.

Moreover, a crucial consideration lies in how and where the models will perform inferences. So far, the simulated environment has allowed for efficient use of NVIDIA V100 GPUs, enabling fast inferences. However, with the adoption of microservices, there is latency in information transfer, potentially resulting in system slowness, an undesirable situation in the real-world context.

To optimize efficiency in the real world, it would be advantageous for all models to reside in the AUV and perform inferences locally, eliminating the need to transmit information to the vessels for inference execution. This would reduce the risks of failures, such as data loss or delays in information transfer.

Table 5.9 compares inference times between an Intel Xeon CPU running at 2.20GHz and an NVIDIA V100 GPU. It is evident that GPU utilization substantially improves processing times. However, CPU times are not discouraging, suggesting that it is feasible to exclusively opt for the CPU for executing inferences on the AUV itself.

Table 5.9: Model inference time

| Models | CPU time (s) | GPU time (s) |
|---|---|---|
| Autoencoder | 0.1154 | 0.0583 |
| SVM | 0.0034 | 0.0025 |
| DRQN (1 step) | 0.1736 | 0.0629 |

# 6
# Conclusions

This work represents an advancement in the submarine oil industry, with the objective of automating processes that pose challenges to human perception.

In this work, a Visual Inspection of Pipelines (VIP) framework for an autonomous underwater vehicle to accomplish an underwater inspection was created. This system allows the agent to perform the inspection requiring only images from its camera, providing a significant economic advantage.

Through this system, the agent has the ability to approach the object of interest, aiming to enhance classification metrics. Furthermore, in case the object is unknown, it is possible to retrain the SVM model in order to enable its recognition in future encounters with similar objects.

To develop the system, an autoencoder network was trained to identify innovative features in the images captured by the autonomous vehicle's camera. Subsequently, the object's cropping in the same image will be used by the viewpoint planning (DRQN + SVM) if novelties are detected.

Several autoencoder architectures have been trained to find anomalies in subsea inspection images, for specialists to more easily find anomalies in inspection videos.

The choice of using autoencoders was motivated by the fact that most works use these networks to detect novelties, but they use grayscale images, unlike this work, which also uses RGB images. In addition to featuring a characteristic blue or green tonality, these images generally exhibit a blurred appearance.

A study was carried out on the best architecture to be used to find anomalies in images of different sizes and origins, and it can be concluded that dense autoencoders and convolutional autoencoders are good networks to find novelties in RGB images. However, the latter tends to generate noise that hinders the detection of anomalies.

Moreover, this work can evaluate the most suitable loss to be used in autoencoder networks aiming to efficiently detect anomalies, reaching the conclusion that utilizing SSIM+MSE is the optimal choice for a loss function.

Regarding the viewpoint planning, the results were excellent, successfully achieving the objective of optimizing viewpoint planning while showcasing the most advantageous balance between accuracy and the number of required steps. In other words, the DRQN approach outperformed the baseline, demon-

strating the ability to analyze the problem with a substantially smaller amount of images, resulting in significant time savings.

With these highly satisfactory results, reinforcement learning in viewpoint planning opens up new possibilities for optimizing underwater inspections and monitoring in the oil industry. Combining advanced technologies and intelligent decision-making systems brings us closer to achieving safer, more efficient, and cost-effective operations in the exploration and production of underwater resources. This allows camera-guided robots to increasingly extend their autonomy and overcome limitations present in traditional reinforcement learning algorithms.

In our future research, we aim to enhance the exploration of underwater image anomalies by leveraging advanced techniques such as Variational Adversarial Autoencoder (VAE). Our objective is to compare and contrast the results obtained from these methods with the findings presented in this work. Additionally, we plan to investigate the efficacy of other reinforcement learning algorithms, including Dueling DRQN and Deep Transformer Q-Network (111), in this domain. By exploring a diverse range of approaches, we hope to gain deeper insights and further contribute to the field.

Furthermore, as part of our future work, we have plans to utilize cosine similarity to minimize errors in selecting SVM training classes. This way, we can compare the images from each folder stored in the report with new images and have greater confidence that they belong to a new class not yet recognized by the SVM model.

With the aim of further enhancing and simplifying some sub-tasks within the framework, such as calculating the pipeline angle, we are exploring the feasibility of using a neural network to perform this task. Additionally, we are considering replacing the image binarization step in the pipeline with a neural network specialized in semantic segmentation. There are also plans to improve the task of circumnavigation control, which could be more accurate.

Another area of focus for improvements lies in expanding inspection tasks. This includes the ability to inspect larger areas, such as vegetation coverage, as well as incorporating leak inspection tasks. Moreover, we are looking to enhance existing tasks, such as object inspection, which currently primarily focuses on static objects.

There are also plans to make use of the ROV simulator located at Technip, which provides a higher level of realism and is geared towards professional use.

To be used in the real world, the VIP framework must be enhanced to improve the ability to dynamically adapt to the aquatic environment,

optimizing its trajectory and behavior in response to variable conditions. This will enable the AUV to handle unforeseen challenges such as ocean currents, underwater obstacles, and changes in environmental conditions.

# 7
# Bibliography


1 WU, Y. et al. Intelligent control method of underwater inspection robot in netcage. **Aquaculture Research**, Wiley Online Library, v. 53, n. 5, p. 1928–1938, 2022.

2 HE, J. et al. Multi-auv inspection for process monitoring of underwater oil transportation. **IEEE/CAA Journal of Automatica Sinica**, IEEE, v. 10, n. 3, p. 828–830, 2023.

3 WASZAK, M. et al. Semantic segmentation in underwater ship inspections: Benchmark and data set. **IEEE Journal of Oceanic Engineering**, IEEE, v. 48, n. 2, p. 462–473, 2022.

4 AMRI, S. S.; GHANI, A. A.; BAHARIN, M. K. Implementation of underwater image enhancement for corrosion pipeline inspection (uiecpi). In: IEEE. **2023 19th IEEE International Colloquium on Signal Processing & Its Applications (CSPA)**. [S.l.], 2023. p. 195–200.

5 JACOBI, M. Autonomous inspection of underwater structures. **Robotics and Autonomous Systems**, Elsevier, v. 67, p. 80–86, 2015.

6 HOU, S. et al. Underwater inspection of bridge substructures using sonar and deep convolutional network. **Advanced Engineering Informatics**, Elsevier, v. 52, p. 101545, 2022.

7 STENIUS, I. et al. A system for autonomous seaweed farm inspection with an underwater robot. **Sensors**, MDPI, v. 22, n. 13, p. 5064, 2022.

8 BESADA, J. A. et al. Drone mission definition and implementation for automated infrastructure inspection using airborne sensors. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 18, n. 4, p. 1170, 2018.

9 DORAFSHAN, S.; MAGUIRE, M. Bridge inspection: human performance, unmanned aerial systems and automation. **Journal of Civil Structural Health Monitoring**, Springer, v. 8, n. 3, p. 443–476, 2018.

10 MA, Y. et al. Path planning of uuv based on hqpso algorithm with considering the navigation error. **Ocean Engineering**, Elsevier, v. 244, p. 110048, 2022.

11 HE, J. et al. Uuv path planning for collision avoidance based on ant colony algorithm. In: IEEE. **2020 39th Chinese Control Conference (CCC)**. [S.l.], 2020. p. 5528–5533.

12 GALYAEV, A. A. et al. Path planning in threat environment for uuv with non-uniform radiation pattern. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 7, p. 2076, 2020.

13 MCCLAIN, R. T. **Feasibility of using active fiducial markers for uuv navigation and localization**. Tese (Doutorado) — Monterey, CA; Naval Postgraduate School, 2019.

14 CHO, H. et al. Study on control system of integrated unmanned surface vehicle and underwater vehicle. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 9, p. 2633, 2020.

15 MCLEOD, D. et al. Autonomous inspection using an underwater 3d lidar. In: IEEE. **2013 OCEANS-San Diego**. [S.l.], 2013. p. 1–8.

16 MATTERN, J. et al. Underwater navigation using 3d vision, edge processing, and autonomy. In: IEEE. **OCEANS 2021: San Diego–Porto**. [S.l.], 2021. p. 1–6.

17 ZHANG, W. et al. Visual location method based on asymmetric guiding light array in uuv recovery progress. In: IEEE. **2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)**. [S.l.], 2019. p. 2671–2675.

18 YAN, Z. et al. Autonomous underwater vehicle vision guided docking experiments based on l-shaped light array. **IEEE Access**, IEEE, v. 7, p. 72567–72576, 2019.

19 GRIGORESCU, S. et al. A survey of deep learning techniques for autonomous driving. **Journal of Field Robotics**, Wiley Online Library, v. 37, n. 3, p. 362–386, 2020.

20 AHMAD, S. et al. Autoencoder-based condition monitoring and anomaly detection method for rotating machines. In: IEEE. **2020 IEEE International Conference on Big Data (Big Data)**. [S.l.], 2020. p. 4093–4102.

21 RUSSO, S. et al. Anomaly detection using deep autoencoders for in-situ wastewater systems monitoring data. **arXiv preprint arXiv:2002.03843**, 2020.

22 JEONG, S.-K. et al. A study on anomaly detection of unmanned marine systems using machine learning. **Measurement and Control**, SAGE Publications Sage UK: London, England, v. 56, n. 3-4, p. 470–480, 2023.

23 ROKA, S.; DIWAKAR, M. Deep stacked denoising autoencoder for unsupervised anomaly detection in video surveillance. **Journal of Electronic Imaging**, Society of Photo-Optical Instrumentation Engineers, v. 32, n. 3, p. 033015–033015, 2023.

24 CHANG, Y. et al. Clustering driven deep autoencoder for video anomaly detection. In: SPRINGER. **Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16**. [S.l.], 2020. p. 329–345.

25 CHOW, J. K. et al. Anomaly detection of defects on concrete structures with the convolutional autoencoder. **Advanced Engineering Informatics**, Elsevier, v. 45, p. 101105, 2020.

26 NGUYEN, H. D. et al. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. **International Journal of Information Management**, Elsevier, v. 57, p. 102282, 2021.

27 GAO, H. et al. Tsmae: a novel anomaly detection approach for internet of things time series data using memory-augmented autoencoder. **IEEE Transactions on network science and engineering**, IEEE, 2022.

28 LEE, H. et al. Autonomous underwater vehicle control for fishnet inspection in turbid water environments. **International Journal of Control, Automation and Systems**, Springer, v. 20, n. 10, p. 3383–3392, 2022.

29 LEWIS, A. et al. Virtual planning and testing of auv paths for underwater photogrammetry. In: **VISIGRAPP (1: GRAPP)**. [S.l.: s.n.], 2020. p. 93–101.

30 ZACCHINI, L.; FRANCHI, M.; RIDOLFI, A. Sensor-driven autonomous underwater inspections: A receding-horizon rrt-based view planning solution for auvs. **Journal of Field Robotics**, Wiley Online Library, v. 39, n. 5, p. 499–527, 2022.

31 PALOMERAS, N. et al. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. **IEEE Robotics and Automation Letters**, IEEE, v. 4, n. 2, p. 1619–1625, 2019.

32 WU, J. et al. Multi-auv motion planning for archeological site mapping and photogrammetric reconstruction. **Journal of Field Robotics**, Wiley Online Library, v. 36, n. 7, p. 1250–1269, 2019.

33 MABOUDI, M. et al. A review on viewpoints and path planning for uav-based 3d reconstruction. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, IEEE, 2023.

34 PEUZIN-JUBERT, M. et al. Survey on the view planning problem for reverse engineering and automated control applications. **Computer-Aided Design**, Elsevier, v. 141, p. 103094, 2021.

35 BOLOURIAN, N.; HAMMAD, A. Lidar-equipped uav path planning considering potential locations of defects for bridge inspection. **Automation in Construction**, Elsevier, v. 117, p. 103250, 2020.

36 MENON, R.; ZAENKER, T.; BENNEWITZ, M. Viewpoint planning based on shape completion for fruit mapping and reconstruction. **arXiv preprint arXiv:2209.15376**, 2022.

37 ZENG, J. et al. Efficient view path planning for autonomous implicit reconstruction. In: IEEE. **2023 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.], 2023. p. 4063–4069.

38 LODEL, M. et al. Where to look next: Learning viewpoint recommendations for informative trajectory planning. In: IEEE. **2022 International Conference on Robotics and Automation (ICRA)**. [S.l.], 2022. p. 4466–4472.

39 YANG, D. et al. Side-scan sonar image segmentation based on multi-channel cnn for auv navigation. **Frontiers in Neurorobotics**, Frontiers Media SA, v. 16, p. 928206, 2022.

40 TANG, Y. et al. Real-time processing and high-quality imaging of navigation strip data using sss based on auvs. **Journal of Marine Science and Engineering**, MDPI, v. 11, n. 9, p. 1769, 2023.

41 WANG, Z. et al. Acoustic communication and imaging sonar guided auv docking: system infrastructure, docking methodology and lake trials. **Control Engineering Practice**, Elsevier, v. 136, p. 105529, 2023.

42 ZHANG, Y.; GE, W. Auv path planning and image recognition based on convolutional neural network. In: IEEE. **2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)**. [S.l.], 2022. p. 605–608.

43 GUTNIK, Y. et al. On the adaptation of an auv into a dedicated platform for close range imaging survey missions. **Journal of Marine Science and Engineering**, Multidisciplinary Digital Publishing Institute, v. 10, n. 7, p. 974, 2022.

44 TAYE, M. M. Theoretical understanding of convolutional neural network: concepts, architectures, applications, future directions. **Computation**, MDPI, v. 11, n. 3, p. 52, 2023.

45 UNIVERSITY of Standford. Convolutional Neural Networks for Visual Recognition. <http://cs231n.github.io/convolutional-networks/>. Accessed: 2023-06-30.

46 SZEGEDY, C. et al. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9.

47 GUO, Y. et al. Deep learning for visual understanding: A review. **Neurocomputing**, Elsevier, v. 187, p. 27–48, 2016.

48 YU, T.; ZHU, H. Hyper-parameter optimization: A review of algorithms and applications. **arXiv preprint arXiv:2003.05689**, 2020.

49 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

50 ZIMEK, A.; SCHUBERT, E. Outlier detection, encyclopedia of database systems. **Springer**, v. 10, p. 978–1, 2017.

51 PANG, G. et al. Deep learning for anomaly detection: A review. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 54, n. 2, p. 1–38, 2021.

52 PIMENTEL, M. A. et al. A review of novelty detection. **Signal processing**, Elsevier, v. 99, p. 215–249, 2014.

53 BERROUKHAM, A. et al. Deep learning-based methods for anomaly detection in video surveillance: a review. **Bulletin of Electrical Engineering and Informatics**, v. 12, n. 1, p. 314–327, 2023.

54 PATRIKAR, D. R.; PARATE, M. R. Anomaly detection using edge computing in video surveillance system. **International Journal of Multimedia Information Retrieval**, Springer, v. 11, n. 2, p. 85–110, 2022.

55 SULTANI et al. Real-world anomaly detection in surveillance videos. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018.

56 ZHU et al. Video anomaly detection for smart surveillance. In: . [S.l.: s.n.], 2020.

57 SAEEDI, J.; GIUSTI, A. Anomaly detection for industrial inspection using convolutional autoencoder and deep feature-based one-class classification. In: **VISIGRAPP (5: VISAPP)**. [S.l.: s.n.], 2022. p. 85–96.

58 JIN, P. et al. Anomaly detection in aerial videos with transformers. **IEEE Transactions on Geoscience and Remote Sensing**, IEEE, v. 60, p. 1–13, 2022.

59 LU, B.; XU, D.; HUANG, B. Deep-learning-based anomaly detection for lace defect inspection employing videos in production line. **Advanced Engineering Informatics**, Elsevier, v. 51, p. 101471, 2022.

60 CONTRERAS-CRUZ; AL, M. A. et. Vision-based novelty detection using deep features and evolved novelty filters for specific robotic exploration and inspection tasks. In: . [S.l.: s.n.], 2019.

61 TSCHUCHNIG, M. E.; GADERMAYR, M. Anomaly detection in medical imaging-a mini review. In: SPRINGER. **Data Science–Analytics and Applications: Proceedings of the 4th International Data Science Conference–iDSC2021**. [S.l.], 2022. p. 33–38.

62 LE, K. H. et al. Learning from multiple expert annotators for enhancing anomaly detection in medical image analysis. **IEEE Access**, IEEE, v. 11, p. 14105–14114, 2023.

63 HANSEN, S. et al. Anomaly detection-inspired few-shot medical image segmentation through self-supervision with supervoxels. **Medical Image Analysis**, Elsevier, v. 78, p. 102385, 2022.

64 ZHANG, H. et al. Unsupervised deep anomaly detection for medical images using an improved adversarial autoencoder. **Journal of Digital Imaging**, Springer, v. 35, n. 2, p. 153–161, 2022.

65 TIAN, Y. et al. Unsupervised anomaly detection in medical images with a memory-augmented multi-level cross-attentional masked autoencoder. **arXiv preprint arXiv:2203.11725**, 2022.

66 ALAMRI, M.; YKHLEF, M. Survey of credit card anomaly and fraud detection using sampling techniques. **Electronics**, MDPI, v. 11, n. 23, p. 4003, 2022.

67 JIANG, S. et al. Credit card fraud detection based on unsupervised attentional anomaly detection network. **Systems**, MDPI, v. 11, n. 6, p. 305, 2023.

68 HEMDAN, E. E.-D.; MANJAIAH, D. Anomaly credit card fraud detection using deep learning. **Deep Learning in Data Analytics: Recent Techniques, Practices and Applications**, Springer, p. 207–217, 2022.

69 POURHABIBI; AL, T. et. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. In: . [S.l.: s.n.], 2020.

70 AZZOUAZI; MOHAMED. Anomaly detection in credit card transactions. In: . [S.l.: s.n.], 2019.

71 KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of artificial intelligence research**, v. 4, p. 237–285, 1996.

72 WATKINS, C. J.; DAYAN, P. Q-learning. **Machine learning**, Springer, v. 8, n. 3-4, p. 279–292, 1992.

73 MNIH, V. et al. Playing atari with deep reinforcement learning. **arXiv preprint arXiv:1312.5602**, 2013.

74 LILLICRAP, T. P. et al. Continuous control with deep reinforcement learning. **arXiv preprint arXiv:1509.02971**, 2015.

75 BELLMAN, R.; KALABA, R. Dynamic programming and statistical communication theory. **Proceedings of the National Academy of Sciences of the United States of America**, National Academy of Sciences, v. 43, n. 8, p. 749, 1957.

76 SUTTON, R. S.; BARTO, A. G. Reinforcement Learning: An Introduction. p. 352.

77 BUSONIU, L. et al. **Reinforcement learning and dynamic programming using function approximators**. [S.l.]: CRC press, 2017.

78 HAUSKNECHT, M.; STONE, P. Deep recurrent q-learning for partially observable mdps. In: **2015 aaai fall symposium series**. [S.l.: s.n.], 2015.

79 NGUYEN, V. et al. A Survey on Adaptive Multi-Channel MAC Protocols in VANETs Using Markov Models. **IEEE Access**, v. 6, p. 16493–16514, 2018. ISSN 2169-3536. Disponível em: <http://ieeexplore.ieee.org/document/8314144/>.

80 MNIH, V. et al. Human-level control through deep reinforcement learning. **nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

81 CLARK-TURNER, M.; BEGUM, M. Deep recurrent q-learning of behavioral intervention delivery by a robot from demonstration data. In: IEEE. **2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)**. [S.l.], 2017. p. 1024–1029.

82 LIU, F. et al. Research on path planning of robot based on deep reinforcement learning. In: IEEE. **2020 39th Chinese Control Conference (CCC)**. [S.l.], 2020. p. 3730–3734.

83 HUANG, C. Y. Financial trading as a game: A deep reinforcement learning approach. **arXiv preprint arXiv:1807.02787**, 2018.

84 ZHOU, P.; TANG, J. Improved method of stock trading under reinforcement learning based on drqn and sentiment indicators arbr. **arXiv preprint arXiv:2111.15356**, 2021.

85 YAN, Z. et al. An obstacle avoidance algorithm for auv based on obstacle's detected outline. In: IEEE. **2018 37th Chinese Control Conference (CCC)**. [S.l.], 2018. p. 5257–5262.

86 REHMAN, H. U.; MIURA, J. Viewpoint planning for automated fruit harvesting using deep learning. In: IEEE. **2021 IEEE/SICE International Symposium on System Integration (SII)**. [S.l.], 2021. p. 409–414.

87 SATHER, J.; ZHANG, X. J. Viewpoint optimization for autonomous strawberry harvesting with deep reinforcement learning. **arXiv preprint arXiv:1903.02074**, 2019.

88 CALLI, B. et al. Viewpoint optimization for aiding grasp synthesis algorithms using reinforcement learning. **Advanced Robotics**, Taylor & Francis, v. 32, n. 20, p. 1077–1089, 2018.

89 CALLI, B. et al. Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 15, n. 4, p. 1810–1822, 2018.

90 CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, p. 273–297, 1995.

91 WANG, P.; FAN, E.; WANG, P. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. **Pattern Recognition Letters**, Elsevier, v. 141, p. 61–67, 2021.

92 KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Forward-backward error: Automatic detection of tracking failures. In: IEEE. **2010 20th international conference on pattern recognition**. [S.l.], 2010. p. 2756–2759.

93 LA, H. M. et al. Development of an autonomous bridge deck inspection robotic system. **Journal of Field Robotics**, Wiley Online Library, v. 34, n. 8, p. 1489–1504, 2017.

94 MENENDEZ, E. et al. Tunnel structural inspection and assessment using an autonomous robotic system. **Automation in Construction**, Elsevier, v. 87, p. 117–126, 2018.

95 LI, X. et al. On optimizing autonomous pipeline inspection. **IEEE Transactions on Robotics**, IEEE, v. 28, n. 1, p. 223–233, 2011.

96 CHA, Y.-J. et al. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. **Computer-Aided Civil and Infrastructure Engineering**, Wiley Online Library, v. 33, n. 9, p. 731–747, 2018.

97 SHAH, S. et al. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: SPRINGER. **Field and service robotics**. [S.l.], 2018. p. 621–635.

98 SANDERS, A. **An introduction to Unreal engine 4**. [S.l.]: AK Peters/CRC Press, 2016.

99 CHEN, Z. et al. Autoencoder-based network anomaly detection. In: IEEE. **2018 Wireless Telecommunications Symposium (WTS)**. [S.l.], 2018. p. 1–5.

100 BROCKMAN, G. et al. Openai gym. **arXiv preprint arXiv:1606.01540**, 2016.

101 KELLY, S.; KELLY, S. Basic introduction to pygame. **Python, PyGame and Raspberry Pi Game Development**, Springer, p. 59–65, 2016.

102 AIRSIM. https://microsoft.github.io/airsim. In: . [S.l.: s.n.], Accessed: 2023–06–22.

103 BOUTARFASS; SANAE; BESSERER., B. Convolutional autoencoder for discriminating handwriting styles. In: . [S.l.: s.n.], 2019.

104 CARASS, A. et al. Evaluating white matter lesion segmentations with refined sørensen-dice analysis. **Scientific reports**, Nature Publishing Group, v. 10, n. 1, p. 1–19, 2020.

105 TENSORFLOW. https://www.tensorflow.org. In: . [S.l.: s.n.], Accessed: 2023–06–22.

106 MAATEN, L. Van der; HINTON, G. Visualizing data using t-sne. **Journal of machine learning research**, v. 9, n. 11, 2008.

107 PLATT, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. **Advances in large margin classifiers**, Cambridge, MA, v. 10, n. 3, p. 61–74, 1999.

108 KRAMER, O.; KRAMER, O. Scikit-learn. **Machine learning for evolution strategies**, Springer, p. 45–53, 2016.

109 SMITHSON, M. **Confidence intervals**. [S.l.]: Sage, 2003.

110 XIA, P.; ZHANG, L.; LI, F. Learning similarity with cosine similarity ensemble. **Information sciences**, Elsevier, v. 307, p. 39–52, 2015.

111 ESSLINGER, K.; PLATT, R.; AMATO, C. Deep transformer q-networks for partially observable reinforcement learning. **arXiv preprint arXiv:2206.01078**, 2022.