

Automatic Detection of Unknown Classes of Vessels using
Specialized Machine Learning Models in Passive Sonar Systems

Eduardo Sperle Honorato

Final Undergraduate Project submitted to
the Electronics and Computer Engineering
Course, Polytechnic School, at Federal
University of Rio de Janeiro, as a partial
fulfillment of the requirements for getting
the degree in Electronics and Computer
Engineering.

Advisor: João Baptista de Oliveira e Souza
Filho

Rio de Janeiro
December, 2022

Automatic Detection of Unknown Classes of Vessels using
Specialized Machine Learning Models in Passive Sonar Systems

Eduardo Sperle Honorato

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO
DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA PO-
LITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU
DE ENGENHEIRO EM ELETRÔNICA E DE COMPUTAÇÃO

Autor:

Eduardo Sperle Honorato
Eduardo Sperle Honorato

Orientador:

João Baptista de Oliveira e Souza Filho
João Baptista de Oliveira e Souza Filho, D. Sc

Examinador:

Heraldo Luis S. Almeida
Prof. Heraldo Luis Silveira de Almeida, D.Sc.

Examinador:

Rogério Pinto Espíndola
Prof. Rogério Pinto Espíndola, D.Sc.

Examinador:

Sergio Pinto Gomes Junior
Eng. Sergio Pinto Gomes Junior, M.Sc.

Rio de Janeiro

December, 2022

Declaração de Autoria e de Direitos

Eu, *Eduardo Sperle Honorato* CPF 145.205.267-01, autor da monografia *Automatic detections of unknown classes by specialized models in passive sonar systems*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e idéias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.
5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.
7. Por ser verdade, firmo a presente declaração.

Eduardo Sperle Honorato
Eduardo Sperle Honorato

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfiltrar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

ABSTRACT

Sonar operators are of great importance for identifying potential threats to submarines, making use of underwater acoustic signatures obtained by passive sonar systems. Automatic classification models can alleviate the physical and mental burden of sonar operators improving their job performance. However, the development of these models often requires a database containing a large number of vessels, which is impractical due to the related operational costs and the confidential nature of data of this kind. Thus, it is necessary to develop models able to differentiate between known and unknown classes of vessels. In particular, this work proposes to address this task using class-specialized novelty detectors. It encompasses a comprehensive study, including Machine Learning techniques commonly adopted in the literature and some others not so well explored, such as the specialized Siamese Neural Networks. The experiments include techniques such as Neural Networks, Gaussian Mixture, k -means, k NN, OneClass SVM, among others, in general and class-specialized topologies. For this, we explored a dataset provided by the IPqM, integrating 8 classes of vessels. The figure of merit used to evaluate the techniques was the Area Under Curve (AUC) relative to the Receiver operating characteristic (ROC) curve. The best approach was a class-specialized arquitecture based on the Gaussian Mixture using a diagonal-type covariance matrix, which attained a value of AUC of 0.9602 for a simulation scenario with three known classes.

Key-words: Novelty detection, Passive Sonar System, Machine Learning.

RESUMO

Os operadores de sonar são de grande importância para identificar potenciais ameaças aos submarinos, fazendo uso de assinaturas acústicas subaquáticas obtidas por sistemas de sonar passivo. Os modelos de classificação automática podem aliviar a carga física e mental dos operadores de sonar, melhorando seu desempenho no trabalho. No entanto, o desenvolvimento desses modelos muitas vezes requer um banco de dados contendo um grande número de embarcações, o que é inviável devido aos custos operacionais relacionados e ao caráter confidencial de dados desse tipo. Assim, é necessário desenvolver modelos capazes de diferenciar entre classes conhecidas e desconhecidas de embarcações. Em particular, este trabalho propõe abordar esta tarefa usando detectores de novidades especializados em classes. É realizado um estudo abrangente, que contempla técnicas de Aprendizado de Máquina comumente adotadas na literatura e algumas outras pouco exploradas, como as Redes Neurais Siamesas especializadas. Os experimentos incluem técnicas como Redes Neurais, Mistura de Gaussianas, k -means, k NN, OneClass SVM, entre outras, em topologias de classes gerais e especializadas. Para isso, exploramos um conjunto de dados fornecido pelo IPqM, que integra 8 classes de embarcações. A figura de mérito utilizada para avaliar as técnicas foi a Área sob a Curva (AUC) em relação à curva ROC (Receiver Operating Characteristic). A melhor abordagem foi uma arquitetura classe-especializada baseada em Mistura de Gaussianas usando uma matriz de co-variância do tipo diagonal, que atingiu um valor de AUC de 0.9602 para um cenário de simulação com três classes conhecidas.

ACRONYMS

*k*NN - k-Nearest Neighbors

LOF - Local Outlier Factor

ACS - Automatic Classification Systems

NNd - Nearest Neighbor Density

IPqM - Instituto de Pesquisas da Marinha Brasileira (Brazilian Navy Research Institute)

TPSW - Two-Pass Split Windows

ODIN - Out-of-DIstribution detector for Neural networks

LSTM - Long Short-Term Memory

PCA - Principal Component Analysis

KPCA - Kernel Principal Component Analysis

SVM - Support Vector Machine

GMM - Gaussian Mixture Models

SNN - Siamese Neural Network

SONAR - SOund NAvigation and Ranging

ROC - Receiver Operating Characteristic

AUC - Area under ROC curve

NSD - Non-Specialized Detector

Index

1	Introduction	1
2	Novelty detection	3
2.1	Novelty	3
2.2	Novelty Detection	4
2.3	Class-specialized Topologies for Novelty Detection	5
2.3.1	The Non-specialized topology	6
2.3.2	The Hierarchical topology	6
2.3.3	The Unanimous topology	6
2.4	Machine Learning techniques	7
2.4.1	k NN	8
2.4.2	LOF	9
2.4.3	NNd	11
2.4.4	One-Class SVM	11
2.4.5	KPCA	12
2.4.6	k -Means	13
2.4.7	Gaussian Mixture Models	13
2.4.8	Autoencoder	14
2.4.9	Siamese Neural Networks	18
2.4.10	ODIN	22
3	Materials and Methods	24
3.1	Dataset	24
3.2	Resampling	26
3.3	Emulating the Unknown Classes	26

3.4	Figures of merit	27
3.4.1	Accuracy	27
3.4.2	AUC	27
3.5	Tuning the novelty detector model hyperparameters	29
3.6	Decision threshold setting process	30
3.7	Statistical Analysis	30
3.8	Experimental Computational Environment	30
4	Results	32
4.1	General chapter organization	32
4.2	Distance-based techniques	34
4.3	Mixture-models-based techniques	35
4.4	Neural-Networks-based techniques	42
4.5	Reconstruction and Domain-based techniques	51
4.6	Comparison of the best performing methods	51
5	Conclusion	56
Bibliography		58
A	Hyperparameters' selection	64
A.1	<i>k</i> NN classifier	64
A.2	Non-neural models	65
A.3	Neural-Networks-based models	65
A.3.1	LSTM Autoencoder	65
A.3.2	Siamese Neural Networks	66
A.4	ODIN	67
B	Publications	70

Figures Index

2.1	Scores distribution.	5
2.2	General novelty detection architecture(extracted from [1]).	5
2.3	Hierarchical novelty detection system architecture (extracted from [1]). . .	7
2.4	Unanimous novelty detection system architecture (extracted from [1]). . .	7
2.5	Demonstration of k NN (adapted from [2]).	9
2.6	Local density of two different data instances (extracted from [3])	10
2.7	Illustration of the different assumptions over a hypothetical distribution(extracted from [4]	15
2.8	Symmetric auto-encoder architecture (adapted from [5])	16
2.9	Representation of the LSTM cell (extracted from [6])	17
2.10	An illustration of SNN architecture(extracted from [7])	19
2.11	An illustration of the One Against All strategy for triplet formation. . . .	21
2.12	An illustration of the One Against One strategy for triplet formation. . . .	21
3.1	Diagram of the digital signal processing chain(Adapted from [8]).	25
3.2	Spectrogram for an arbitrary database run,	25
4.1	Demonstration of how the smallest p-value is obtained when comparing topologies of the same technique (see text).	34
4.2	Boxplot diagram for the Three-known-class scenario considering the distance-based novelty detection scores.	36
4.3	Boxplot diagram for the Five-known-classes scenario considering the distance-based novelty detection scores.	37
4.4	Boxplot diagram for the Seven-known-classes scenario considering the distance-based novelty detection scores.	38

4.5	Boxplot diagram for the three and Five-known-classes' scenario considering the mixture-models-based detection scores.	40
4.6	Boxplot diagram for the Seven-known-classes scenario considering the mixture-models-based detection scores.	41
4.7	Results for the Neural-Network-based novelty detection techniques (part I).	44
4.8	Results for the Neural-Network-based novelty detection techniques (part II).	45
4.9	Results for the Neural-Network-based novelty detection techniques (part III).	46
4.10	Results for the Neural-Network-based novelty detection techniques (part IV).	47
4.11	Results for the Neural-Network-based novelty detection techniques (part V).	48
4.12	Results for the Neural-Network-based novelty detection techniques (part VI).	49
4.13	Boxplot diagram for the three and Five-known-classes scenario considering the reconstruction and domain novelty detection scores. . . .	52
4.14	Boxplot diagram for the Seven-known-classes scenario considering the reconstruction and domain novelty detection scores.	53
4.15	Boxplot diagram comparing the best performing methods identified for each group (part I).	54
4.16	Boxplot diagram comparing the best performing methods identified for each group (part II).	55

Tables Index

3.1	Number of spectral windows for each class.	26
3.2	Set of known and unknown classes identified to each evaluation scenario (see text).	27
4.1	Medians and p -values for the comparisons between multiple techniques and topologies for the distance-based models case (see text).	39
4.2	Medians and p -values for the comparisons between multiple techniques and topologies for the mixture-models case (see text).	42
4.3	Medians and p -values for the comparisons between multiple techniques and topologies for the Neural-Networks-based models case (see text).	50
4.4	Medians and p -values for the comparisons between multiple techniques and topologies for the reconstruction and domain based models case (see text).	51
A.1	Hyperparameters investigated and chosen for the k NN classifier.	64
A.2	Hyperparameters investigated and chosen for the three-known-class scenario.	65
A.3	Hyperparameters investigated and chosen for the five-known-classes scenario.	66
A.4	Hyperparameters investigated and chosen for the seven-known-classes scenario.	66
A.5	Hyperparameters chosen for the LSTM Autoencoder.	67
A.6	Hyperparameters chosen for the SNNa architecture.	68
A.7	Hyperparameters chosen for the SNNb architecture.	69

A.8 SNN model (SNNa or SNNb) chosen for each evaluation scenario in the case of the SNNc approach.	69
A.9 ODIN hyperparameters for each evaluation scenario.	69

Chapter 1

Intoduction

The passive sonar system is frequently used by submarines to monitor aquatic noises in military patrol activities. Through this system, the vibrations emitted by propulsion systems of other vessels are captured by an array of hydrophones and processed to extract their tonal and spectral characteristics. Subsequently, this information is analyzed by trained professionals to classify the vessel's acoustic signature.

The automation of this process is relevant, as it can reduce the physical and mental burden of the operator, turning easier the surveillance effort while increasing the reliability of the process of decision making. Typically, automatic classification systems (ACS) are built considering a limited number of classes of vessels. Thus, the existence of mechanisms for identifying those not contemplated during the system's development is of paramount importance. In other words, ACS must be capable of detecting novelties in this context.

This work aims to discuss the development of highly efficient and reliable detectors of unknown classes of vessels, considering the exploitation of class-specialized subsystems. The Machine Learning models considered for identifying new scenarios do not make use of any information about the unknown concept. Therefore, the strategy adopted by the models is inferring how discrepant a new scenario is from those already known, seeking to operate with an attractive trade-off between the rates of recognition of what is new and what is unknown.

The structure of this work is the following: Chapter 2 introduces the concepts of novelty detection and briefly covers the Machine Learning techniques considered in this work. Chapter 3 describes the dataset used in the experiments and the figures of merit exploited for evaluating the models. Chapter 4 presents the results. Finally, in Chapter 5, the conclusions are discussed. Appendix A discusses in more detail the process of hyperparameter selection and the training strategies considered for the Machine Learning techniques. Appendix B summarizes the publications associated with the development of this work.

Chapter 2

Novelty detection

The objective of this chapter is to briefly contextualize the process of novelty detection. In other words, how to distinguish if a given sample represents a novelty or not. For that, first, the definition of novelty and novelty detection is given. In the following, some class-specialized topologies for novelty detection and the Machine Learning techniques to address this problem are covered.

2.1 Novelty

The novelty detection is essentially a pattern recognition task. In this work's context, it corresponds to recognizing if some data provenient from an arbitrary class is not present in the original dataset. Naturally, for a novelty occurrence, the features observed must be sufficiently different from those from other dataset classes. The literature points out two terms often confused with novelty detection: anomaly and outliers [9].

Novelty refers to something new, for instance, a class of vessel never seen before. An anomaly or an outlier refers to samples that are not necessarily new but that exhibit a significant discrepancy with respect to those related to some process of interest. The main aspect behind an anomaly or an outlier is the idea of a pattern that is not so common, for example a wrong measurement or some manufacturing process failure.

Despite the conceptual differences between novelty detection, anomaly and outlier detections, the algorithmic procedure is basically the same, consisting of producing a dissimilarity score and comparing it with some previous established threshold.

2.2 Novelty Detection

As said before, the novelty detection is basically the act of comparing a novelty score with a decision threshold [9]. The decision threshold may be easily obtained by evaluating this novelty score over the database, and setting it to match some specific percentile from the experimental novelty score distribution. Thus, samples whose novelty scores are greater than the decision threshold will be assumed as novelties, otherwise as known.

Naturally, the decision threshold must be chosen considering the fact that some percentage of the dataset instances will be wrongly classified as novelties. For instance, if it is chosen a decision threshold matching the 95° percentile of the dataset novelty score distribution, a total of 5% of such instances will be wrongly classified as novelties.

The relation between the decision threshold choice and the novelty detection performance can be better understood through the graphical example depicted in Figure 2.1. In this figure, there are two distributions of some novelty score. The blue one represents the values obtained with the known dataset; while the red, with the unknown dataset. The purple region between both distributions represents errors regarding novelty detection. If the decision threshold is moved to the right, the number of unknown samples wrongly classified as known will increase. The same goes if the decision threshold is moved to the opposite direction, leading the number of known samples wrongly classified as novelties to increase.

A novelty detector is composed by a novelty score generator, a decision threshold, and a decision maker. This architecture can be seen in Figure 2.2. Basically, the novelty score generator is a mathematical model that receives some pre-processed data as input and generates a novelty score. This novelty score s is then compared

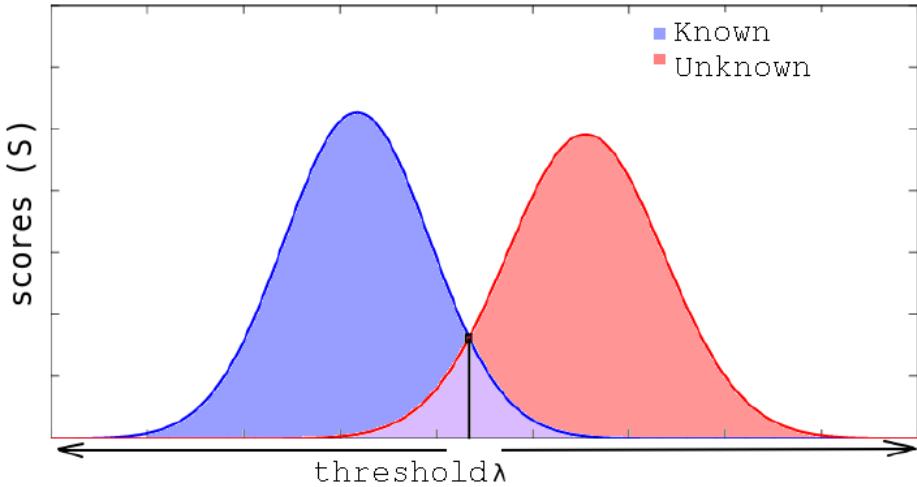


Figure 2.1: Scores distribution

with a decision threshold λ integrating the decision-maker block, whose output must be one for novelties and zero for non-novelties.

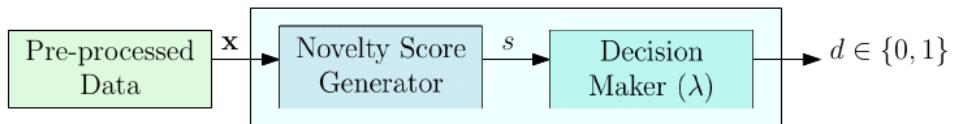


Figure 2.2: General novelty detection architecture (extracted from [1]).

2.3 Class-specialized Topologies for Novelty Detection

The class-specialized novelty detector requires the use of multiple novelty score generators, each one designed to recognize only the class of its expertise, besides multiple decision thresholds settled using the validation set; thus, it differs from the Non-specialized topology to which only one novelty score generator is designed to recognize multiple classes [1]. There are some advantages of this specialization, one of them is the fewer amount of data used in each model training phase. It is

important to note that, for many models, the training phase may have an extremely high computing complexity and involves a high number of training instances that may result in an extensive processing time. Additionally, in techniques like the *k*NN [10], to which the novelty score generation process for a single instance requires the evaluation of all the training dataset, larger datasets may increase substantially the classification time. Besides, the detector specialization in classes tends to produce novelty score generators that better describe more specific nuances in data.

In this work two class-specialized topologies will be evaluated and compared with the Non-specialized approach, which is the default practice in the literature [5], here denoted as: Hierarchical, Unanimous, and Non-specialized [1].

2.3.1 The Non-specialized topology

The here so-called Non-specialized topology is simply the novelty detector architecture featured in Figure 2.2 applied to recognize multiples classes, i.e., assuming unique training and validation sets composed of multiple classes.

2.3.2 The Hierarchical topology

The Hierarchical topology makes use of a classifier to select a specialized novelty detector, i.e., one detector especially tailored for recognizing the class of its expertise. This means that this classifier has the role of identifying the most likely class that the data under analysis must be provenient. Therefore, the classifier model must be trained to recognize the known classes. As a result, the novelty detector selected for some unknown data will correspond to the one with more features in common with the data under analysis. This architecture can be seen in Figure 2.3.

2.3.3 The Unanimous topology

In the Unanimous topology, novelty detectors of each class work in parallel, producing an output that is either one (novelty) or zero (non-novelty). When all novelty detectors' outputs agree regarding novelty, the data under analysis is assumed to be a novelty. If at least one of the detectors disagrees, the novelty hypothesis is

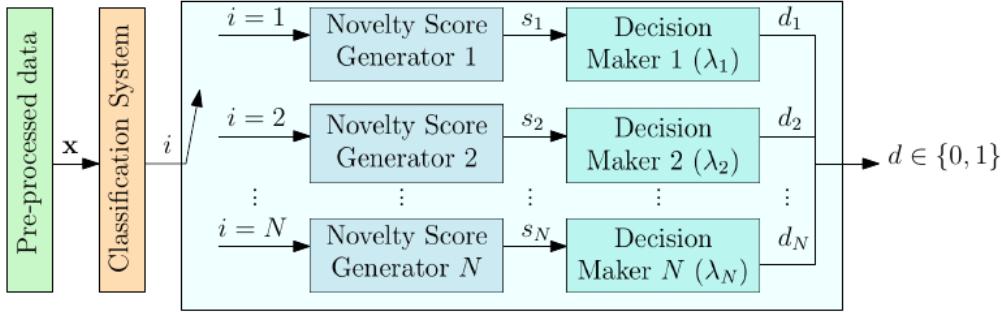


Figure 2.3: Hierarchical novelty detection system architecture (extracted from [1]).

refused. Thus, the fusion mechanism is basically a logical *and* operation involving all the novelty detector outputs. This topology can be seen in Figure 2.4.

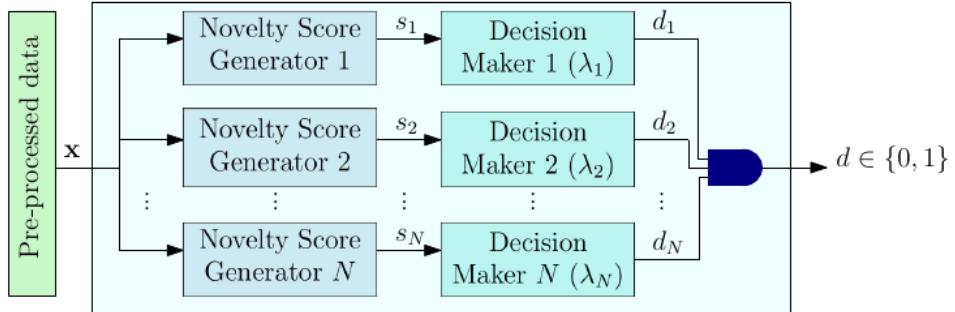


Figure 2.4: Unanimous novelty detection system architecture (extracted from [1]).

2.4 Machine Learning techniques

To generate the novelty scores, Machine Learning (ML) techniques responsible for mapping the input vector into some novelty score space will be explored. For instance, if a specific technique reconstructs the original data, the reconstruction error may represent a possible novelty score, since it is expected that samples from the same class of the detector expertise must obtain a lower reconstruction error than the ones from alternative classes.

To design a novelty detector, at least two datasets are necessary: one for training and another for validating the model(s). Using the training dataset, the algorithm will learn some task from the experience stored in data, where learning refers to the act of acquiring the ability to perform some task [11]. In the case, based on the training dataset, the Machine Learning model will produce some function to generate the novelty score. In turn, the validation dataset will be explored to define one or more decision thresholds or model's hyperparameters required.

The performance of a ML model as a good novelty score generator is directly related to the algorithm chosen and the associated hyperparameters, when applicable. Roughly, a poor choice of hyperparameters can drastically reduce its performance, thus searching for one or more optimal hyperparameters may be of a paramount importance in these cases.

There are many ML techniques capable of generating novelty scores, but this work will concentrate only in a few of them, based on the results of [2], which have led us to focus on instance-based approaches, like k NN, LOF, and NNd [12], as well as some alternatives, such as One-Class SVM [[13], [14], [15]], k -means[[16], [17]], and KPCA [[18], [19], [20]]. Besides, there are some approaches originally evaluated in this problem by this work that include Siamese Neural Networks [[21], [22], [23]] and LSTM-based Autoencoder [1].

2.4.1 k NN

The k -Nearest Neighbors (k NN) is a simple Machine Learning technique based on the geometrical distance from a data under test to its nearest neighbors from the training set [10]. These distances are explored to produce some statistics over this data, typically the median, which is then used as a novelty score[12]. This implies that for an instance from a known class, the median distance must be smaller than that for an instance from a unknown class, based on the premise that instances from a same class tend to be spatially close due to the similarity between their features. A common variant of this algorithm considers the mean instead of the median.

Figure 2.5 illustrates how the k NN algorithm works considering three neighbors ($k = 3$). The circle symbol represents instances from the training set, while the star symbol represents an instance of the same class as the training set but that is under evaluation (test instance). The triangle instance is the furthest away instance, thus it represents a novelty. The pentagon may be classified as normal as it is somewhat close to the class data, depending on the decision threshold settled, as it can also be classified as a novelty.

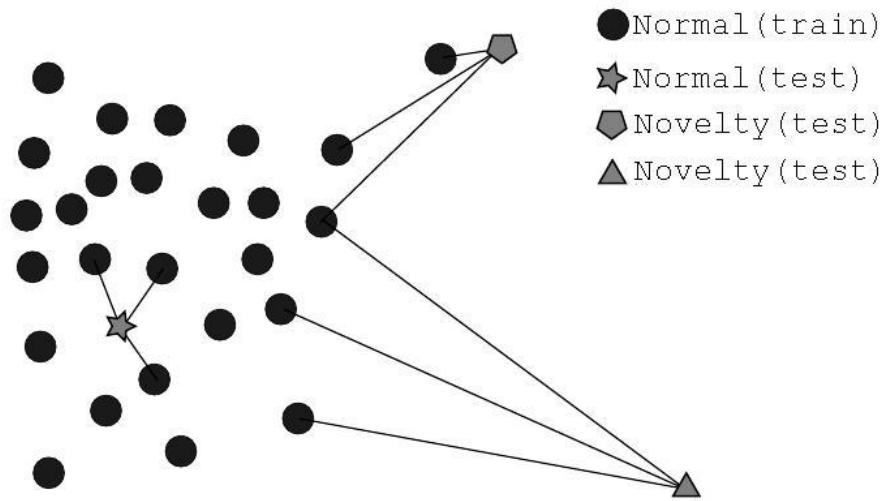


Figure 2.5: Demonstration of k NN (adapted from [2]).

The unique k NN's hyperparameter tuned in the experiments was the number of neighbors, as it was considered the Euclidean distance.

2.4.2 LOF

The Local Outlier Factor (LOF) technique is based on a comparison between the local density of data with the densities from its nearest neighbors, thus it assumes that a novelty(originally, an outlier) is represented by an instance whose density differs from the ones of its neighbors [24].

The concept of local density, in accordance with [3], can be better understood by Figure 2.6, wherein the blue circle represents a novelty data, and the red circles

correspond to normal data. In this plot, the unfilled circles belong to the training set and the filled ones to the test set. Considering $k = 3$, the local densities are represented by the dashed circles. Notably, the radius related to the novelty data is bigger than of the normal data.

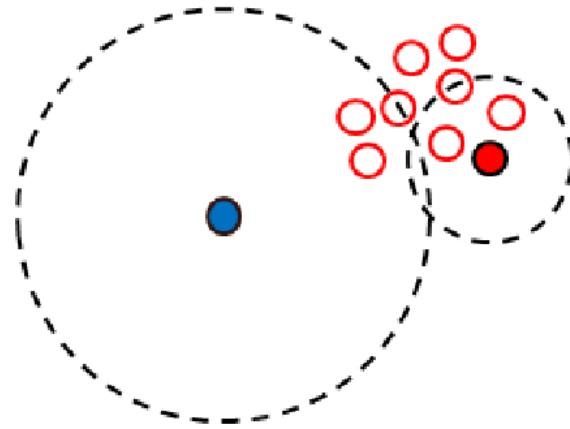


Figure 2.6: Local density of two different data instances (adapted from [3])

The reachability distance of a data under novelty evaluation \mathbf{x} from a data in the training set \mathbf{y} is given by Eq. (2.1), wherein $dist(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between \mathbf{x} and \mathbf{y} , and $dist_k(\mathbf{y})$ the Euclidean distance between \mathbf{y} and its k th neighbor. Thus, the reachability distance is the maximum between these two distances as follows [3]

$$da_k(\mathbf{x}, \mathbf{y}) = \max\{dist_k(\mathbf{y}), dist(\mathbf{x}, \mathbf{y})\}. \quad (2.1)$$

The local reachability density, which is given by the Eq. (2.2) where $\#N_k$ the cardinality of the set with all neighbors with distance equal or lower than $dist_k(\mathbf{x})$, expresses how far an instance must be from \mathbf{x} to reach the nearest data. This means that greater distances would correspond to sparse and less dense regions [3].

$$dal_k(\mathbf{x}) = \frac{\#N_k(\mathbf{x})}{\sum_{\mathbf{y} \in N_k(\mathbf{x})} da_k(\mathbf{x}, \mathbf{y})}. \quad (2.2)$$

The LOF score is defined in terms of $dal_k(\mathbf{x})$ as follows

$$s = LOF_k(\mathbf{x}) = \frac{1}{\#N_k(\mathbf{x})} \sum_{\mathbf{y} \in N_k(\mathbf{x})} \frac{dal_k(\mathbf{y})}{dal_k(\mathbf{x})}. \quad (2.3)$$

A LOF score higher than 1 indicates that the instance represents a novelty, while the opposite means that it must belong to a known data class [12]. The LOF's hyperparameter considered here is the number of neighbors.

2.4.3 NNd

The Nearest Neighbor Density (NNd) technique is a novelty score based on the ratio between two distances: the numerator corresponding to the distance between the data under analysis and its k th nearest neighbor from the training set, denoted as $NN_k^{tr}(\mathbf{x})$, and the denominator given by the distance between this k th nearest neighbor and its corresponding nearest neighbor $NN_1^{tr}(NN_k^{tr}(\mathbf{x}))$ [25], as expressed in Eq. (2.4). Lower score values are related to instances from a known class [25].

The NNd hyperparameter is given by the value of k .

$$s = \frac{dist(NN_k^{tr}(\mathbf{x}))}{dist(NN_1^{tr}(NN_k^{tr}(\mathbf{x})))}. \quad (2.4)$$

2.4.4 One-Class SVM

The One-Class SVM is a classical SVM algorithm variant targeting one-class classification. In this case, the classifier learns how to maximally separate training data from the origin, a place where the novelties are supposed to be, using a hyperplane with a maximum margin [26]. To obtain this hyperplane, Schölkopf [27] proposed the cost function described in Eq. (2.5), where ν is the margin violation penalizing factor, such that $\nu \in (0, 1]$; n is the number of instances in the training set; ξ_i are slack variables to account for the margin violations, and $\phi(\cdot)$ is some non-linear intrinsic high-dimensional mapping function. It is expected that the values of \mathbf{w} and ρ that solve the problem will establish a positive decision function for most of the instances in the training set [26].

$$\min_{\mathbf{w}, \xi_i, \rho} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu n} \sum_{i=1}^n \xi_i - \rho, \quad \text{subject to: } \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle \geq \rho - \xi_i, \forall i \\ \xi_i \geq 0, \forall i \quad (2.5)$$

This algorithm exploits the kernel trick to avoid the need of explicit high-dimensional data mappings into the feature space, thus allowing an effective data classification by using simple linear classifiers that operate in this space. Some kernel functions are RBF, Linear and Polynomial, where Eq. (2.6) exhibits the RBF case considering two feature vector instances \mathbf{x}_1 and \mathbf{x}_2 , wherein σ is a hyperparameter to be settled in the experiments.

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2}{2\sigma}}. \quad (2.6)$$

The major difference from this technique to the others is that the resulting model is strongly tied with the ν parameter, related to the percentage of data in the training set that is assumed as novelty *a priori*.

2.4.5 KPCA

The KPCA (Kernel Principal Components Analysis) is a non-linear extension of the Principal Component Analysis (PCA) that assumes an implicit non-linear transformation applied to the dataset instances [28]. PCA is a statistical method that produces an orthogonal linear transform responsible for mapping correlated observations into uncorrelated observations [29]. The geometric interpretation is that this transform will be defined in terms of directions known as the principal components that are mutually orthogonal. Over these directions, the data is projected, such that the variance of the corresponding projections are maximum, resulting in an optimal linear compression [29]. KPCA extends PCA operation to a high-dimensional feature space expanding these projections using a kernel function [18].

Novelty detection based on KPCA employs the reconstruction error as the novelty score, as expressed in Eq. (2.7), wherein $\|\tilde{\Phi}_*\|^2$ is given by Eq. (2.8), the vector \mathbf{x} is the instance under novelty evaluation, N is the training set size, $\kappa(\mathbf{X}_*, \mathbf{X}_*)$ represents the kernel function, \mathbf{K} is the Gram matrix, \mathbf{V} is related to the SVD

decomposition of the Gram matrix, the vector $\|\mathbf{z}_*\|^2$ is provided by Eq. (2.9), and $\mathbf{k}_* = [k(\mathbf{x}_1, \mathbf{x}_*), k(\mathbf{x}_2, \mathbf{x}_*), \dots, k(\mathbf{x}_n, \mathbf{x}_*)]$ is the empirical kernel mapping [2].

$$\|\mathbf{e}_*\|^2 = \|\tilde{\Phi}_* - \hat{\Phi}_*\|^2 = \|\tilde{\Phi}_*\|^2 - \|\mathbf{z}_*\|^2. \quad (2.7)$$

$$\|\tilde{\Phi}_*\|^2 = k(\mathbf{X}_*, \mathbf{x}_*) - \frac{2}{N} \sum_{i=1}^N k(\mathbf{X}_*, \mathbf{x}_i) + \frac{1}{N^2} \sum_{j=1}^N \sum_{i=1}^N k(\mathbf{x}_i, \mathbf{x}_j). \quad (2.8)$$

$$\|\mathbf{z}_*\|^2 = \mathbf{V}^T \mathbf{k}_*. \quad (2.9)$$

The KPCA hyperparameter is the number of columns of \mathbf{V} , which is related to some percentage of the explained variance retained in $\hat{\Phi}_*$.

2.4.6 *k*-Means

The *k*-Means is a simple and fast iterative clustering method whose the goal is splitting data into a specified number of clusters defined by the value of *k* [30]. It is initialized by randomly selecting the centers of the presumed *k* clusters. After that, each dataset instance is assigned to its nearest center, using the Euclidean distance. Then, all cluster centers are updated to the value of their centroids. This process is repeated until no change is observed in the cluster centers. The cost function associated to the *k*-means algorithm is presented in Eq. (2.10), wherein the vector \mathbf{x} represents the data under analysis, and \mathbf{x}_i corresponds to the centroid of the cluster \mathcal{C}_i [30]

$$E = \sum_{i=1}^k \sum_{\mathbf{x} \in \mathcal{C}_i} \|\mathbf{x} - \mathbf{x}_i\|_2^2. \quad (2.10)$$

The novelty score for the *k*-Means algorithm is the Euclidean distance between the data under analysis and its closest centroid. Thus, this method's hyperparameter is the number of clusters.

2.4.7 Gaussian Mixture Models

The Gaussian Mixture Models (GMM) are probabilistic models that assume a data distribution composed by a linear combination of Gaussians [31], as depicted

in Eq. (2.11), where $\mathbb{N}(\mathbf{x}|\mu_{\mathbf{k}}, \Sigma_{\mathbf{k}})$ represents the k th Gaussian distribution that integrates the mixture, with mean $\mu_{\mathbf{k}}$ and covariance $\Sigma_{\mathbf{k}}$, as described in Eq. (2.12) [3].

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathbb{N}(x|\mu_{\mathbf{k}}, \Sigma_{\mathbf{k}}). \quad (2.11)$$

$$\mathbb{N}(\mathbf{x}|\mu_{\mathbf{k}}) = \frac{1}{(2\pi)^{N/2} |\Sigma_{\mathbf{m}}|^{1/2}} \exp\left(\frac{-1}{2} (\mathbf{x} - \mu_{\mathbf{m}})^T \Sigma_{\mathbf{m}}^{-1} (\mathbf{x} - \mu_{\mathbf{m}})\right). \quad (2.12)$$

The following covariance matrix forms can be assumed [2]:

1. Spherical: It is given by $\Sigma_{\mathbf{k}} = \sigma_k^2 I$, thus it assumes that the base distributions are isotropic.
2. Diagonal: It is given by $\Sigma_{\mathbf{k}} = \text{diag}(\sigma_k)^2$, wherein the base distributions are the same but each dimension may have a particular variance.
3. Full: It is given by arbitrary multivariate Gaussian distributions, one to each member of the mixture, representing the most general case.
4. Tied: It is given by $\Sigma_k = \Sigma$, thus the covariances are arbitrary but shared by all distribution members.

The difference between these premises are illustrated in Figure 2.7 for a hypothetical distribution.

The GMM training is based on the Expectation-Maximization algorithm, typically used for deriving maximum likelihood solutions [31]. The novelty score for GMM is the log-likelihood. The hyperparameters correspond to the type of covariance assumed and the number of Guassians that integrate the mixture.

2.4.8 Autoencoder

The Autoencoder is an artificial neural network that aims to learn some non-linear data mapping into a reduced-dimensional latent space, such that the data can be mapped back to the original data space with the lowest possible reconstruction error [32].

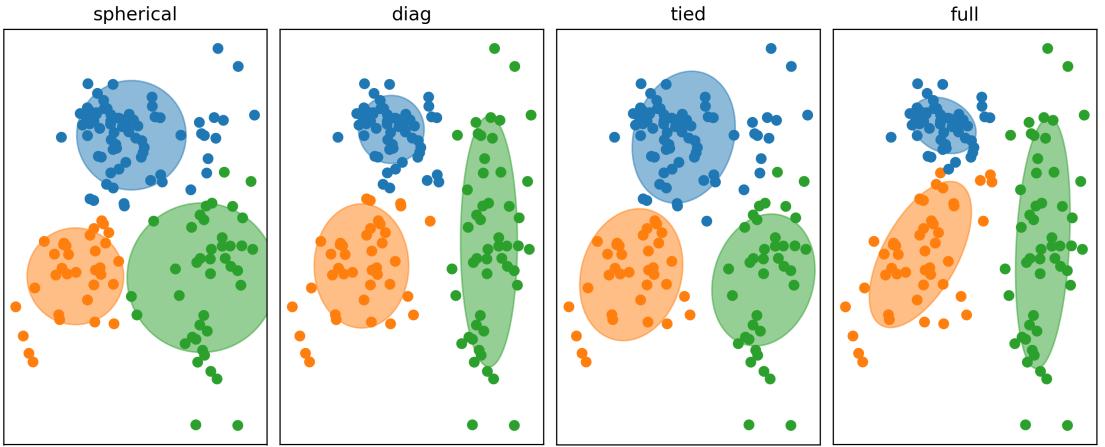


Figure 2.7: Illustration of the different assumptions over the covariance matrix format for an hypothetical distribution example (extracted from [4]).

Basically, two components integrates an Autoencoder: the encoder part, whose task is mapping the data to this latent space; and the decoder, responsible for the inverse mapping, i.e., for reconstructing data into the original data space. Encoder and decoder parts are implemented by one or more network layers; thus, the flow of information is Hierarchical. For convenience, a standard procedure is adopting symmetric encoding and decoding layers [1]. Figure 2.8 depicts an Autoencoder network architecture with symmetric encoding and decoding layers.

Neural Networks' training requires the optimization of parameters, like weight and bias, using some optimization algorithm, such as ADAM [33] or SGD [34], targeting to minimize(or maximize) some loss function. Autoencoders often exploit the squared reconstruction error computed between the reconstructed and the original data as the loss function for training. Similarly, the reconstruction error is used as novelty score. The premise is that this error will be lower for instances belonging to the class(es)to which the Autoencoder was trained upon, since it learns how to represent this data in a compact form.

An interesting variant of the standard Autoencoder is the LSTM-based Autoencoder, which uses LSTM cells instead of *perceptrons*, a well-known approach in models that process data sequences [1].

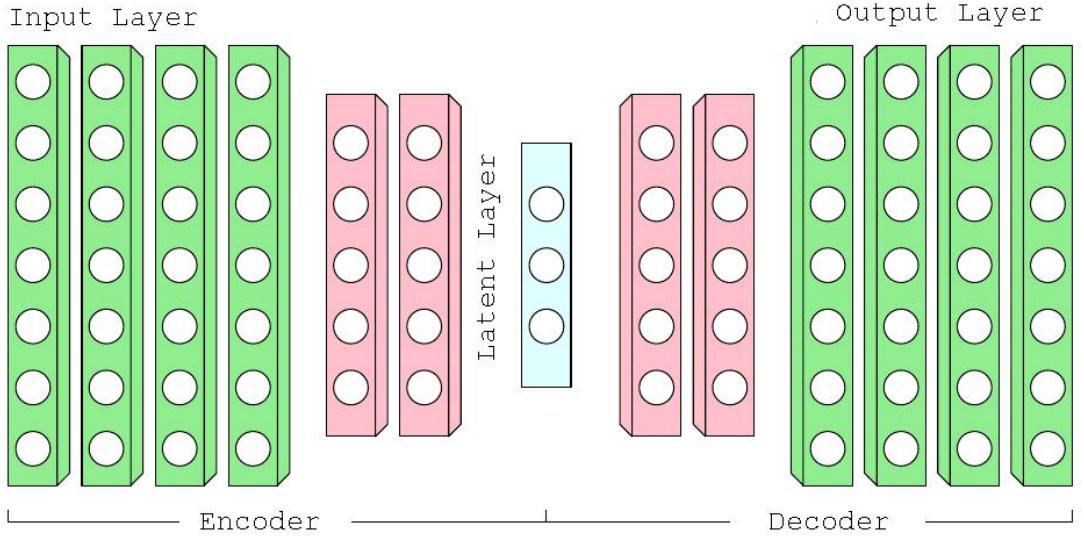


Figure 2.8: Symmetric auto-encoder architecture (adapted from [5]).

LSTM stands for Long Short-Term Memory [35]. It is a recurrent neural network composed by cells. Each cell is responsible for storing some state and has three gates controlling the information flow. The cell acts as a memory, allowing old input values to be remembered after arbitrary time intervals [1]. LSTM units include a *input gate*, a *forget gate*, and a *output gate*. The *input gate* defines how the current input and hidden state values would impact in the current state update. The *forget gate* decides how much of the information stored in the cell should remain after an update. Finally, the *output gate* responds for how much of the hidden state must accompany the current cell state value. The Figure 2.9 depicts a LSTM cell.

The LSTM feedforward equation are given by:

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_{xi}\mathbf{x}^{(t)} + \mathbf{W}_{hi}\mathbf{h}^{(t-1)} + \mathbf{W}_{ci}\mathbf{c}^{(t-1)} + \mathbf{b}_i), \quad (2.13)$$

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_{xf}\mathbf{x}^{(t)} + \mathbf{W}_{hf}\mathbf{h}^{(t-1)} + \mathbf{W}_{cf}\mathbf{c}^{(t-1)} + \mathbf{b}_f), \quad (2.14)$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_{xo}\mathbf{x}^{(t)} + \mathbf{W}_{ho}\mathbf{h}^{(t-1)} + \mathbf{W}_{co}\mathbf{c}^{(t-1)} + \mathbf{b}_o), \quad (2.15)$$

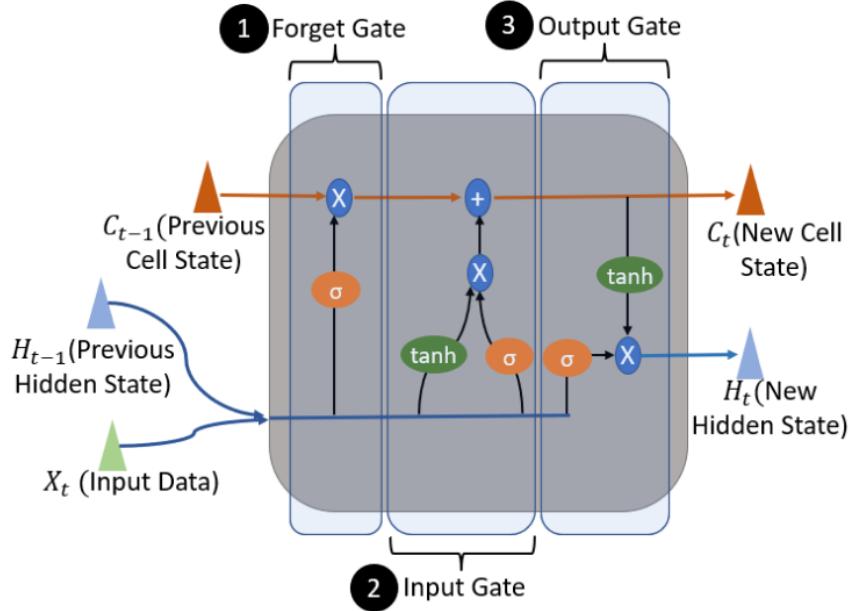


Figure 2.9: Representation of the LSTM cell(extracted from [6]).

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \psi(\mathbf{c}^{(t)}), \quad (2.16)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \psi(\mathbf{W}_{xc} \mathbf{x}^{(t)} + \mathbf{W}_{hc} \mathbf{h}^{(t-1)} + \mathbf{b}_c). \quad (2.17)$$

where $\mathbf{x}^{(t)}$ is the input vector; $\mathbf{f}^{(t)}$ is the forget gate's activation vector; $\mathbf{i}^{(t)}$ is the input gate's activation vector; $\mathbf{o}^{(t)}$ is the output gate's activation vector; $\mathbf{h}^{(t)}$ is the hidden state vector, also known as the output vector; $\mathbf{c}^{(t)}$ is the cell state vector; \mathbf{W} are weight matrices related to the gates and internal cell processing; \mathbf{b} are bias vectors shared across multiple-time steps within a same layer; σ is the sigmoid activation function used in the three gates; ψ is an activation function initially defined as the hyperbolic tangent but that can be arbitrary chosen [1]. The dimensionality of the all these vectors are referred as the number of LSTM units.

The hyperparameters of a LSTM Autoencoder are the number of LSTM units, the number of LSTM layers, and the activation function of each layer. It is a hard task to optimally determine the best number of layers to each model. This work adopted

a greedy approach: new layers are inserted in the model until its performance starts to level off. Besides, for each layer added, optimal hyperparameters are identified and kept frozen in the subsequent experiments. To avoid overfitting and increase the model generalization, the concept of denoising Autoencoder [36] was explored. This strategy consists of adding zero-mean random Gaussian noise to the network inputs. As a result, the resulting model becomes more robust, since the Autoencoder must reconstruct it based on a corrupted version the original data instance. Another strategy explored was the *dropout* [37] which simply consists of randomly dropping some neurons (LSTM units) during training. The hyperparameters in this case are the dropout rate and the standard deviation of the noise added to the network inputs.

2.4.9 Siamese Neural Networks

Siamese Neural Networks (SNNs) is an architecture composed by two or more neural networks with shared parameters dedicated to solving some task [38]. The main SNN goal is similarity learning, i.e., this network learns low-dimensional input data mappings to which the similarity or dissimilarity between input data instances are somewhat enhanced. Therefore, this technique requires contrasting data, which restricts its use only to class-specialized detectors, since it demands the existence of class and non-class instances beforehand, here referred as positive and negative datasets, respectively.

Figure 2.10 illustrates an arbitrary SNN architecture composed by two networks (with inputs \mathbf{x}_1 e \mathbf{x}_2) having N_1 neurons in the input layer and N_2 neurons in the second layer. Note that the weights from both networks are the same. The SNN can have two or more input vectors provenient from the same or from different classes. For each pair of inputs, SNN encodes them through its hidden layers, producing two outputs that correspond to specific embeddings for these inputs. Then, the difference between these embeddings is processed by a distance layer that essentially computes the distance between them. Depending on the type of data that is fed to this architecture, the optimization of this loss function will affect this distance. Note that the optimization of this loss function targets to minimize the

distance from input vector mappings that are from the same class and maximize those from different classes. As a result, the model produces output vectors with higher distances when submitted to input vectors from different classes, while smaller distances for input vectors that belong to the particular class used in its train step.

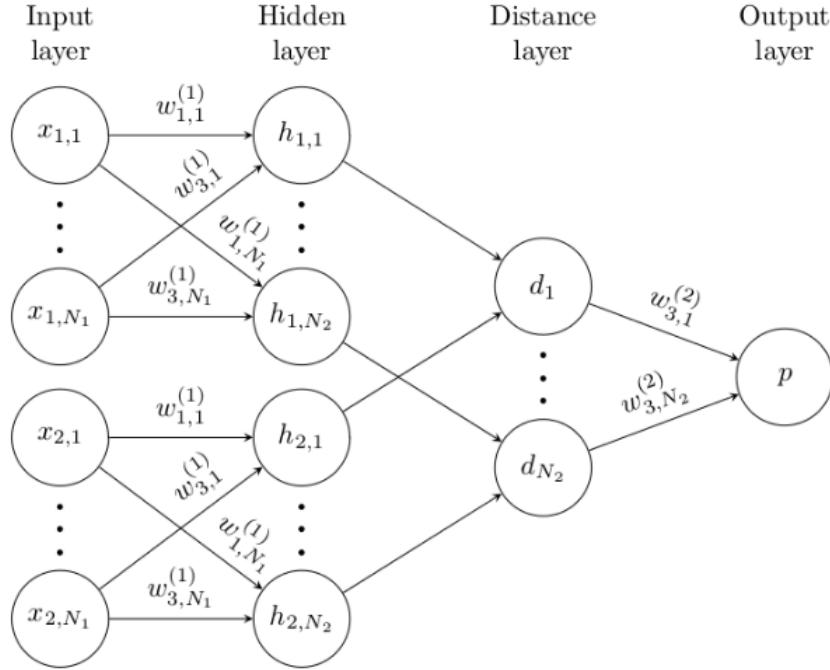


Figure 2.10: An illustration of the SNN architecture (extracted from [7]).

To train a SNN it is necessary dispose of two training sets: one for the positive samples and another for the negative samples. In this work, the positive set will correspond to the class of detector expertise, while the negative set will be composed by the others known classes of the system.

The triplet loss [39] was the loss function explored to train the SNN's novelty detectors. This function requires settling three inputs simultaneously: Positive Sample, Negative Sample, and Anchor. Both the positive sample and the anchor must be from the same class of detector expertise, while the negative sample should be represented by a data instance from another class. Then, the SNN will produce mappings for each one of these inputs. As previously stated, the goal is maxi-

zing the distance between the anchor and the negative instance while minimizing the distance between the anchor and positive sample. The Triplet Loss function is depicted in Eq. (2.18), where f denotes the output of the network, the subscripts a , p , n denote the anchor, positive and negative inputs respectively; i denotes the i th instance; N is the total number of training instances in the batch, and w denotes a multiplicative weight applied to the negative distance that scales the negative-anchor distance contribution, since it grows rapidly, making difficult for the model learn how to reduce the positive-anchor distance.

$$L = \sum_i^N \max(\|f(\mathbf{x}_{i=1}^a) - f(\mathbf{x}_i^p)\|_2^2 - w\|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^n)\|_2^2, 0). \quad (2.18)$$

A typical literature approach consists of creating triplets by combining every positive and negative sample, a very time-consuming process. However, some work [39] showed that not all examples equally contribute to learning. Here, a simple strategy is adopted to define the positive and negative datasets using only the nearest neighbor of the anchor vector in both cases, which is called the easy-positive-hard-negative mining strategy [40].

As the detectors follow a class-oriented focus, it is possible to explore more information about the non-class space, for instance, considering an approach that selects one nearest neighbor for each one of the existing non-classes in the system. Thus, two strategies are possible to compose the triplets: a) One Against All, where a neighbor is selected among all data of the non-class together, which is the most common approach in the literature, and b) One Against One, where a neighbor is selected for each non-class in particular, which is a new approach proposed by this work. These strategies are illustrated in Figures 2.11 and 2.12, where $NN(x)$ in Y indicates the nearest neighbor from the element x in the set Y , and $Y - \{x\}$ represents the set Y excluding the element x .

The novelty score in this case is defined by the distance between the SNN mapping for the data under analysis and the one correspondent to its nearest neighbor from the training set.

Detector Class: A
 System Class: A, B & C
 Positive Dataset(P): $\{a_1, a_2 \dots a_N \in A\}$
 Negative Dataset(N): $\{b_1, b_2 \dots b_N \in B\}$
 $c_1, c_2 \dots c_N \in C\}$

Triplet Formation

Positive	Anchor	Negative
NN(a1) in P-{a1}	a1	NN(a1) in N
NN(a2) in P-{a2}	a2	NN(a2) in N
⋮	⋮	⋮
NN(aN) in P-{aN}	aN	NN(aN) in N

Figure 2.11: An illustration of the One Against All strategy for the triplet formation.

Detector Class: A
 System Class: A, B & C
 Positive Dataset(P): $\{a_1, a_2 \dots a_N \in A\}$
 Negative Dataset B(NB): $\{b_1, b_2 \dots b_N \in B\}$
 Negative Dataset C(NC): $\{c_1, c_2 \dots c_N \in C\}$

Triplet Formation

Positive	Anchor	Negative
NN(a1) in P-{a1}	a1	NN(a1) in NB
NN(a1) in P-{a1}	a1	NN(a1) in NC
NN(a2) in P-{a2}	a2	NN(a2) in NB
NN(a2) in P-{a2}	a2	NN(a2) in NC
⋮	⋮	⋮
NN(aN) in P-{aN}	aN	NN(aN) in NB
NN(aN) in P-{aN}	aN	NN(aN) in NC

Figure 2.12: An illustration of the One Against One strategy for the triplet formation.

For the sake of simplicity, the SNN models in this work are based on the Multilayer Perceptron (MLP) architecture, whose hyperparameters are the number of layers,

the quantity of neurons per layer, and the activation function. Like the Autoencoder, the number of layers was chosen incrementally.

As SNN can learn mappings that better isolate classes from non-classes. Therefore, this feature space can be also used in conjunction with other novelty detection techniques, particularly the k NN.

2.4.10 ODIN

ODIN is a Neural Network technique destined to detecting *out-of-distribution* data based on standard NN classifiers [41]. This out-of-distribution detector involves two factors: temperature scaling and input preprocessing. To briefly describe ODIN, assume a neural network trained to classify N classes, with *softmax* outputs. The class label predicted by the network is $\hat{y}(\mathbf{x})$ and it is given by Eq. (2.19), wherein S_i is computed by Eq. (2.20). The factor T represents the temperature scaling parameter, settled to 1 during the training. The *softmax* score for the winner class is given by Eq. (2.21). The purpose of the temperature scaling factor is turning the output neurons more selective, allowing a more efficient out-of-distribution detection [41].

$$\hat{y}(\mathbf{x}) = \operatorname{argmax}_i S_i(\mathbf{x}; T). \quad (2.19)$$

$$S_i(\mathbf{x}; T) = \frac{\exp(f_i(\mathbf{x})/T)}{\sum_{j=1}^N \exp(f_j(\mathbf{x})/T)}. \quad (2.20)$$

$$S_{\hat{y}}(\mathbf{x}; T) = \max_i S_i(\mathbf{x}; T). \quad (2.21)$$

The strategy behind ODIN is perturbing input data to maximally deviate *softmax* outputs. Thus, small perturbations considering the gradient of network outputs relatively the input data with a factor ϵ are produced by Eq. (2.22), where ϵ is the perturbation magnitude parameter.

$$\tilde{\mathbf{x}} = \mathbf{x} - \epsilon \operatorname{sign}(-\nabla_{\mathbf{x}} \log S_{\hat{y}}(\mathbf{x}; T)). \quad (2.22)$$

The novelty score generator combines input preprocessing and the temperature scaling. For each input \mathbf{x} , it computes the preprocessed version of it, represented by the vector $\tilde{\mathbf{x}}$, which will define the input for the NN when computing the calibrated *softmax* score, described by Eq. (2.21) [41]. The novelty score is defined by this *softmax* score.

Note that the perturbation magnitude and the temperature scaling factor are ODIN hyperparameters. An additional design factor refers to the MLP architecture adopted. Since this technique is based on a monolithic N class classifier, it does not allow the development of class-specialized novelty detectors.

Chapter 3

Materials and Methods

One of the major work objectives is showing that class specialization may be beneficial for novelty detection. Thus, this experimental study requires a dataset for models' learning and evaluation. Assessing models' performance also requires some figures of merit. Next sections will focus on discussing these issues.

3.1 Dataset

The dataset used in this work was provided by the Brazilian Navy Research Institute (IPqM). It is constituted by noise signals irradiated by 28 vessels of 8 classes during 263 experimental runs in an acoustic lane situated in Arraial do Cabo, Rio de Janeiro. The original vessels classes were hidden using alphabetical letters due to security reasons [1]. In each run a vessel maintained its operational conditions, having its noise acquired by one hydrophone situated on the seabed [2].

The noise signals were sampled at a frequency of 22050 Hz with a resolution of 16 bit per sample, having underwent by the preprocessing chain depicted in Figure 3.1 [8]. First, the signal was converted from the analog to the digital domain and subsequently normalized. After, a Hanning window[42] was used to reduce the artifacts introduced by the signal windowing. Then, the amplitudes of the Discrete Fourier Transform of the signal with a size of 4096 were computed. The resulting signal was then submitted to the Two-Pass Split Windows(TPSW)[43] normalization to remove the background noise. Lastly, a subset composed by the

first 557 frequency spectral points were taken, covering the frequency range of 0 to 3kHz, which is associated to the vessel's machinery noise signature. The resulting signal can be exhibited by a plot named spectrogram, depicted in Figure 3.2, where some vertical patterns (in a hot color) can be observed, which are related to the vessel's machinery harmonics.

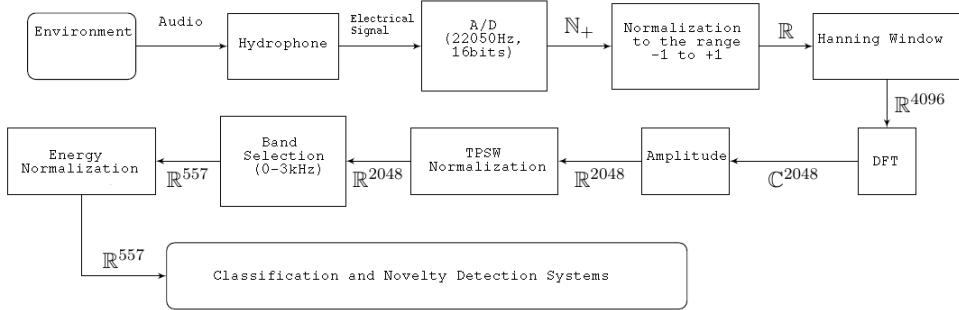


Figure 3.1: Diagram of the digital signal processing chain (Adapted from [8]).

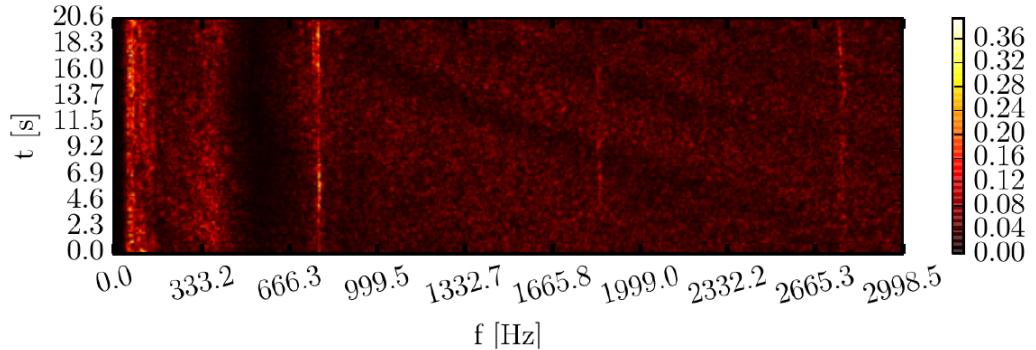


Figure 3.2: Spectrogram for an arbitrary database run (Extracted from [8]).

The procedure described above was applied to each vessel run, resulting in a total of 29277 spectral windows with 557 dimensions each. Table 3.1 shows the number of spectral windows available to each class. There is an evident class imbalance, but the adopted performance assessment metrics as well as the use of specialized class detectors mitigate possible negative effect of this on the detector's performance.

Table 3.1: Number of spectral windows for each class.

Class	Spectral Windows
A	2432
B	3432
C	4797
D	3072
E	7075
F	2934
G	2143
H	3392

3.2 Resampling

The novelty detector design requires three datasets: training, validation, and test; the first two for deriving the models, while the last for evaluating their performance.

This work considered the *10-fold resampling technique* [44], where the dataset was divided into 10 approximately equal folds and for each of these folds, one was reserved for testing, while the others were joined and then split in training (90%) and validation (10%), resulting in a total of 10 trios of different training (81% of total), validation (9% of total) and test (10% of total) sets.

3.3 Emulating the Unknown Classes

A rigorous evaluation of novelty detectors requires the knowledge of the unknown classes, which is counter-intuitive and impossible. Thus, to allow a quantitative evaluation of the detectors, the Dataset was divided into two groups: one composed by the classes presumably known, while the other containing those assumed as unknown. Additionally, our experiments explored three different scenarios regarding the number of known classes, considering 3, 5, and 7 classes.

To define which classes would integrate each set, an experiment using the Nearest Neighbour model (a particular use case of the *kNN* algorithm to which the value

of k is set equal to 1) with the Non-specialized topology was conducted to identify the worst combination of the known and unknown classes (to be selected from the available dataset), assuming the AUC under the ROC curve as the figure of merit (described in the following). The choice for the Nearest Neighbor model was motivated by its simplicity, good performance, and absence of any additional hyperparameter to be tuned. The classes identified in this analysis for each evaluation scenario are summarized in Table 3.2.

Table 3.2: Set of known and unknown classes identified to each evaluation scenario (see text).

Scenario	Known	Unknown
Three-known classes	{A, C, G}	{B, D, E, F, H}
Five-known classes	{A, B, C, F, G}	{D, E, H}
Seven-known classes	{A, B, C, H, F, G, H}	{D}

3.4 Figures of merit

Basically, two figures of merit were considered here: the accuracy and the area under the ROC curve (AUC). The first was used for tuning the hyperparameters of the classifier adopted in the hierarchical novelty detector topology as well as for assessing its performance. The second was used for tuning the hyperparameters of the novelty detectors.

3.4.1 Accuracy

The accuracy is defined as the ratio between the overall number of correctly predicted instances by the total number of instances evaluated. It is related to the diagonal of the confusion matrix, while off-diagonal entries are related to classification errors.

3.4.2 AUC

Before getting into the process for generating the ROC curve considered in this work, first, two metrics need to be described: the known-class detection rate and

the unknown-class detection rate. Let us assume a set of known classes represented by $K = \{K_1, K_2, \dots, K_N\}$ and unknown classes by $U = \{U_1, U_2, \dots, U_R\}$.

The detection rate of the i th known class is given by

$$TC_i = \frac{1}{\#K_i} \sum_{\mathbf{x} \in K_i} \mathbb{I}(d(\mathbf{x}) = 0), \quad (3.1)$$

where \mathbf{x} is the data instance under evaluation; $d(\mathbf{x})$ is the novelty detector system output for the given instance; $\#A$ represents the cardinality of some set A i.e., the number of data instances integrating it; \mathbb{I} is an indicative function that outputs 1 whenever the condition on its argument is true, otherwise returns 0. Thus, a global performance indicator related to the known-class recognition is given by the average detection rate described as [12]

$$TC = \frac{1}{N} \sum_{i=1}^N TC_i. \quad (3.2)$$

The same goes for the detection of unknown classes. The detection rate of the i th unknown class is given by

$$TD_i = \frac{1}{\#U_i} \sum_{\mathbf{x} \in U_i} \mathbb{I}(d(\mathbf{x}) = 1). \quad (3.3)$$

The global performance indicator related to the unknown class is the average unknown detection rate defined as

$$TD = \frac{1}{R} \sum_{i=1}^R TD_i. \quad (3.4)$$

Since the indicators TC and TD are dependent from the decision threshold settled, the ROC curve can be used to express the values of TD and TC for a variety of decision thresholds. Therefore, a good summary of the detector performance is the area under the ROC curve, whose interpretation is simple: the closer the value of AUC is to the unity, closer will be the ROC curve to the ideal square defined between the origin and the point $TD=TC=100$, thus better will be the model.

3.5 Tuning the novelty detector model hyperparameters

To set the hyperparameters of each novelty detector, the design options with the highest average AUC values inferred over the validation set were considered. For the Hierarchical and Unanimous topologies, each class-specialized novelty detector had its hyperparameters tuned based on a *one-against-all strategy*, where only the known-classes were considered. This means that for the three-known-classes evaluation scenario, for instance, the AUC curve considered for tuning the novelty detector of the class A assumes the two other known classes (C and G) as if they are unknown. Thus, two AUC ($A \times G$ and $A \times C$) are averaged to define the final AUC considered for setting its hyperparameters. The same goes for the classes C and G.

The hyperparameter tuning strategy for the LSTM autoencoder was incremental, i.e., layers were incrementally added to the network until the model performance stopped improving. The search for the best set of hyperparameters was made in four stages, wherein the first stage considered the search for the number of LSTM units and the activation function to be adopted in the first layer. The second stage included the optimal noise standard deviation. The third stage aimed to define the search for the optimal dropout rate. The fourth stage evaluated the addition of a new layer, so involved again properly defining the related number of LSTM units and the activation functions. Previous steps were repeated until the model's performance does not improve.

The same approach was followed with the Siamese Neural Networks but restricted only to the three first stages. The first involved the search for the optimal combination of the number of neurons, activation function, and multiplicative weight that regulates the negative part of the triplet loss. The second stage aimed to establish an appropriate dropout rate. The third stage considered the addition of a new layer, thus involved the same steps from the stage one. The previous steps were repeated until the model's performance stopped to improve.

Appendix A summarizes the range of hyperparameters evaluated for each Machine Learning technique and the best setups identified for each class and evaluation scenario considered in this work.

3.6 Decision threshold setting process

As discussed previously in Chapter 3, the figure of merit used to assess the detectors' performance was the AUC, which is based on the ROC curve. In practical terms, the ROC curves generated in this work considered 21 decision-threshold values settled to span TC and TD values in the range of 0 to 100%.

3.7 Statistical Analysis

Results underwent a statistical analysis using the *Friedman* test with a 5% level of significance [44] to verify if the AUC indexes obtained for the different algorithms were overall statistically different. The HSD test [44] was also applied to identify between each pair of methods this difference could be confirmed, i.e., the pairs to which the p -value obtained with the HSD test were lower than 0.05.

3.8 Experimental Computational Environment

All experiments were performed using the Python programming language. The novelty score generators and classifiers were created using the following libraries: Tensorflow[45] for Autoencoder LSTM and Siamese Neural Networks; pytorch [46], for ODIN using the code provided by the authors, and scikit-learn [47] for other machine learning techniques. To manipulate the datasets, store and compare results, the following libraries were used: Pandas [48], Numpy [49], Matplotlib [50], Scipy [51] and scikit posthoc [52].

The computers' hardware used to run the experiments varied. In general, for machine learning techniques based on Neural Networks, computers with an I7 processor of the fourth generation or with a second generation Ryzen 7, both equiped with NVidia video cards (GTX 1080, GTX 1080 Ti, Titan X, RTX 2070, RTX 2080

Ti) and RAM memory ranged from 32 to 64GB DDR4 were used. For the other machine learning techniques, servers with XEON 5120 with 128 GB of RAM.

Chapter 4

Results

This chapter will summarize the experimental results related to the different novelty detection approaches considered in this work.

4.1 General chapter organization

The results were grouped according to the general approach assumed by the technique considered for the generation of the novelty score [9]: Distance based (k NN, NNd, LOF and k -means), Mixture models (Gaussian Mixtures), Neural Networks (Siamese Neural Networks, Autoencoder and ODIN), and Reconstruction and Domain (KPCA and One Class SVM). A specific section will be dedicated to each one of these groups and one additional section will provide an overall comparison between the best methods identified in each group.

The boxplot diagram, which is a graphical summary for the AUC values inferred for all the ten testing folds of each technique and topology, will be used to compare the models. This plot express the concepts of locality, spread, and skewness of experimental outcomes graphically through their quartiles, with the upper part of the box referring to the third quartile and the lower part to the first quartile as the median signalized in orange. In the following boxplot diagrams, the label H after some technique name will refer to the Hierarchical topology; U , the unanimous; while N , Non-specialized.

The statistical comparison of the results was made through hypothesis tests and will be presented using a table in each section. Each of these tables contains four columns, with the first three representing the medians of the ten test folds per topology and the last column corresponding to the lowest p -value related to the statistical comparisons between the highest median result and the medians from the remaining detector topologies. Each table row responds for one particular novelty detection score generator, while the last row depicts the lowest p -values obtained by comparing the highest performance technique from a given general approach from the competing methods. Furthermore, the last column of the last line compares the technique-topology pair with the highest median with each other technique-topology pair considering only the topology with the highest median for this technique to be compared.

Figure 4.1 demonstrate how the p -value shown in the following chapter tables were obtained. Two tables are considered: (a), containing the median of some score generator for three topologies with the highest result marked in red; (b) shows the p -value of the comparison between pairs of topologies. In blue, the row of interest, as it corresponds to the topology with the highest median, wherein the row with smallest p -value is in green, which represents the p -value of interest. The way to find the p -value when comparing different score generators from a same topology as well as scores from different generator-topology pairs is similar. When there are only two topologies or two techniques being compared, the only p -value that exists is shown and in this case the equal sign is applied instead of the greater sign.

Additionally, markers and symbols are used in the table to indicate the superiority or statistical equivalence between techniques and topologies. The topology with the highest median, when compared with alternatives from the same technique (i.e., between alternatives from the same row) marked in bold. The " \rightarrow " symbol shows which topologies are statistically equivalent to the topology marked in bold for the same technique (table row). The outcome of the technique with the highest median, among those from the same topology (table column) is marked in blue. The " \downarrow " symbol indicates the results that are statistically equivalent to those marked in blue when considering a same topology (table column). Marked in red is the pair of

	Hierarchical	Unanimous	Non-specialized
Median	0.9370	0.9346	0.9291

a)

	Hierarchical	Unanimous	Non-specialized
Hierarchical	1.000000	0.503942	0.001009
Unanimous	0.503942	1.000000	0.037036
Non-specialized	0.001009	0.037036	1.000000

b)

Figure 4.1: Demonstration of how the smallest p -value is obtained when comparing topologies of the same technique(see text).

technique and topology with the highest median. Note that the red mark overrides the blue mark and the bold one for obvious reasons. The combinations statistically equivalent to the ones marked in red are signalized by the “*” symbol.

4.2 Distance-based techniques

In the Figures 4.2, 4.3 and 4.4 are depicted the boxplot graphs for the Distance-based novelty detection methods, which include the k NN-median, k NN-mean (denoted as kNNb), LOF, NNd, and k -means.

For all evaluation scenarios (3, 5, and 7 known classes), Friedman tests pointed out that methods performed differently ($\chi^2(14) > 125.81$; $p < 0.001$), thus the p -values were lower than 0.01.

Table 4.1 shows the main results, which can be summarized as follows:

1. Three-known-class scenario: the Non-specialized topology obtained the highest median values for most of the cases, except for k NN-median and k NN-mean.

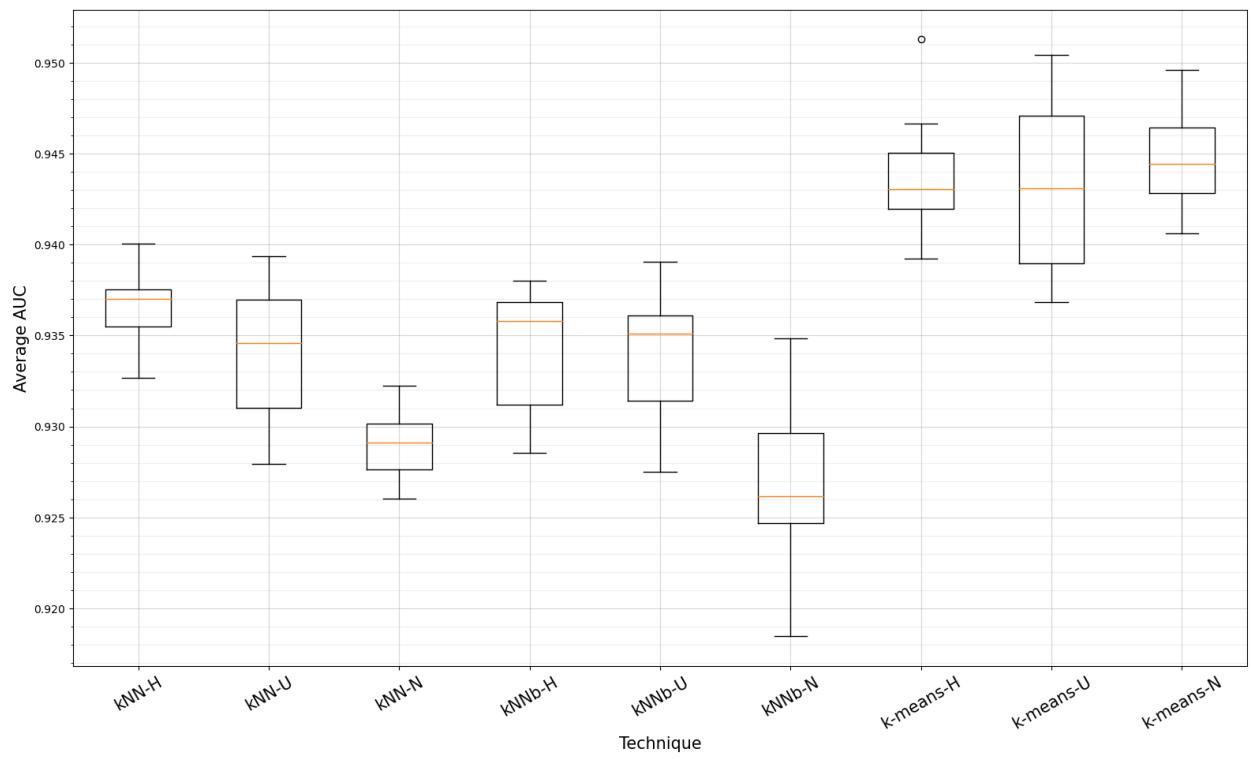
The kNN models achieved the highest medians with the Hierarchical topology, despite the related results being statistically equivalent to those derived with the Unanimous topology. The k -means technique figures out as the one with the highest medians for all topologies, and the Non-specialized topology was the leading one; however, it is statistically equivalent to the Hierarchical topology with the kNN median.

2. Five-known-classes scenario: the Non-specialized topology was the best solution for the LOF and NNd-based novelty detectors. For the remaining cases, the Hierarchical was the best topology. Hierarchical and Unanimous topologies obtained the highest medians with the k -means technique, however these results are statistically equivalent to those achieved by the LOF Non-specialized.
3. Seven-known-classes scenario: the Non-specialized topology is only the best one when considering the LOF score, while for the remaining cases the Hierarchical one. For all the other techniques, the Hierarchical topology represented the best alternative. The technique with the highest performance was the k NN-mean, which was statistically equivalent to the k NN-median and k -means.
4. This analysis can be summarized as follows: the Hierarchical topology exploiting the k -means novelty score represent the best topology for most of the scenarios evaluated.

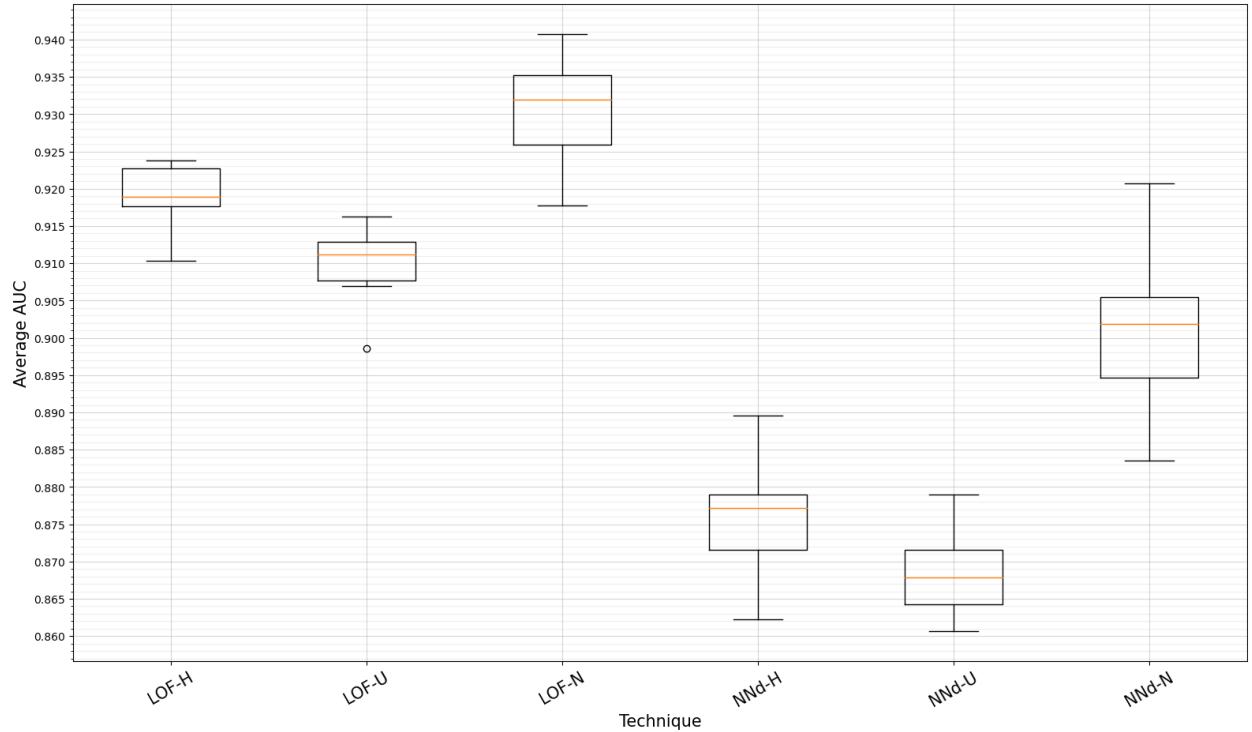
4.3 Mixture-models-based techniques

Figures 4.5 and 4.6 depicts the boxplot graphs for the Mixture-models-based novelty detection methods, which comprises Gaussian-mixture models with different covariance types. In this figure, GM-spherical refers to the spherical covariance assumption; GM-diag, the diagonal covariance; GM-full, the full covariance, and GM represents a mix of covariances types (please refer to Appendix A for more details).

In summary, for all scenarios, the Friedman tests pointed out that the methods performed differently ($\chi^2(11) > 91.06$; $p < 0.001$).

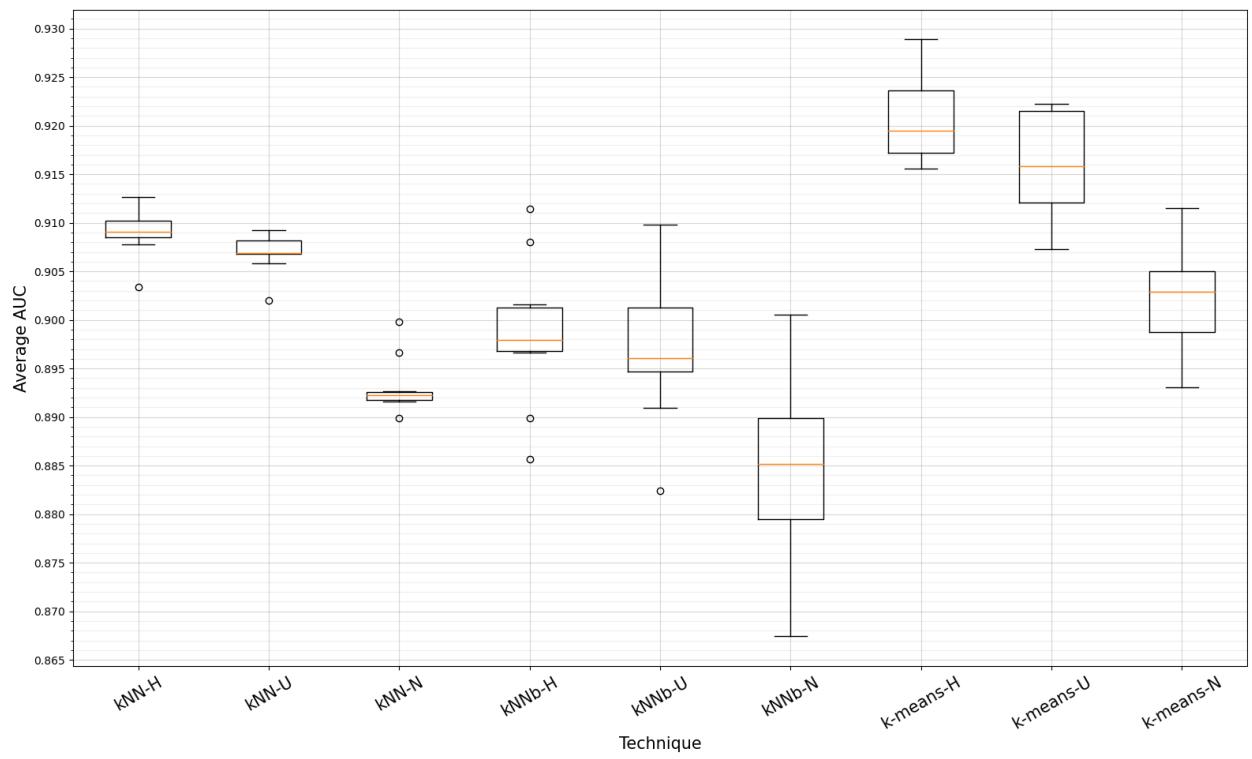


(a) $k\text{NN}$, $k\text{NNb}$ and $k\text{-means}$.

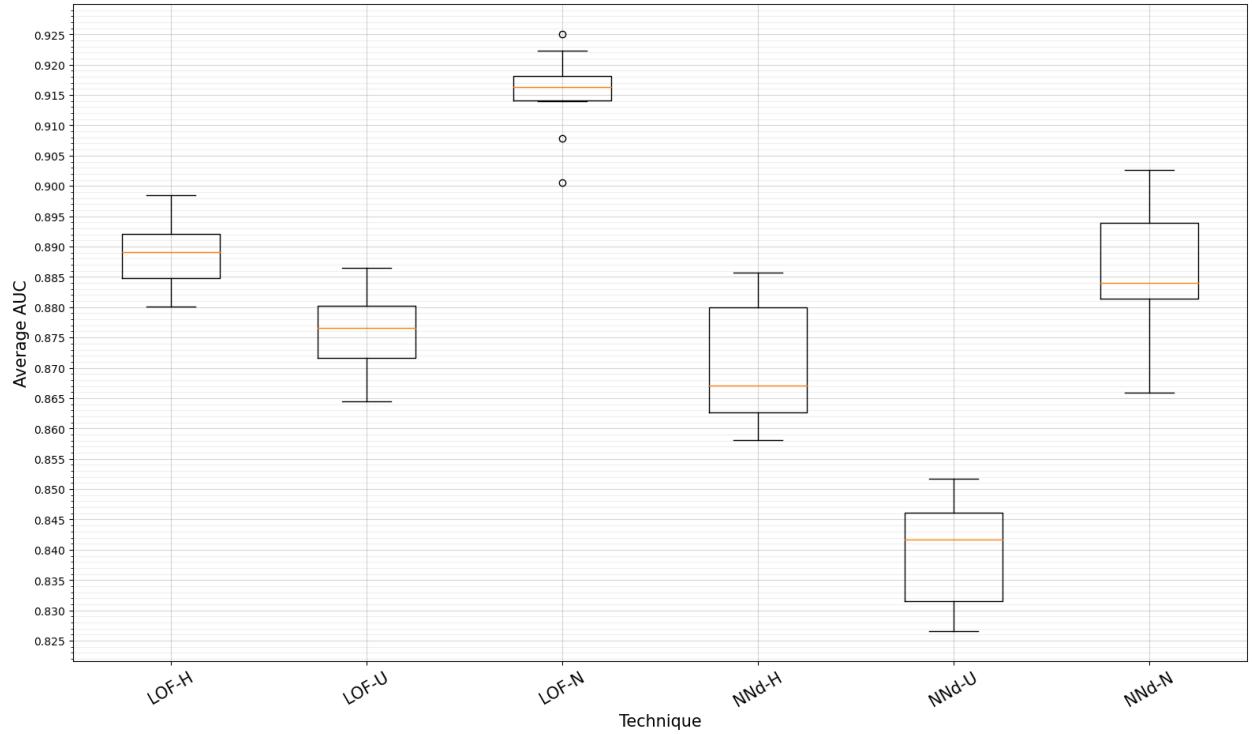


(b) LOF and NNd.

Figure 4.2: Boxplot diagram for the Three-known-class scenario considering the distance-based novelty detection scores.

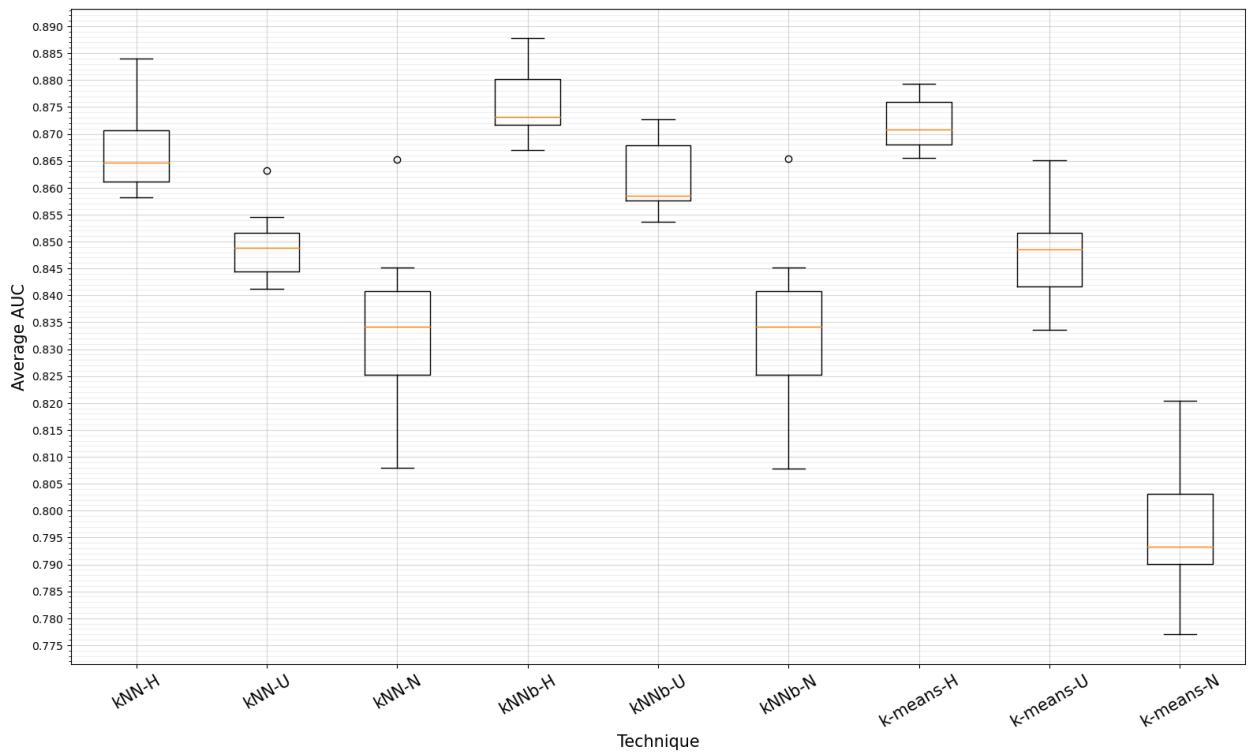


(a) $k\text{NN}$, $k\text{NNb}$ and $k\text{-means}$.

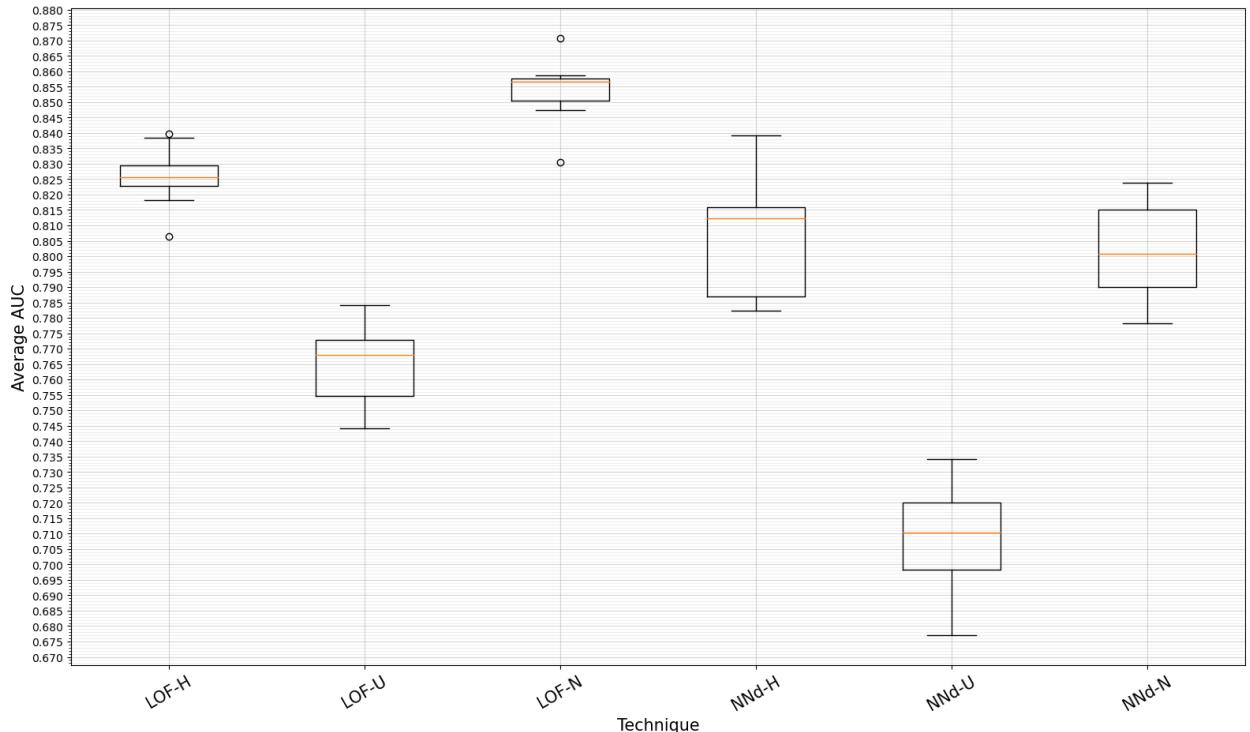


(b) LOF and NNd.

Figure 4.3: Boxplot diagram for the Five-known-classes scenario considering the distance-based novelty detection scores.



(a) $k\text{NN}$, $k\text{NNb}$ and $k\text{-means}$.

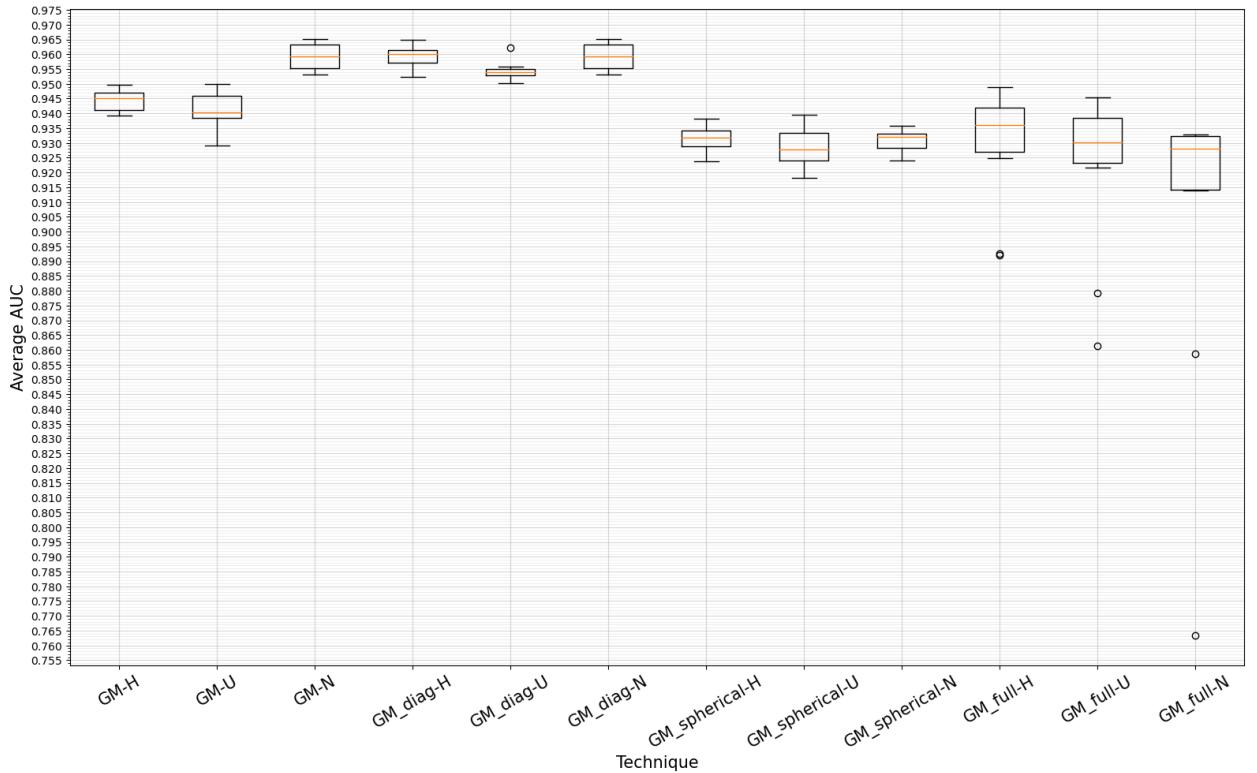


(b) LOF and NNd.

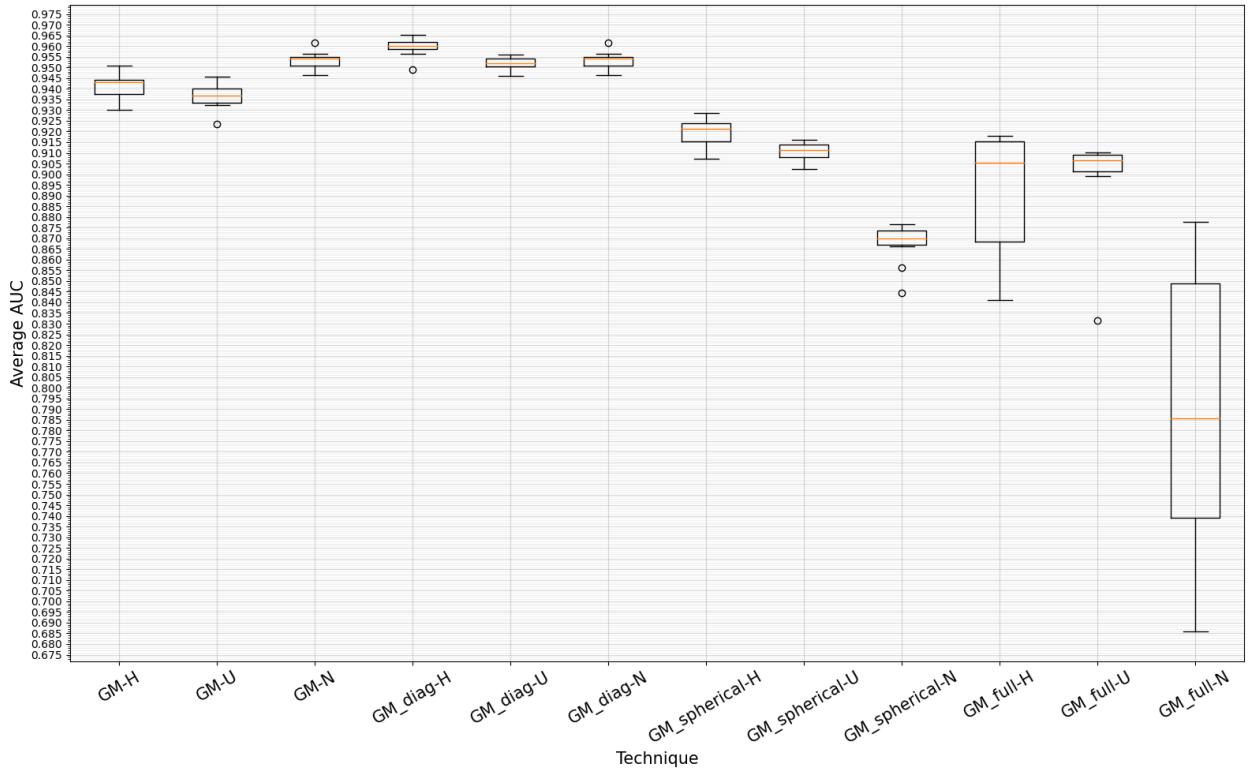
Figure 4.4: Boxplot diagram for the Seven-known-classes scenario considering the distance-based novelty detection scores.

Table 4.1: Medians and p -values for the comparisons between multiple techniques and topologies for the distance-based models case (see text).

Scenario	Technique	Hierarchical	Unanimous	Non-specialized	p -value
3	kNN-median	0.9370 *	0.9346 →	0.9291	$p > 0.001009$
	kNN-mean	0.9358	0.9351 →	0.9262	$p > 0.001009$
	LOF	0.9189 →	0.9111	0.9319	$p > 0.037036$
	NNd	0.8772	0.8678	0.9018	$p > 0.001000$
	k-means	0.9431 →	0.9431 →	0.9444	$p > 0.9$
	p-value	$p > 0.001000$	$p > 0.001000$	$p > 0.001000$	$p > 0.001000$
5	kNN-median	0.9091 ↓ *	0.9069 → ↓	0.8922	$p > 0.001000$
	kNN-mean	0.8979	0.8961 →	0.8852	$p > 0.001000$
	LOF	0.8891 →	0.8765	0.9164 *	$p > 0.001000$
	NNd	0.8671 →	0.8416	0.8841	$p > 0.001000$
	k-means	0.9195	0.9158 →	0.9029 ↓	$p > 0.001000$
	p-value	$p > 0.001000$	$p > 0.001000$	$p > 0.001000$	$p > 0.001000$
7	kNN-median	0.8647 ↓ *	0.8488 → ↓	0.8342 ↓	$p > 0.001000$
	kNN-mean	0.8731	0.8585 →	0.8342 ↓	$p > 0.001000$
	LOF	0.8257 →	0.7679	0.8567	$p > 0.001000$
	NNd	0.8123	0.7103	0.8008 →	$p > 0.001009$
	k-means	0.8707 ↓ *	0.8485 → ↓	0.7934	$p > 0.001000$
	p-value	$p > 0.001000$	$p > 0.001000$	$p > 0.001000$	$p > 0.001000$



(a) Three-known-class scenario.



(b) Five-known-classes scenario.

Figure 4.5: Boxplot diagram for the three and Five-known-classes' scenario considering the mixture-models-based detection scores.

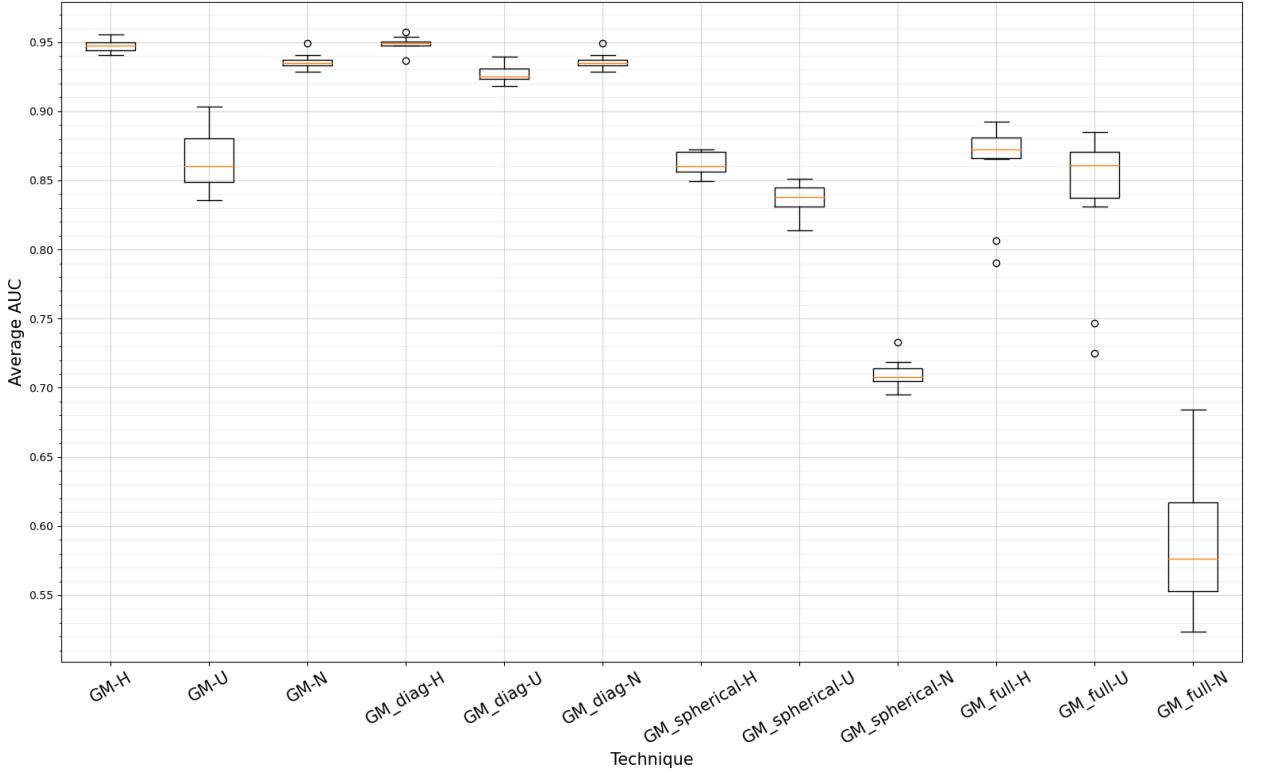


Figure 4.6: Boxplot diagram for the Seven-known-classes scenario considering the mixture-models-based detection scores.

Table 4.2 depicts the main results, which can be summarized as follows:

- Three-known-class scenario: the best performing topology is the Hierarchical, except for GM with the Non-specialized topology and diagonal covariance. The diagonal covariance type contains the highest medians for all topologies and is only statistical equivalent to the Hierarchical topology with GM. The global highest median is achieved by the Hierarchical topology with the diagonal covariance type, which has also shown to be statistical equivalent to the Non-specialized topology with diagonal covariance.
- Five-known-classes scenario: For the diagonal and spherical covariance types, the Hierarchical topology performed best. In the case of the full covariance type, the best topology is Unanimous. For the mix of covariance types, the best topology is again the Non-specialized. The best overall covariance type is diagonal. The global highest median is achieved by the Hierarchical topology with diagonal covariance, which has shown to be statistical equivalent to the Non-specialized topology with the mix of covariance types.

- Seven-known-classes scenario: the Hierarchical topology and the diagonal covariance figured out as the best design options, which has shown to be statistical equivalent to the Hierarchical topology with GM.
- This analysis can be summarized as follows: the Hierarchical topology and the diagonal covariance performs best for most evaluation scenarios.

Table 4.2: Medians and p -values for the comparisons between multiple techniques and topologies for the mixture-models case (see text).

Scenario	Technique	Hierarchical	Unanimous	Non-specialized	p -value
3	GM	0.9450 ↓	0.9403	0.9594 ↓ *	$p > 0.001009$
	GM-Diag	0.9602	0.9539	0.9594 →	$p > 0.037036$
	GM-Spherical	0.9319	0.9277 →	0.9319 →	$p > 0.631856$
	GM-Full	0.9362	0.9302 →	0.9280	$p > 0.037036$
	<i>p</i> -value	$p > 0.001000$	$p > 0.001000$	$p > 0.001572$	$p > 0.001000$
5	GM	0.9432 ↓	0.9367 ↓	0.9540 ↓ *	$p > 0.001009$
	GM-Diag	0.9602	0.9520	0.9540	$p > 0.001009$
	GM-Spherical	0.9214	0.9114 →	0.8699	$p > 0.001000$
	GM-Full	0.9054 →	0.9064	0.7856	$p > 0.002297$
	<i>p</i> -value	$p > 0.001000$	$p > 0.001000$	$p > 0.017062$	$p > 0.001000$
7	GM	0.9476 ↓ *	0.8604 ↓	0.9347 ↓ →	$p > 0.001000$
	GM-Diag	0.9492	0.9252	0.9347	$p > 0.001000$
	GM-Spherical	0.8602	0.8377 →	0.7079	$p > 0.001000$
	GM-Full	0.8725	0.8608 →	0.5764	$p > 0.002297$
	<i>p</i> -value	$p > 0.001000$	$p > 0.001000$	$p > 0.00100$	$p > 0.001000$

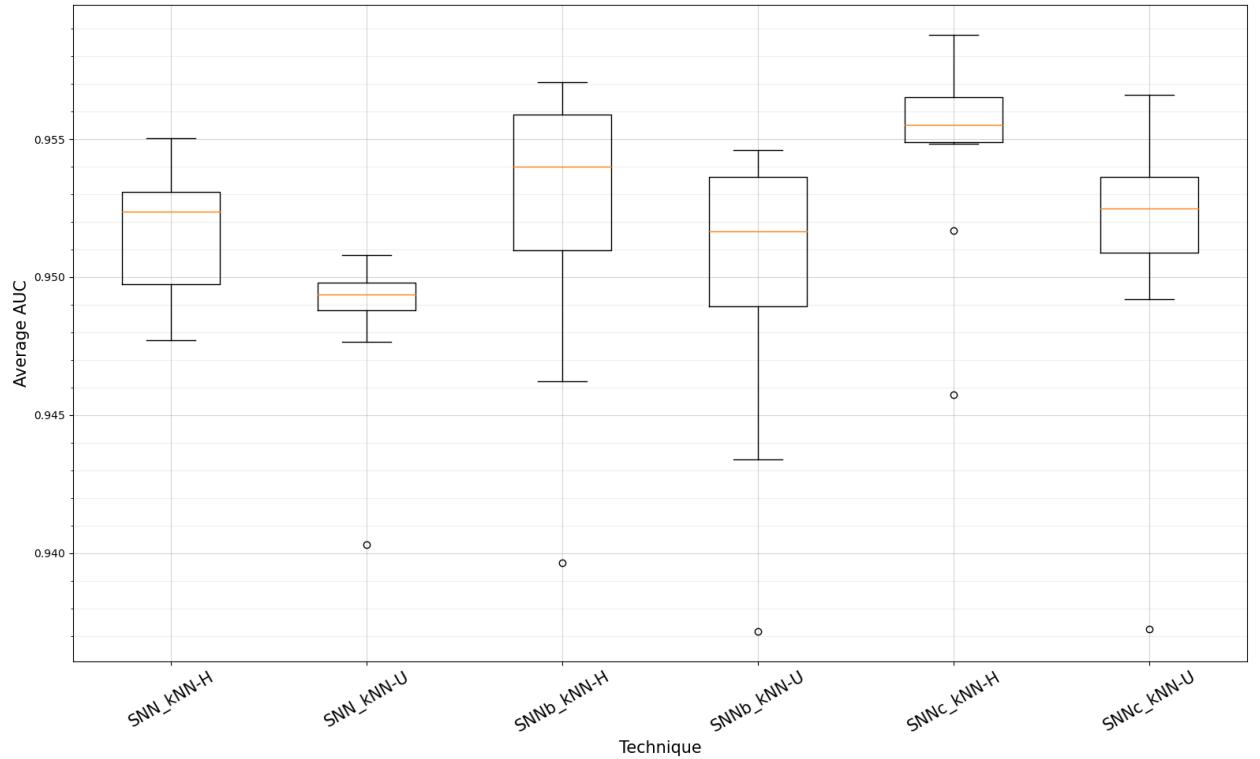
4.4 Neural-Networks-based techniques

Figures 4.7-4.12 depict the boxplot graphs for the Neural-Networks-based novelty detection methods, which include the Siamese Neural Networks following two models: SNN (e.g., with all the non-classes assumed as a unique non-class) and SNNb

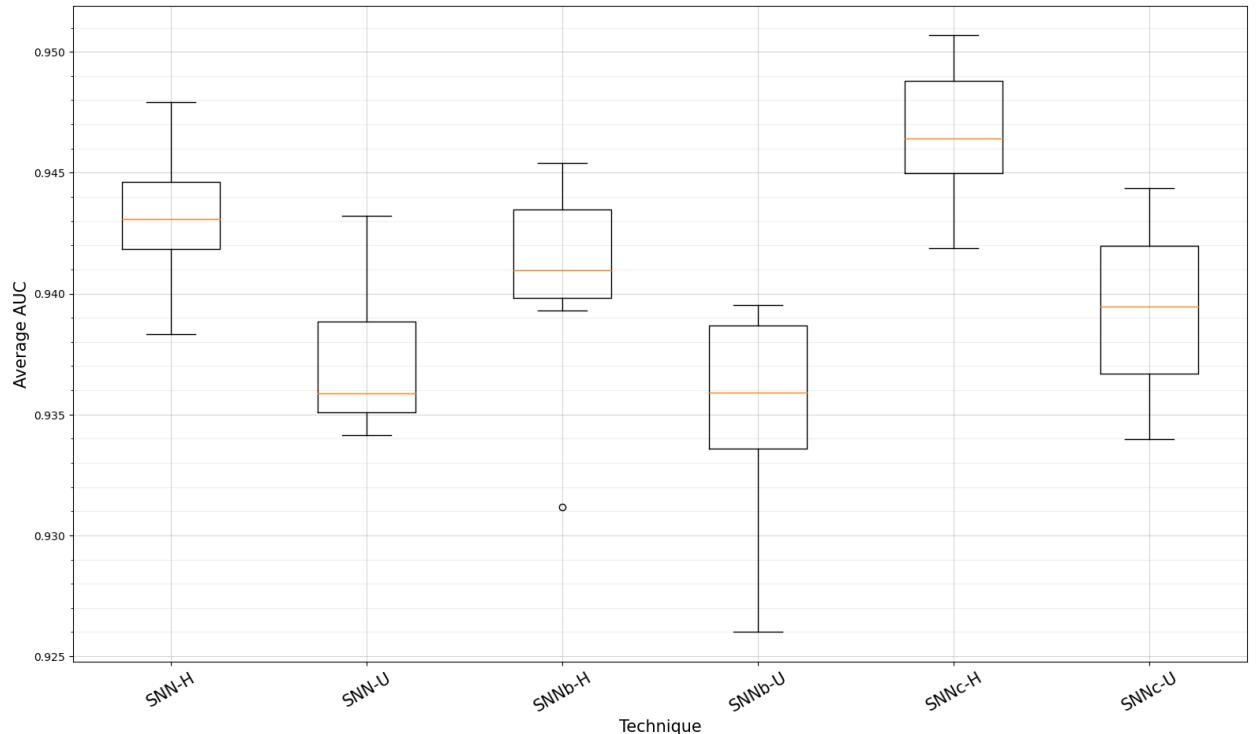
(e.g., with each non-class individually considered). SNNc refers to the best model chosen between SNN and SNNb for each class (for more details please refer to the Appendix A). In turn, SNN-*k*NN, SNNb-*k*NN, and SNNc-*k*NN refers to models that exploit the output vector of each of respective SNN as the input of a subsequent *k*NN novelty detector, as described in Chapter 2. Finally, LSTM refers to the LSTM Autoencoder.

The Friedman test reported that these methods performed differently ($\chi^2(15) > 115.1; p < 0.001$). Table 4.3 depicts the main results, which can be summarized as follows:

- Three-known-class scenario: the best topology for all methods was the Hierarchical. The LSTM Autoencoder was the best method for all topologies, being statistically equivalent to methods exploiting *k*NN with SNN’s mappings.
- Five-known-classes scenario: the best topology in all cases was the Hierarchical. SNNb-*k*NN was the best method for both the Hierarchical and Unanimous topologies, while the LSTM Autoencoder when considering the Non-specialized topology. The highest global median was achieved by the SNNc-*k*NN, which has also shown to be statistically equivalent to SNNb-*k*NN, SNNb, SNNc, and the LSTM Autoencoder.
- Seven-known-classes scenario: once more the Hierarchical topology performed best. SNNb was the best method for the Hierarchical and Unanimous topologies, while the LSTM Autoencoder when considering the Non-specialized topology, which has also shown to be statistically equivalent to ODIN. The highest global median was obtained by the SNNb, which has similarly shown to be statistically equivalent to SNNb-*k*NN, SNNc-*k*NN, and SNNc.
- In summary, the Hierarchical topology and the LSTM Autoencoder were the best design choices for the Three-known-class scenario, while the Siamese Neural Networks approaches performed best for the five and Seven-known-classes scenarios.

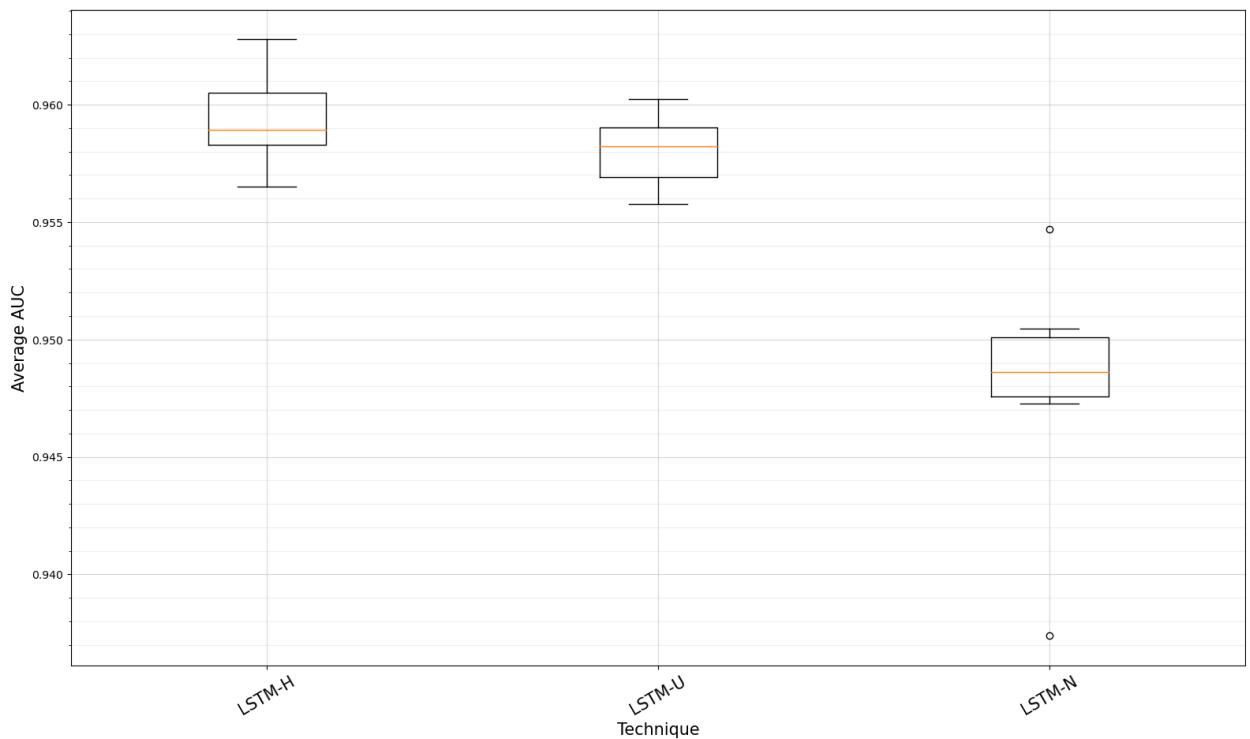


(a) Three-known-class scenario for detectors based on SNN and k NN (see text).

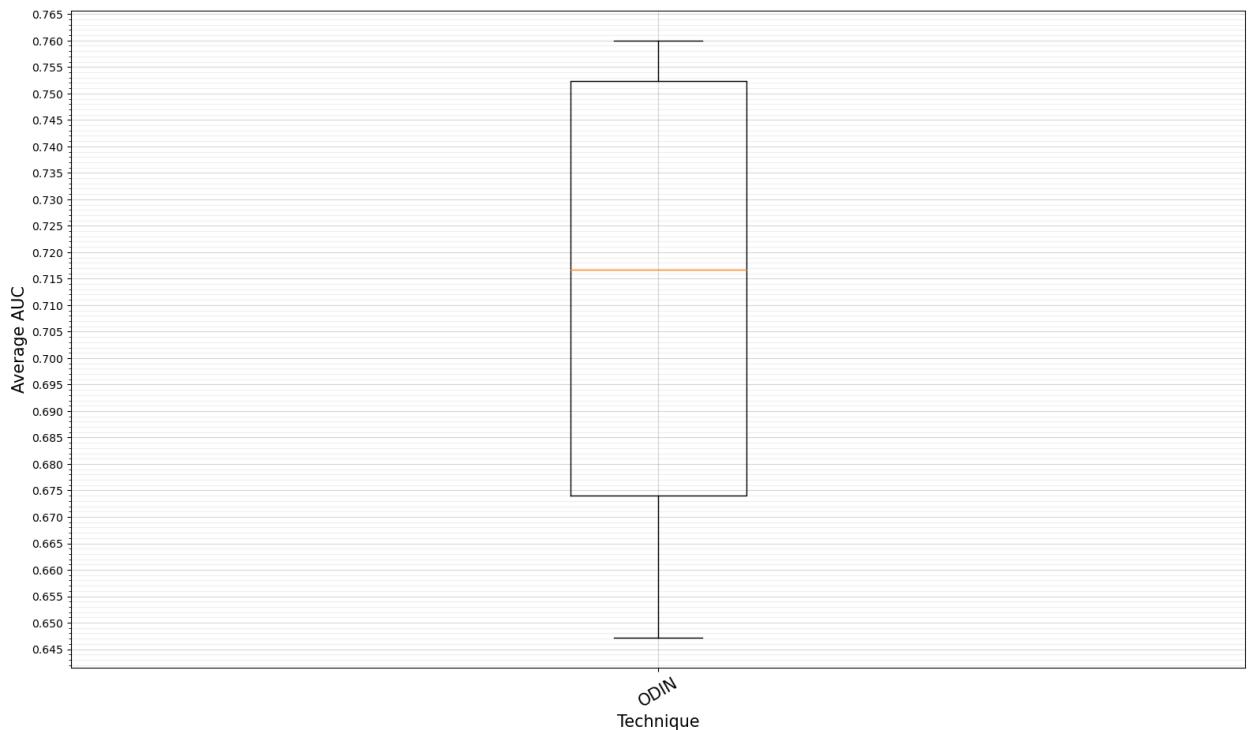


(b) Three-known-class scenario for SNNs.

Figure 4.7: Results for the Neural-Network-based novelty detection techniques (part I).

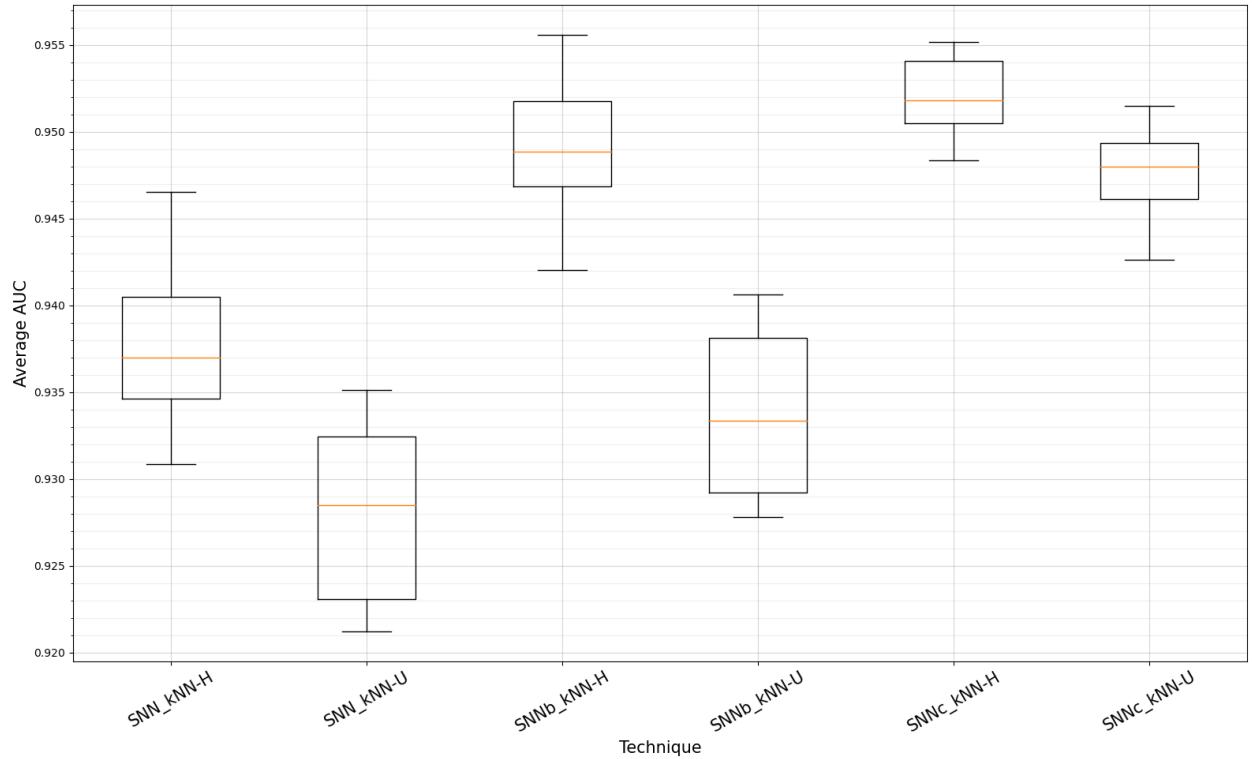


(a) Three-known-class scenario for the LSTM Autoencoder.

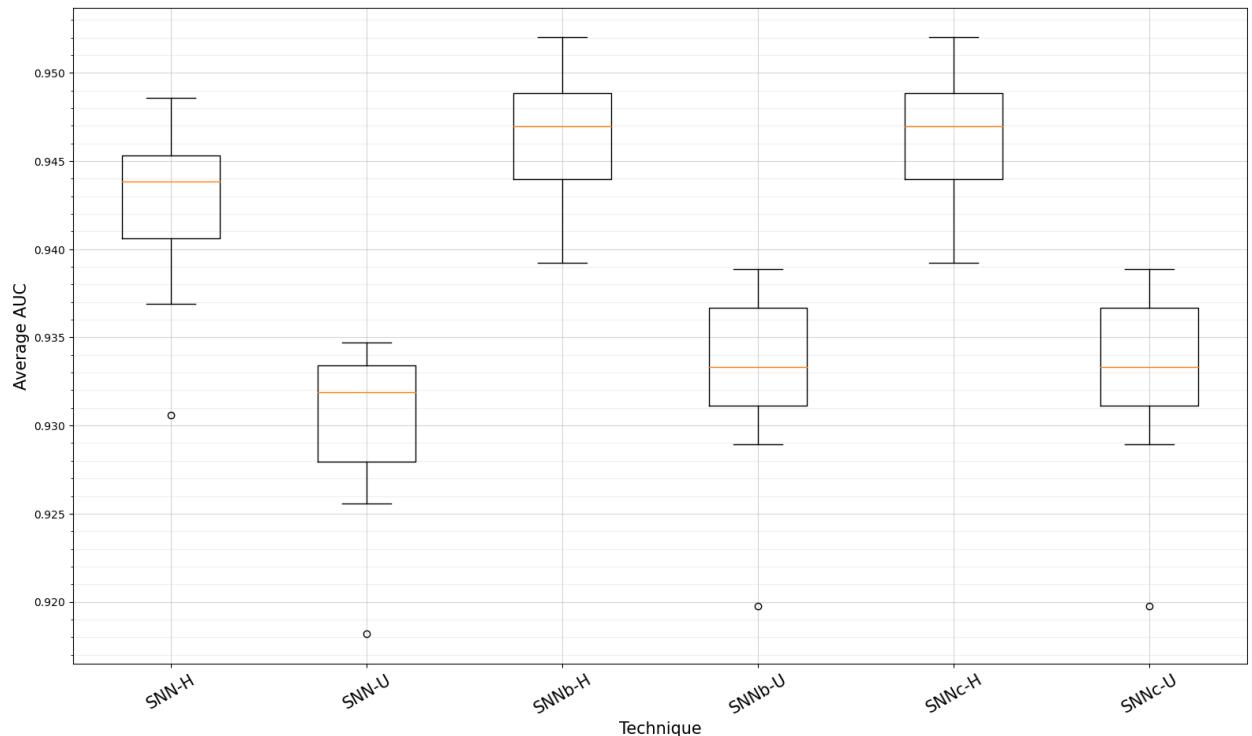


(b) Three-known-class scenario for the ODIN.

Figure 4.8: Results for the Neural-Network-based novelty detection techniques (part II).

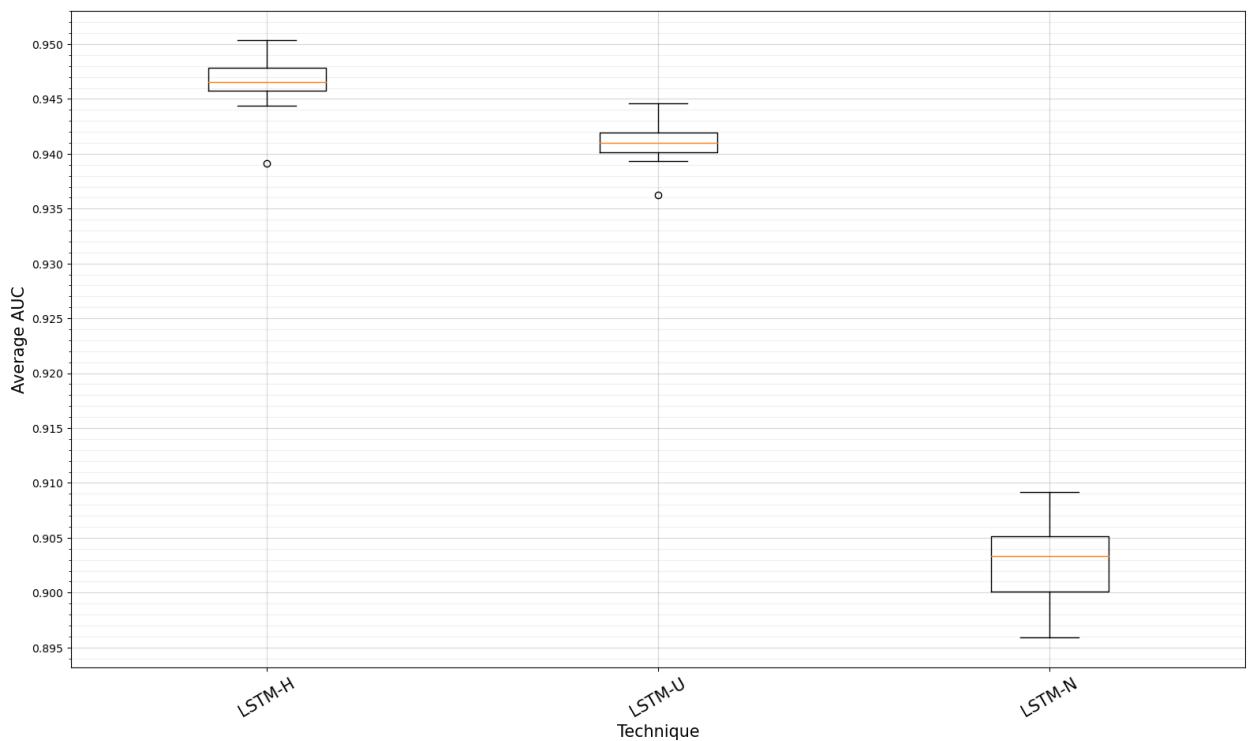


(a) Five-known-class scenario for detectors based on SNN and k NN (see text).

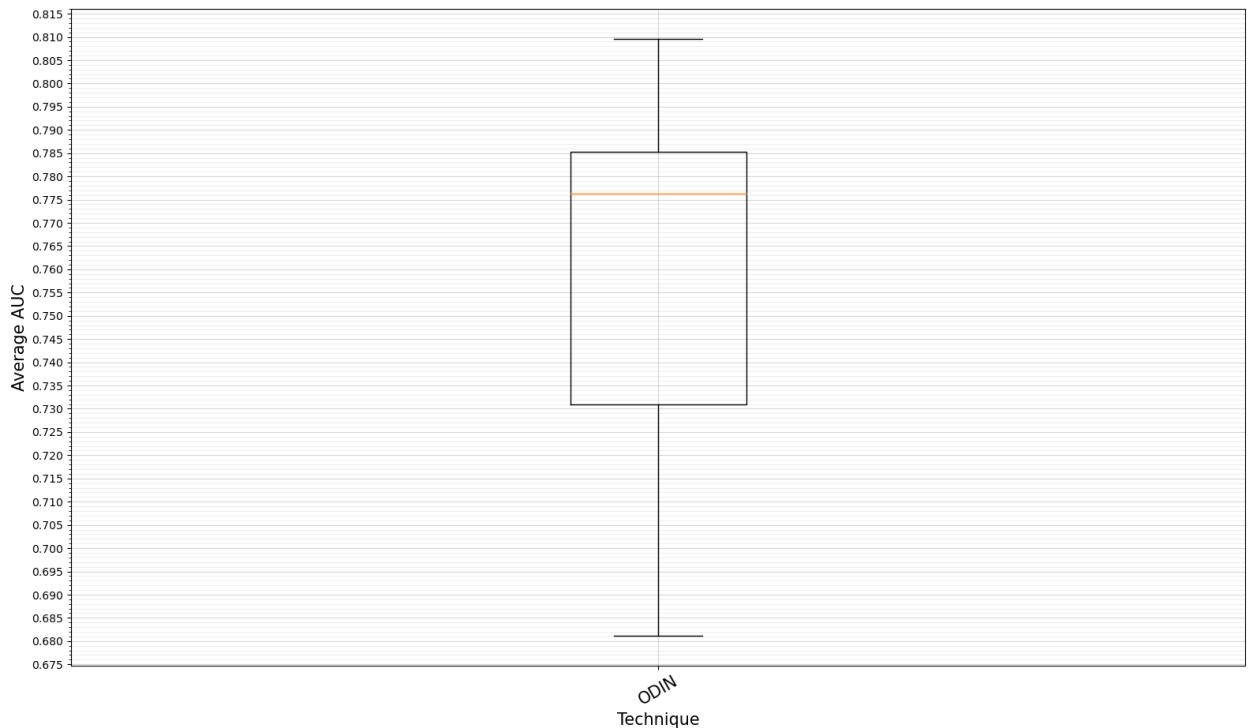


(b) Five-known-class scenario for SNNs.

Figure 4.9: Results for the Neural-Network-based novelty detection techniques (part III).

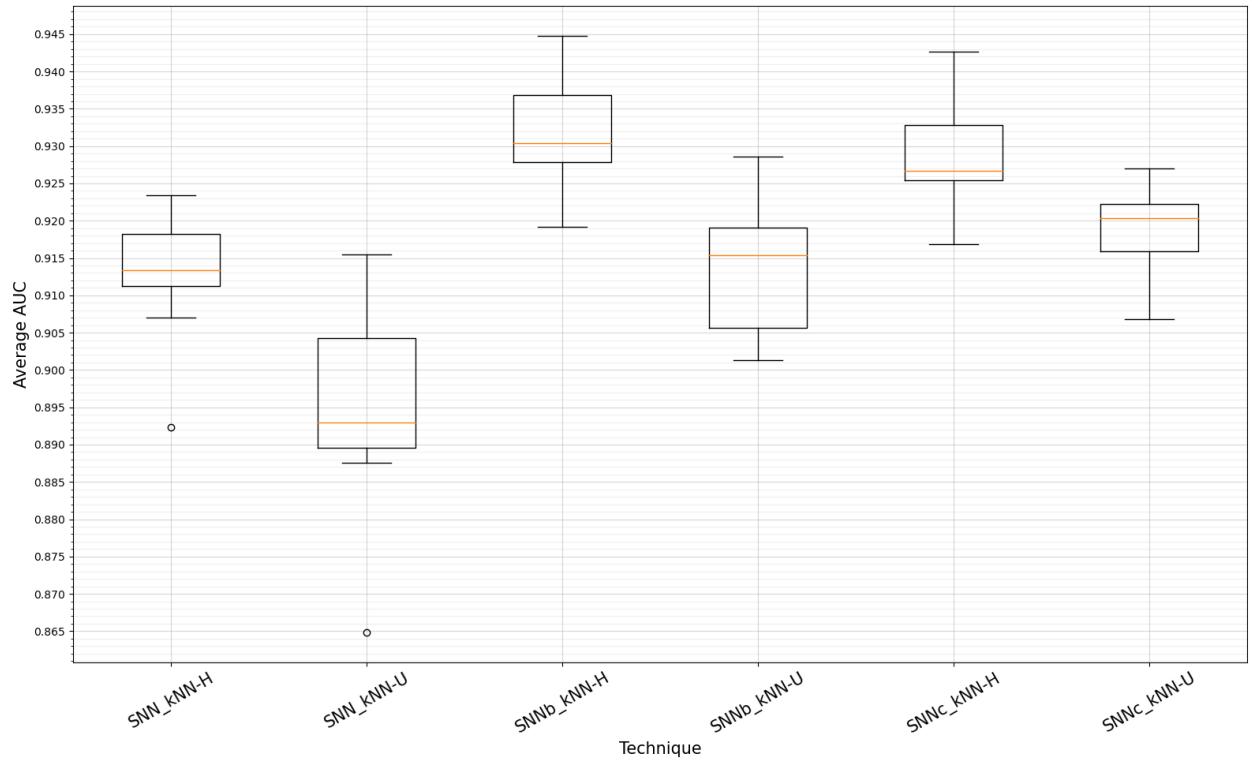


(a) Five-known-class scenario for the LSTM Autoencoder.

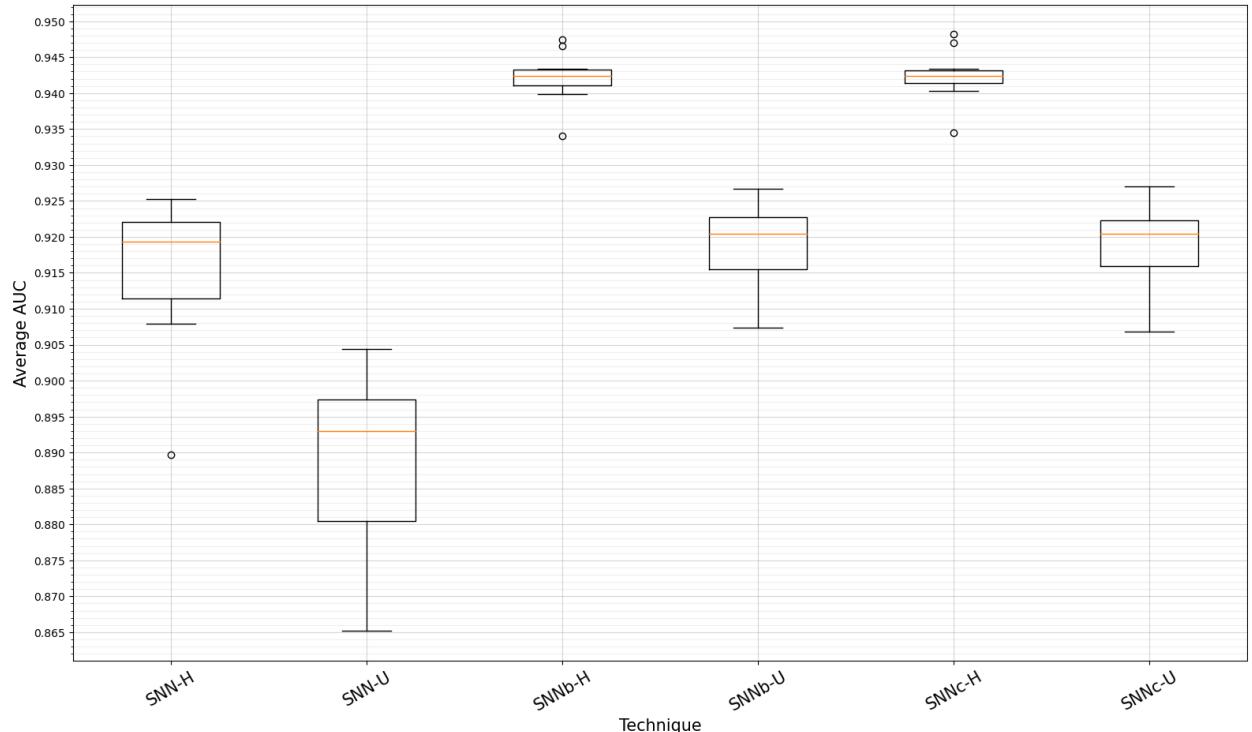


(b) Five-known-class scenario for the ODIN.

Figure 4.10: Results for the Neural-Network-based novelty detection techniques (part IV).

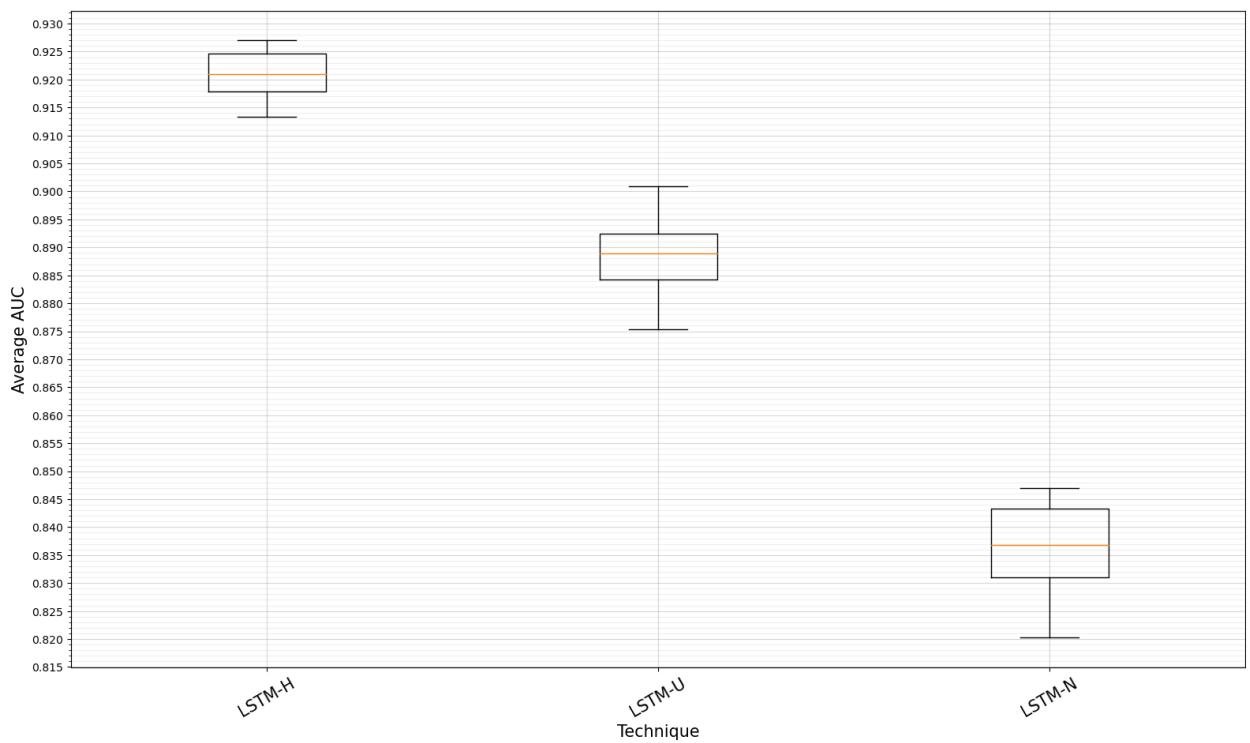


(a) Seven-known-class scenario for detectors based on SNN and k NN (see text).

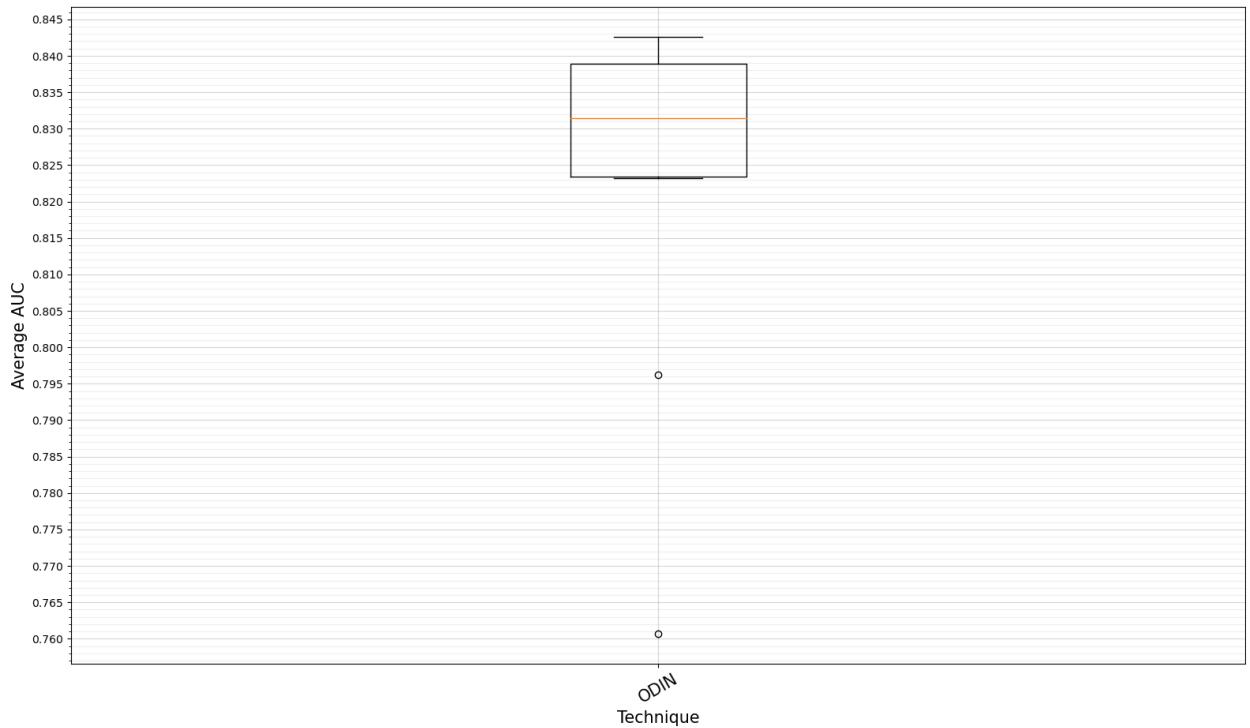


(b) Seven-known-class scenario for SNNs.

Figure 4.11: Results for the Neural-Network-based novelty detection techniques (part V).



(a) Seven-known-class scenario for the LSTM Autoencoder.



(b) Seven-known-class scenario for the ODIN.

Figure 4.12: Results for the Neural-Network-based novelty detection techniques (part VI).

Table 4.3: Medians and p -values for the comparisons between multiple techniques and topologies for the Neural-Networks-based models case (see text).

Scenario	Technique	Hierarchical	Unanimous	Non-specialized	p -value
3	SNN_kNN	0.9524 ↓ *	0.9494 ↓	-	$p=0.001563$
	SNNb_kNN	0.9540 ↓ *	0.9517 ↓ →	-	$p=0.057779$
	SNNc_kNN	0.9555 ↓ *	0.9525 ↓	-	$p=0.011413$
	SNN	0.9431	0.9359	-	$p=0.001563$
	SNNb	0.9410	0.9359	-	$p=0.001563$
	SNNc	0.9464	0.9395	-	$p=0.001563$
	ODIN	-	-	0.7167	-
	LSTM	0.9589	0.9582 →	0.9486	$p>0.001000$
	<i>p</i> -value	$p >0.001000$	$p >0.001000$	$p=0.001563$	$p >0.001000$
5	SNN_kNN	0.9370	0.9285	-	$p=0.001563$
	SNNb_kNN	0.9488 ↓ *	0.9333	-	$p=0.001563$
	SNNc_kNN	0.9518	0.9480	-	$p=0.001563$
	SNN	0.9438	0.9319	-	$p=0.001563$
	SNNb	0.9470 * ↓	0.9333	-	$p=0.001563$
	SNNc	0.9470 * ↓	0.9333	-	$p=0.001563$
	ODIN	-	-	0.7762	-
	LSTM	0.9466 *	0.9410 → ↓	0.9486	$p>0.001000$
	<i>p</i> -value	$p >0.001000$	$p >0.001000$	$p=0.001563$	$p >0.001000$
7	SNN_kNN	0.9133	0.8930 ↓	-	$p=0.001563$
	SNNb_kNN	0.9304 ↓ *	0.9154 ↓	-	$p=0.001563$
	SNNc_kNN	0.9267 ↓ *	0.9204 →	-	$p=0.057779$
	SNN	0.9194	0.8930	-	$p=0.001563$
	SNNb	0.9424	0.9204	-	$p=0.001563$
	SNNc	0.9424 ↓ *	0.9204 ↓	-	$p=0.001563$
	ODIN	-	-	0.8314 ↓	-
	LSTM	0.9209	0.8890 →	0.8368	$p>0.001000$
	<i>p</i> -value	$p >0.001000$	$p >0.005411$	$p=0.205903$	$p >0.001000$

4.5 Reconstruction and Domain-based techniques

Figures 4.13 and 4.14 shows the results for the KPCA and One-Class SVM (OCSVM) techniques. The Friedman test reported ($\chi^2(5) > 47.6; p < 0.001$) that for all evaluation scenarios the methods performed differently. Table 4.4 summarizes the main results. Clearly, the best design options in this case are the Hierarchical topology with KPCA.

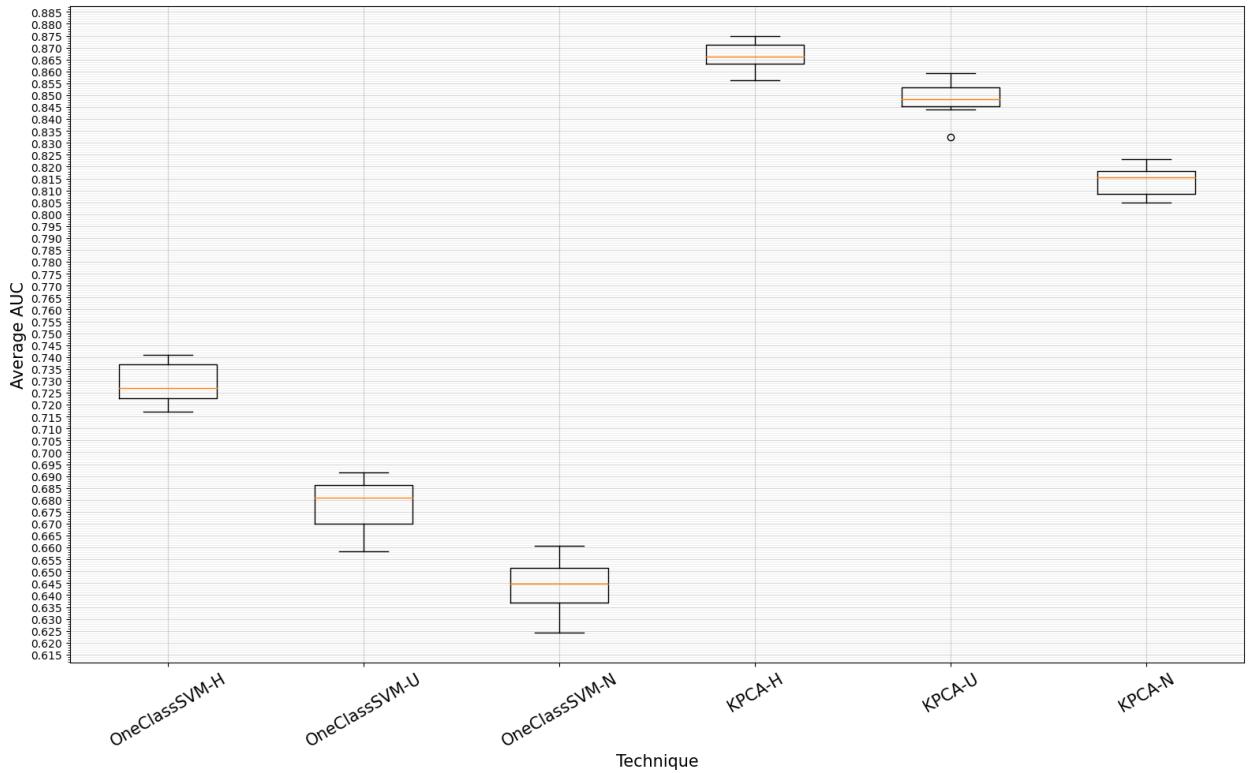
Table 4.4: Medians and p -values for the comparisons between multiple techniques and topologies for the reconstruction and domain based models case (see text).

Scenario	Technique	Hierarchical	Unanimous	Non-specialized	p -value
3	OCSVM	0.7270	0.6809 →	0.6448	$p>0.001000$
	KPCA	0.8662	0.8485 →	0.8154	$p>0.001000$
5	p -value	$p=0.001563$	$p=0.0015634$	$p=0.001563$	$p=0.001563$
	OCSVM	0.6592	0.5133	0.5182	$p>0.001000$
7	KPCA	0.8059	0.7785 →	0.6740	$p>0.001000$
	p -value	$p=0.001563$	$p=0.001563$	$p=0.001563$	$p=0.001563$
	OCSVM	0.4	0.12	0.2027 →	$p>0.001000$
	KPCA	0.7620	0.6514	0.6243	$p>0.001000$
	p -value	$p=0.001563$	$p=0.001563$	$p=0.001563$	$p=0.001563$

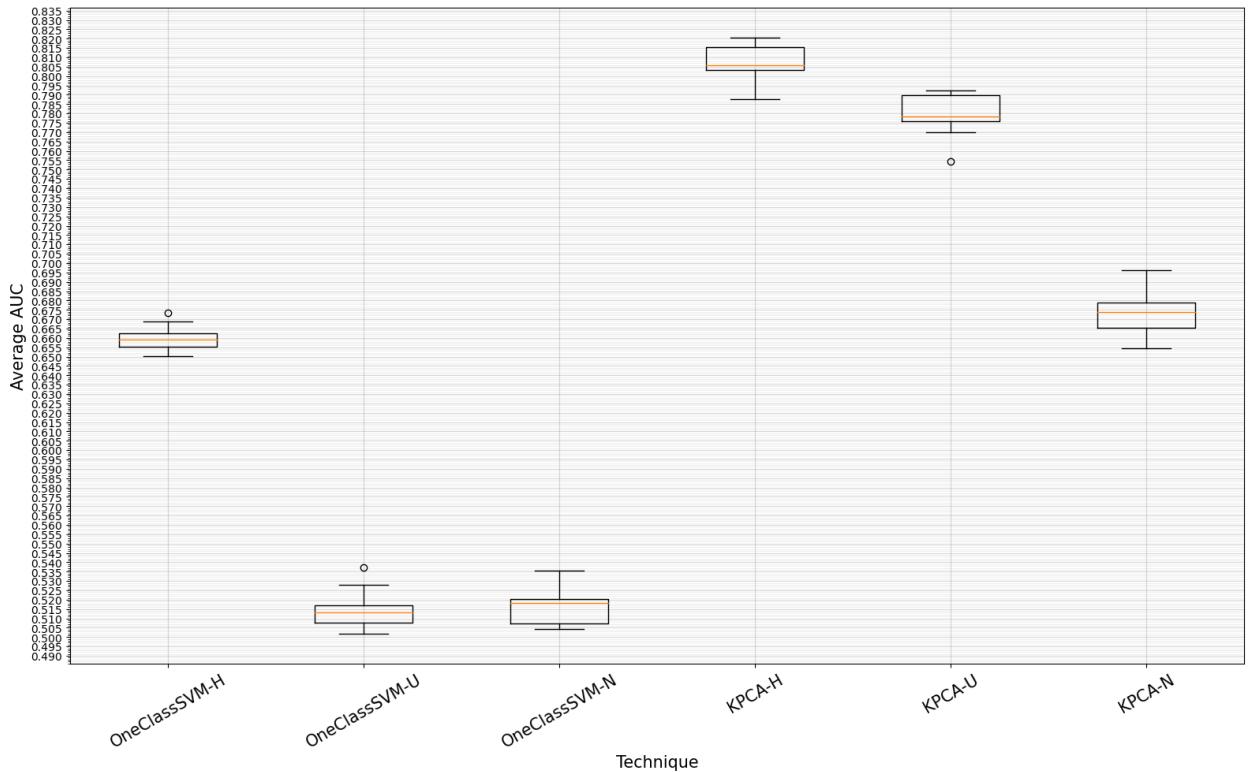
4.6 Comparison of the best performing methods

Figures 4.15 and 4.16 exhibit a comparison between the best performing methods identified for each previous group. It is important to note that all topologies in this analysis are Hierarchical. The final results can be summarized as follows:

- Three-known-class: the highest median is achieved by the Gaussian Mixture model with the diagonal covariance type, which has shown to be statistically equivalent to the LSTM ($p=0.9$).
- Five-known-classes: the highest median corresponds to the Gaussian Mixture



(a) Three-known-class scenarios for the OCSVM and KPCA.



(b) Five-known-classes scenarios for the OCSVM and KPCA.

Figure 4.13: Boxplot diagram for the three and Five-known-classes scenario considering the reconstruction and domain novelty detection scores.

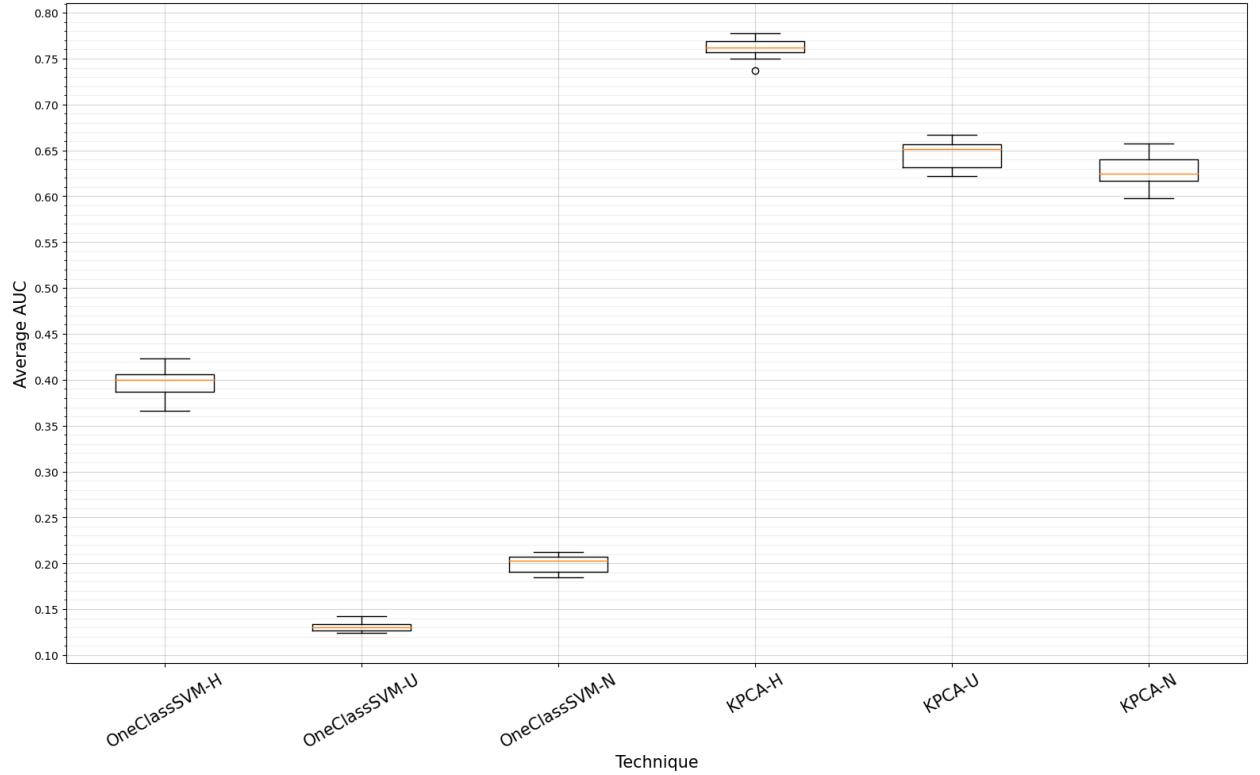
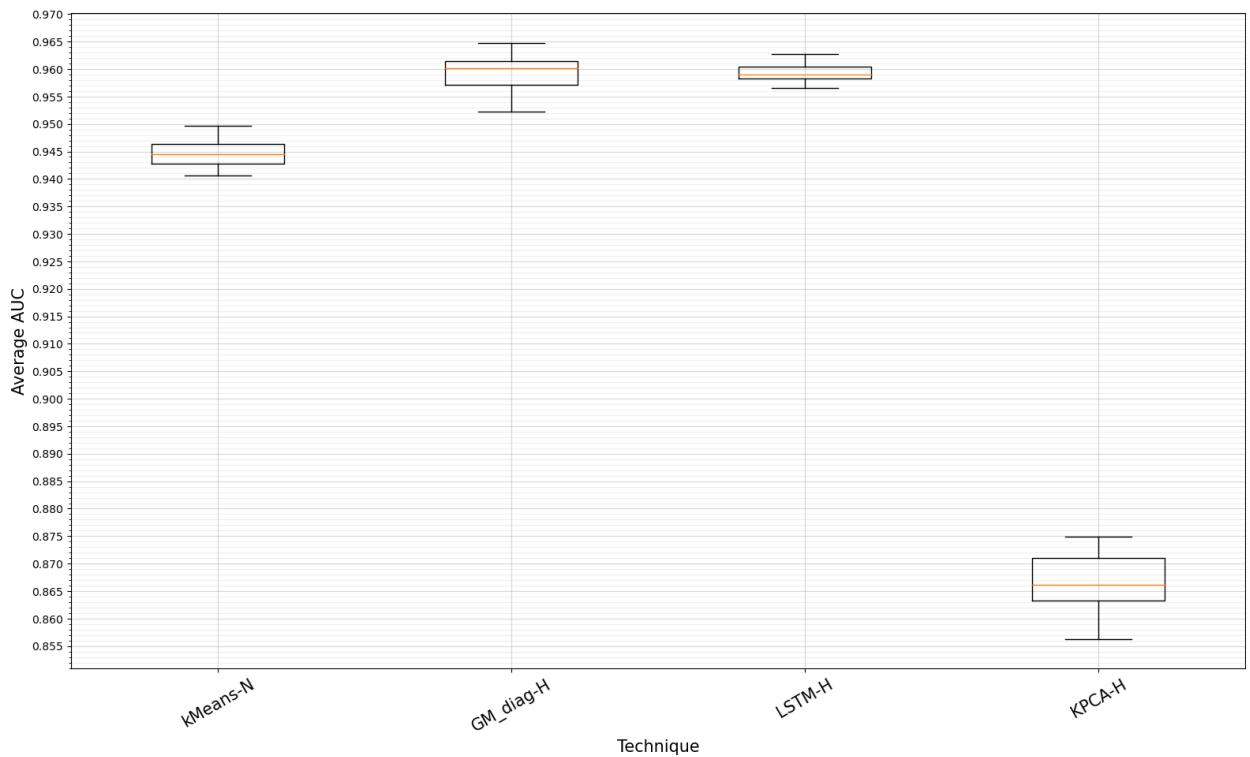


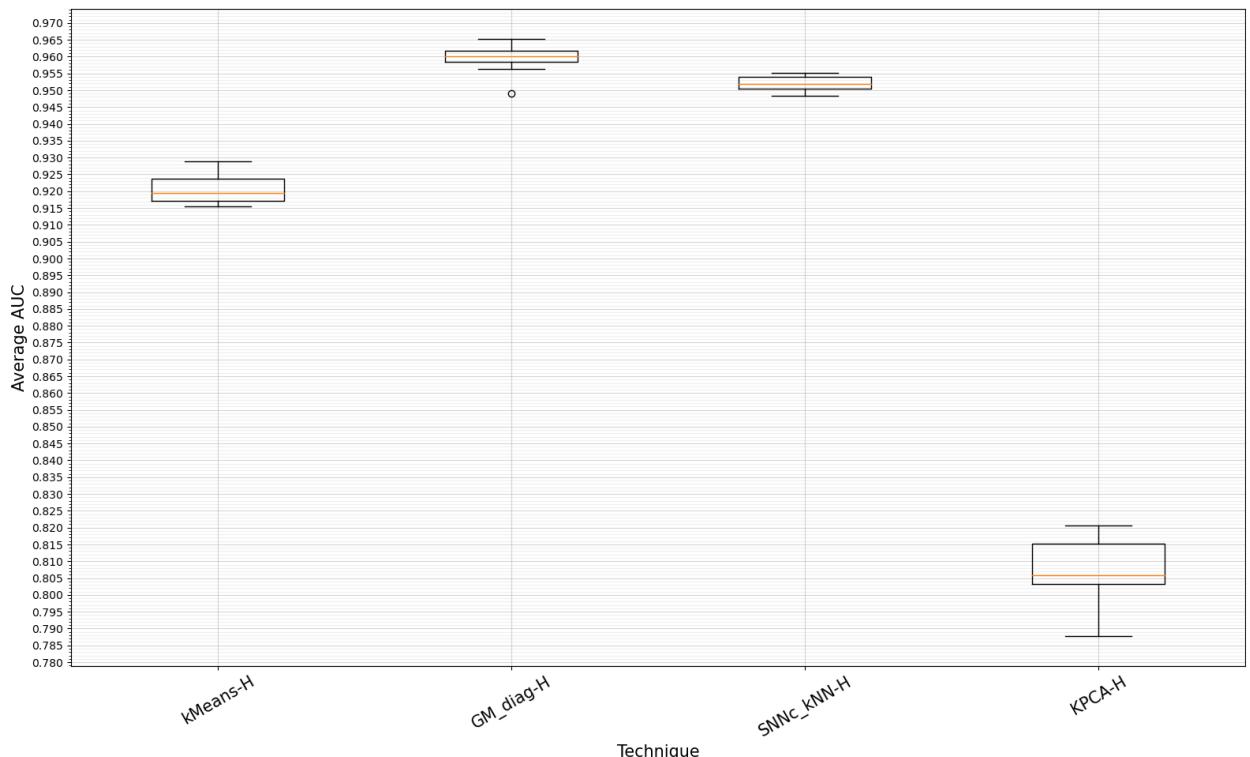
Figure 4.14: Boxplot diagram for the Seven-known-classes scenario considering the reconstruction and domain novelty detection scores.

model with the diagonal covariance type, which has shown to be statistically equivalent to the SNNc_kNN ($p=0.507386$).

- Seven-known-classes: the highest median was obtained with the Gaussian Mixture model with the diagonal covariance type, which has shown to be statistically equivalent to the SNNb ($p=0.507386$).



(a) Three-known-class scenario



(b) Five-known-classes scenario

Figure 4.15: Boxplot diagram comparing the best performing methods identified for each group (part I).

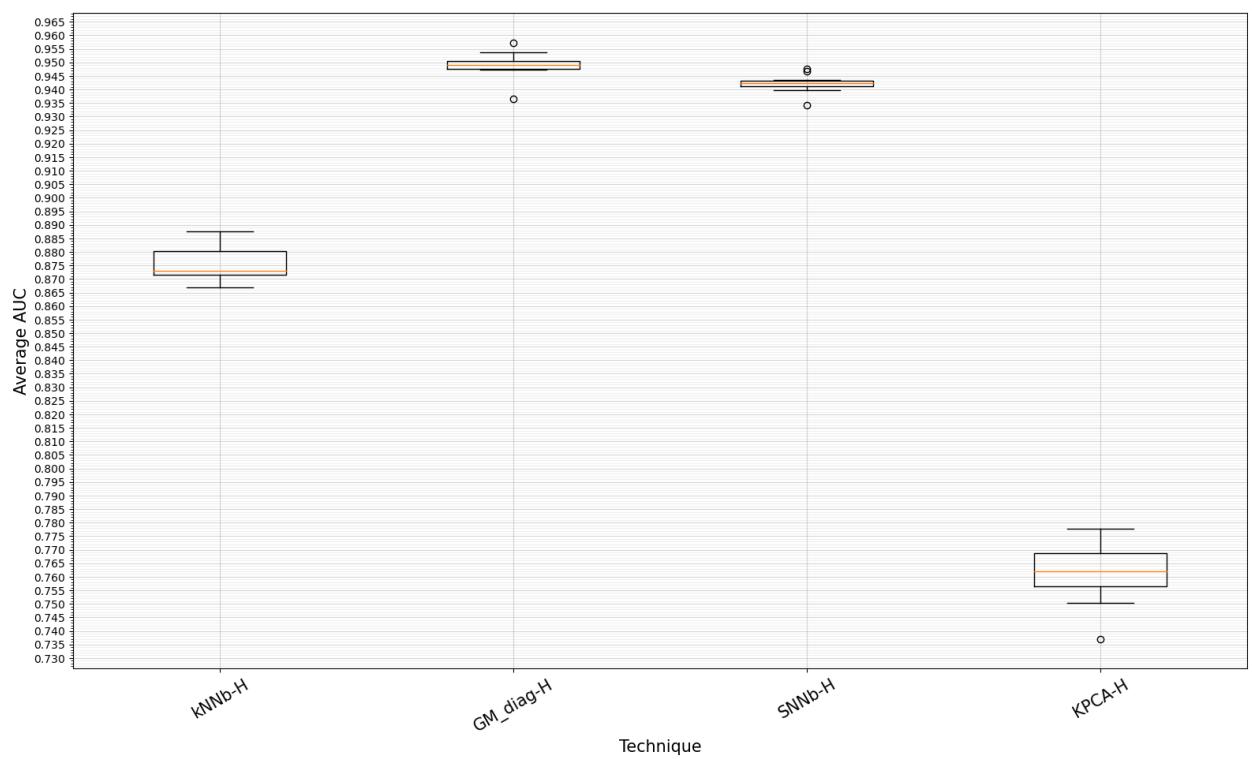


Figure 4.16: Boxplot diagram comparing the best performing methods identified for each group (part II).

Chapter 5

Conclusion

This work addressed the problem of detecting new classes of contact (vessels) in passive sonar signals exploiting class-specialized Machine Learning (ML) detectors. Two modalities of class specialization were considered in this case: Hierarchical and Unanimous. In the former, a high-performance classifier has the role of identifying which known class the event under observation is more similar to, triggering a specialized detector responsible for defining if it represents a novelty or not. In the latter, the same event is submitted to many detectors (one to each class) simultaneously, and a novelty is identified only when all detectors agree. For the sake of simplicity, the Hierarchical models considered a k NN classifier.

In both cases, a comprehensive study of ML techniques for defining the novelty score generators were conducted, including Distance-based (k NN, LOF, NND and k-Means), Domain-based (One Class SVM), Reconstruction-based (KPCA), Probabilistic-based (GMM), and Neural Network approaches (ODIN, Autoencoder and Siamese Neural Network). The AUC was the figure of merit for all detectors evaluation.

This study exploited a dataset provided by the IPqM, composed by signals irradiated by eight classes of vessels, which were acquired by a hydrophone situated in the seabed, during experimental runs conducted in an acoustic lane from the Brazilian Navy. These signals were windowed (without overlap) and submitted to a processing chain that outputs a vector with 557 dimensions, containing the spectral content of the noise irradiated by the vessels' engines in the range from 0 to 3kHz.

Based on the experiments, it is possible to claim that the best design option for the novelty detector in general is the Hierarchical approach (the exception is the three-known-classes case), considering the following techniques for the novelty score generation:

1. Three-known-classes scenario: k -means, Gaussian Mixture with diagonal covariance, LSTM and KPCA. Except for the k -means, which pointed in favor of the Non-specialized topology, the Hierarchical topology is the best.
2. Five-known-classes scenario: k -means, Gaussian Mixture with diagonal covariance, SNNc with k NN, and KPCA.
3. Seven-known-classes scenario: k NN-mean, Gaussian Mixture with diagonal covariance, SNNb and KPCA.

In overall terms, the Gaussian Mixture model with diagonal covariance was the best design option when considering all evaluation scenarios.

In future works, the selection strategies of hard and semi-hard online triplets [39] will be explored for training the Siamese Neural Network with the triplet loss function. These strategies tries to impose a more strict and harder separation task between positive and negative samples, which may lead to better model generalization and consequently detector performance. And also is intended to explore Generative Neural Networks models for novelty detector, such as GAN [53] and Variational Autoencoder [54] models.

Bibliography

- [1] HONORATO, E. S., Souza Filho, J. B. O., MUNIZ, V. H. D. S., “A Hierarchical Ensemble of LSTM-based Autoencoders For Novelty Detection In Passive Sonar Systems”, *LACCI*, , 2021.
- [2] ROCHA, G. G. M. D., *Detecção de Novidades Em Sistemas de Sonar Passivo*. M.Sc. dissertation, CEFET/RJ, Julho 2016.
- [3] MUNIZ, V. H. D. S., *Detectores de Novidades e Classificadores Especializados em Sistemas de Sonar Passivo*. M.Sc. dissertation, UFRJ, Julho 2019.
- [4] MUELLE, A., “Github”, <https://github.com/scikit-learn/scikit-learn/issues/10863>, 2018, (Acesso em 27 Janeiro 2022).
- [5] HONORATO, E. S., MUNIZ, V. H. D. S., Souza Filho, J. B. O., “Detecção Hierárquica de Classes Desconhecidas em Sonar por ’Autoencoders’ Convolucionais”, *SBrT*, , 2020.
- [6] WEI, Y., JANG-JACCARD, J., XU, W., *et al.*, “LSTM-Autoencoder based Anomaly Detection for Indoor Air Quality Time Series Data”, 2022.
- [7] KOCH, G., ZEMEL, R., SALAKHUTDINOV, R., “Siamese Neural Networks For One-Shot Image Recognition”, *ICML Deep Learning Workshop*, v. 2, 2015.
- [8] Souza Filho, J. B. O., *Classificação Neural De Sinais De Sonar Passivo*. Ph.D. dissertation, UFRJ, Julho 2007.
- [9] PIMENTEL, M. A., CLIFTON, D. A., CLIFTON, L., *et al.*, “A Review Of Novelty Detection”, *Signal Processing*, v. 99, pp. 215–249, 2014.
- [10] MURPHY, K. P., *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

- [11] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., *Deep Learnng*, chapter 5, MIT Press, p. 99, 2016.
- [12] MUNIZ, V. H. D. S., Souza Filho, J. B. O., HONORATO, E. S., “Aprendizado por Instância para a IdentificaÇão de Classes Desconhecidas em Sonares Passivos”, *SBIC*, , 2020.
- [13] WANG, Y., WONG, J., MINER, A., “Anomaly intrusion detection using one class SVM”. In: *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pp. 358–364, 2004.
- [14] HEJAZI, M., SINGH, Y. P., “ONE-CLASS SUPPORT VECTOR MACHINES APPROACH TO ANOMALY DETECTION”, *Applied Artificial Intelligence*, v. 27, n. 5, pp. 351–366, 2013.
- [15] PERDISCI, R., GU, G., LEE, W., “Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems”. In: *Sixth International Conference on Data Mining (ICDM'06)*, pp. 488–498, 2006.
- [16] MUNIYANDI, A. P., RAJESWARI, R., RAJARAM, R., “Network Anomaly Detection by Cascading K-Means Clustering and C4.5 Decision Tree algorithm”, *Procedia Engineering*, v. 30, pp. 174–182, 2012. International Conference on Communication Technology and System Design 2011.
- [17] SPINOSA, E. J., DE LEON F. DE CARVALHO, A. P., GAMA, J. a., “OLINDDA: A Cluster-Based Approach for Detecting Novelty and Concept Drift in Data Streams”. In: *Proceedings of the 2007 ACM Symposium on Applied Computing, SAC '07*, pp. 448–452, New York, NY, USA, 2007.
- [18] HOFFMANN, H., “Kernel PCA For Novelty Detection”, *Pattern Recognition*, v. 40, n. 3, pp. 863–874, 2007.
- [19] XIAO, Y., WANG, H., XU, W., *et al.*, “L1 norm based KPCA for novelty detection”, *Pattern Recognition*, v. 46, n. 1, pp. 389–396, 2013.
- [20] XIAO, Y., WANG, H., XU, W., “Model selection of Gaussian kernel PCA for novelty detection”, *Chemometrics and Intelligent Laboratory Systems*, v. 136, pp. 164–172, 2014.

- [21] ZHOU, X., LIANG, W., SHIMIZU, S., *et al.*, “Siamese Neural Network Based Few-Shot Learning for Anomaly Detection in Industrial Cyber-Physical Systems”, *IEEE Transactions on Industrial Informatics*, v. 17, n. 8, pp. 5790–5798, 2021.
- [22] UTKIN, L. V., ZABOROVSKY, V. S., LUKASHIN, A. A., *et al.*, “A Siamese Autoencoder Preserving Distances for Anomaly Detection in Multi-robot Systems”. In: *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pp. 39–44, 2017.
- [23] TAKIMOTO, H., SEKI, J., SITUJU, S. F., *et al.*, “Anomaly Detection Using Siamese Network with Attention Mechanism for Few-Shot Learning”, *Applied Artificial Intelligence*, v. 36, n. 1, pp. 2094885, 2022.
- [24] BREUNIG, M. M., KRIEGEL, H.-P., NG, R. T., *et al.*, “LOF: Identifying Density-Based Local Outliers”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’00, pp. 93–104, New York, NY, USA, 2000.
- [25] TAX, D. M. J., DUIN, R. P. W., “Outlier detection using classifier instability”. In: Amin, A., Dori, D., Pudil, P., *et al.* (eds.), *Advances in Pattern Recognition*, pp. 593–601, Berlin, Heidelberg, 1998.
- [26] BOUNSIAR, A., MADDEN, M. G., “One-Class Support Vector Machines Revisited”. In: *2014 International Conference on Information Science & Applications (ICISA)*, pp. 1–4, 2014.
- [27] SCHÖLKOPF, B., WILLIAMSON, R. C., SMOLA, A., *et al.*, “Support Vector Method For Novelty Detection”. In: Solla, S., Leen, T., Müller, K. (eds.), *Advances in Neural Information Processing Systems*, v. 12, 1999.
- [28] SCHÖLKOPF, B., SMOLA, A., MÜLLER, K.-R., “Nonlinear Component Analysis as a Kernel Eigenvalue Problem”, *Neural Computation*, v. 10, n. 5, pp. 1299–1319, 1998.

- [29] GUEDES, R. M., Souza Filho, J. B. O., “Identificação de Classes Desconhecidas em Sinais de Sonar Passivo através de Componente Principais”, *Congresso Brasileiro de Automática Sociedade Brasileira de Automática*, , 2014.
- [30] HONORATO, E. S., MUNIZ, V. H. D. S., Souza Filho, J. B. O., “Clustering-and-Bagging-based Ensemble for Novelty Detection in Passive Sonar Systems”. In: *2022 IEEE Latin American Conference on Computational Intelligence (LACCI)*, pp. 1–6, 2022.
- [31] BISHOP, C. M., *Pattern Recognition And Machine Learning*, chapter 9, Springer, p. 430, 2006.
- [32] BALDI, P., “Autoencoders, Unsupervised Learning, and Deep Architectures”. In: Guyon, I., Dror, G., Lemaire, V., *et al.* (eds.), *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, v. 27, *Proceedings of Machine Learning Research*, pp. 37–49, Bellevue, Washington, USA, 02 Jul 2012.
- [33] KINGMA, D. P., BA, J., “Adam: A Method for Stochastic Optimization”, *International Conference on Learning Representations (ICLR)*, , 2015.
- [34] TSURUOKA, Y., TSUJII, J., ANANIADOU, S., “Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty”. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 477–485, Suntec, Singapore, Aug. 2009.
- [35] HOCHREITER, S., SCHMIDHUBER, J., “Long Short-Term Memory”, *Neural Comput.*, v. 9, n. 8, pp. 1735–1780, nov 1997.
- [36] BISHOP, C. M., “Training with Noise is Equivalent to Tikhonov Regularization”, *Neural Computation*, v. 7, n. 1, pp. 108–116, 1995.
- [37] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., *et al.*, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, *Journal of Machine Learning Research*, v. 15, n. 56, pp. 1929–1958, 2014.

- [38] NANDY, A., HALDAR, S., BANERJEE, S., *et al.*, “A Survey on Applications of Siamese Neural Networks in Computer Vision”. In: *2020 International Conference for Emerging Technology (INSET)*, pp. 1–5, 2020.
- [39] SCHROFF, F., KALENICHENKO, D., PHILBIN, J., “FaceNet: A Unified Embedding for Face Recognition and Clustering”, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, , Jun 2015.
- [40] XUAN, H., STYLIANOU, A., LIU, X., *et al.*, “Hard negative examples are hard, but useful”. In: *European Conference on Computer Vision*, pp. 126–142, Springer, 2020.
- [41] LIANG, S., LI, Y., SRIKANT, R., “Enhancing the Reliability of Out-of-distribution Image Detection in Neural Networks”, 2020.
- [42] DINIZ, P., DA SILVA, E., NETTO, S., *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, 2002.
- [43] NIELSEN, R., *Sonar Signal Processing*, Acoustics Library. Artech House, 1991.
- [44] JAPKOWICZ, N., SHAH, M., *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.
- [45] ABADI, M., AGARWAL, A., BARHAM, P., *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems”, 2015, Software available from tensorflow.org.
- [46] PASZKE, A., GROSS, S., MASSA, F., *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 8024–8035, 2019.
- [47] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., *et al.*, “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011.
- [48] TEAM, T. P. D., “pandas-dev/pandas: Pandas”, Feb. 2020.
- [49] HARRIS, C. R., MILLMAN, K. J., WALT, S. J. V. D., *et al.*, “Array programming with NumPy”, *Nature*, v. 585, n. 7825, pp. 357–362, Sep. 2020.

- [50] HUNTER, J. D., “Matplotlib: A 2D graphics environment”, *Computing in Science & Engineering*, v. 9, n. 3, pp. 90–95, 2007.
- [51] VIRTANEN, P., GOMMERS, R., OLIPHANT, T. E., *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”, *Nature Methods*, v. 17, pp. 261–272, 2020.
- [52] TERPILOWSKI, M. A., “scikit-posthocs: Pairwise multiple comparison tests in Python”, *Journal of Open Source Software*, v. 4, n. 36, pp. 1169, 2019.
- [53] SCHLEGL, T., SEEBÖCK, P., WALDSTEIN, S. M., *et al.*, “Unsupervised anomaly detection with generative adversarial networks to guide marker discovery”. In: *International conference on information processing in medical imaging*, pp. 146–157, Springer, 2017.
- [54] STEINMANN, L., MIGENDA, N., VOIGT, T., *et al.*, “Variational Autoencoder Based Novelty Detection for Real-World Time Series”. In: *2021 3rd International Conference on Management Science and Industrial Engineering*, MSIE 2021, pp. 1–7, New York, NY, USA, 2021.

Appendix A

Hyperparameters' selection

This appendix describes in more detail the process of hyperparameters' selection. The first section addresses the choice of the hyperparameters required by the k NN classifiers explored in the Hierarchical topology. In the sequence, the following content will be split into two big groups: the Non-neural models and the Neural-Networks-based models, since the tuning of the latter followed an incremental approach.

A.1 k NN classifier

The k NN classifier explored by the Hierarchical model was tunned using the accuracy as the figure of merit inferred over the validation set. The hyperparameters chosen for each evaluation scenario and the median of the accuracy in the 10 folds are described in Table A.1.

Table A.1: Hyperparameters investigated and chosen for the k NN classifier.

Class	Number of neighbors chosen	Hyperparameter range	Accuracy
ACG	9	1-30	0.98531
ABC ^G	11	1-30	0.98538
ABCEFGH	11	1-30	0.97887

A.2 Non-neural models

Tables A.2, A.3 and A.4 show the range of hyperparameters investigated and chosen for the three evaluation scenarios considering the Non-neural models. In the case of GM entries, the letters indicates the covariance type, i.e., full (F) or diagonal (D). For the SNN_kNNc technique, the model chosen was the best performing between SNN_kNNa and SNN_kNNb. The following tables include a column called NSD that represents the hyperparameters of the detector in the case of the Non-specialized topology.

Table A.2: Hyperparameters investigated and chosen for the three-known-class scenario.

Technique	A	C	G	NSD	Hyperparameter range
<i>k</i>NN-Mean	6	2	15	6	1-30
<i>k</i>NN-Median	7	2	15	8	1-30
LOF	12	4	21	3	2-30
NNd	1	1	2	1	1-5
<i>k</i>-Means	325	425	275	625	25-750
KPCA	45	60	30	40	5-95
GM_full	125	195	75	245	5-255
GM_diag	35	155	255	205	5-255
GM_spherical	145	255	15	255	5-255
GM	125-F	195-F	255-D	205-D	-
SNNa_kNN	1	2	1	-	1-30
SNNb_kNN	1	2	3	-	1-30
SNNc_kNN	a	b	a	-	-

A.3 Neural-Networks-based models

A.3.1 LSTM Autoencoder

The hyperparameters chosen for the LSTM Autoencoder models are summarized in Table A.5, where each column represents a network layer, and for each entry there

Table A.3: Hyperparameters investigated and chosen for the five-known-classes scenario.

Technique	A	B	C	F	G	NSD	Hyperparameter range
kNN-Mean	5	2	2	2	15	4	1-30
kNN-Median	5	2	2	2	15	5	1-30
LOF	14	11	4	9	19	3	2-30
NNd	1	1	1	1	1	1	1-5
k-Means	325	750	425	725	275	725	25-750
KPCA	45	80	60	50	30	75	5-95
GM_full	105	155	255	255	115	245	5-255
GM_diag	35	85	145	85	125	255	5-255
GM_spherical	155	255	255	155	95	255	5-255
GM	35-D	85-D	255-F	85-D	125-D	255-D	-
SNN_kNNa	2	3	1	3	4	-	1-30
SNN_kNNb	4	1	2	5	12	-	1-30
SNN_kNNc	b	a	b	b	a	-	-

Table A.4: Hyperparameters investigated and chosen for the seven-known-classes scenario.

Technique	A	B	C	E	F	G	H	NSD	Hyperparameter range
kNN-Mean	5	2	2	2	2	15	14	2	1-30
kNN-Median	9	2	2	4	3	15	17	2	1-30
LOF	10	11	4	4	8	20	28	3	2-30
NNd	1	1	1	1	1	1	1	1	1-5
k-Means	200	750	425	750	725	175	225	750	25-750
KPCA	45	80	60	85	50	35	30	90	5-95
GM_full	105	215	255	255	135	75	245	15	5-255
GM_diag	35	85	145	135	85	125	85	255	5-255
GM_spherical	155	225	255	255	125	115	85	255	5-255
GM	35-D	85-D	255-S	135-D	85-D	125-D	85-S	255-D	-
SNN_kNNa	6	1	3	5	4	5	14	-	1-30
SNN_kNNb	2	4	2	23	2	1	2	-	1-30
SNN_kNNc	b	b	b	b	a	b	b	-	-

is a tuple indicating the number of LSTM units in the layer, the layer’s activation functions (S for SeLU; R for ReLU, and L for LeakyReLU), the noise standard deviation value (restricted to the first layer), and the dropout rate (also restricted to the first layer), respectively. There is also a column entitled NSD that refers to the Non-specialized topology.

A.3.2 Siamese Neural Networks

In this case, the tuples will be composed by the number of neurons, the activation function, the dropout rate (restricted to the first layer), and the weight factor (restricted to the last layer). Table A.6 and A.7 summarizes the results for SNNa and SNNb. Table A.8 indicates which SNN model (SNNa or SNNb) was chosen for composing the SNNc model.

Table A.5: Hyperparameters chosen for the LSTM Autoencoder.

Scenario	Class	1st Layer	2nd Layer	3rd Layer	4th Layer
3	A	{512, S, 0.07, 0.3}	{512, R}		
3	C	{512, S, 0.07, 0.3}	{512, R}	{256, S}	
3	G	{512, R, 0.02, 0.3}	{256, R}		
3	NSD	{512, S, 0.07, 0.3}	{512, R}		
5	A	{512, S, 0.07, 0.3}	{512, R}		
5	B	{512, S, 0.07, 0}			
5	C	{512, S, 0.07, 0.3}	{512, R}	{256, S}	
5	F	{512, S, 0.08, 0.0}	{256, S}		
5	G	{512, R, 0.02, 0.3}	{256, R}		
5	NSD	{512, S, 0.07, 0.0}	{512, S}	{512, S}	
7	A	{512, S, 0.0, 0.4}	{512, S}		
7	B	{512, S, 0.07, 0}			
7	C	{512, S, 0.07, 0.05}	{256, S}	{256, R}	{256, S}
7	E	{256, S, 0.06, 0.0}	{256, L}		
7	F	{512, S, 0.05, 0.05}	{128, S}		
7	G	{512, R, 0.02, 0.1}	{512, L}		
7	H	{512, L, 0.09, 0.25}	{512, S}		
7	NSD	{512, S, 0.07, 0.0}	{512, S}	{512, S}	

A.4 ODIN

ODIN hyperparameters are the temperature and the disturbance magnitude. The first one considered values from the list [1, 10, 100, 1000], and for the second, we have the values [0, 0.1, 0.1, 0.01, 0.001]. The Neural Network model was a MLP with 512 neurons at the first layer and the subsequent layers have the number of neurons from the immediately previous layer halved until this value reached 8 neurons. The activation function in each layer was the ReLU. The hyperparameters chosen are presented in Table A.9.

Table A.6: Hyperparameters chosen for the SNNa architecture.

Scenario	Class	1st Layer	2nd Layer	3rd Layer
3	A	{512, L, 0.1, 0.1}		
3	C	{512, S, 0.15, 1}		
3	G	{512, L, 0.0, 0.01}		
5	A	{512, L, 0.0}	{512, L, 0.01}	
5	B	{512, S, 0.45, }	{512, L}	{256, L, 1}
5	C	{512, L, 0.0, 0.1}		
5	F	{512, S, 0.45, 1}		
5	G	{256, L, 0.0, 0.01}		
7	A	{512, L, 0.0, 0.1}		
7	B	{512, S, 0.4, }	{256, S, 1}	
7	C	{512, L, 0.0, 0.1}		
7	E	{512, S, 0.0, 0.1}		
7	F	{512, S, 0.15, 1}		
7	G	{256, L, 0.0, 0.01}		
7	H	{256, L, 0.05}	{256, L}	{256, L, 0.1}

Table A.7: Hyperparameters chosen for the SNNb architecture.

scenario	Class	1st Layer	2nd Layer	3rd Layer
3	A	{256, S, 0.0, 0.1}		
3	C	{512, L, 0.05}	{256, S, 1}	
3	G	{512, L, 0.0, 0.01}		
5	A	{512, S, 0.0, 0.1}		
5	B	{512, S, 0.0, 0.1}		
5	C	{128, L, 0.0, 0.01}		
5	F	{512, S, 0.05, 0.1}		
5	G	{128, L, 0.0, 0.01}		
7	A	{256, S, 0.0, 0.1}		
7	B	{512, S, 0.4,}	{512, L}	{512, L, 0.1}
7	C	{512, L, 0.0, 0.1}		
7	E	{512, S, 0.0, 0.1}		
7	F	{512, L, 0.05, 0.1}		
7	G	{256, L, 0.05, 0.1}		
7	H	{256, L, 0.00, 0.1.}		

Table A.8: SNN model (SNNa or SNNb) chosen for each evaluation scenario in the case of the SNNc approach.

scenario	A	B	C	E	F	G	H
3	SNNa	-	SNNb	-	-	SNNa	-
5	SNNb	SNNa	SNNb	-	SNNb	SNNa	-
7	SNNb	SNNb	SNNb	SNNb	SNNa	SNNb	SNNb

Table A.9: ODIN hyperparameters for each evaluation scenario.

Class	Temperature	Disturbance Magnitude
ACG	1	0
ABC ^F G	10	0
ABCEFGH	1	0

Appendix B

Publications

Most of this work was developed during my Scientific Initiation Stage at UFRJ, and has resulted in four conference papers and one journal article, whose abstracts are reproduced below for the reader's convenience.

- Title: Aprendizado por Instância para a Identificação de Classes Desconhecidas em Sonares Passivos.

Authors: Victor Hugo da Silva Muniz, João Baptista de Oliveira e Souza Filho, Eduardo Sperle Honorato.

Conference: XIV Congresso Brasileiro de Inteligência Computacional(2019).

Abstract: In submarines, the task of sonar operators is to identify possible threats (contacts), mainly using the passive sonar system. Automatic contact classification systems require the identification of vessels of unknown classes during their operation. This work discusses the construction of a hierarchical system for the recognition of such occurrences, considering an experimental study involving learning techniques by instance, in scenarios of increasing complexity, for this purpose. The experiments, exploring data collected in acoustic lanes from 28 ships belonging to 8 classes in different operational conditions, showed a better performance of the k-Nearest Neighbors technique, reaching a novelty detection rate of 78.0%, combined with an average rate identification of known cases of 95.0%, for a scenario with 3 known classes.

- Title: Deteção Hierárquica de Classes Desconhecidas em Sonar por "Autoencoders" Convolucionais.

Authors: Eduardo Sperle Honorato, Victor Hugo da Silva Muniz, João Baptista de Oliveira e Souza Filho.

Conference: XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais(2020).

Abstract: Acoustic waves captured by passive sonar systems are analyzed by human operators, aiming to identify possible threats in the subsea environment. Automatic Classification Systems can aid in the work of this professional, however requiring mechanisms to deal with the presence of unknown classes. This article proposes the use of a hierarchical committee of convolutional autoencoders to build these systems, as a more robust alternative to the k-nearest neighbors algorithm, which represents the state-of-the-art in this problem. Real data belonging to 8 classes of ships under different operational conditions were evaluated. Results signalize a competitive performance of the proposed technique.

- Title: A Hierarchical Ensemble of LSTM-based Autoencoders for Novelty Detection in Passive Sonar Systems.

Authors: Eduardo Sperle Honorato, Victor Hugo da Silva Muniz, João Baptista de Oliveira e Souza Filho.

Conference: seventh Latin American Conference on Computational Intelligence(2021).

Abstract: Sonar operators represent a vital workforce for identifying potential threats to submarines (referred to as contacts) by analyzing underwater acoustic signatures acquired by their passive sonar system. Automatic contact classification models may alleviate the sonar operator task but require additional tools for identifying any class of contact not considered in system development. This paper addresses the use of a hierarchical detector of unknown classes of contact for passive sonar systems, considering the modeling of signal spectra by Long Short-Term Memory Autoencoders networks. The proposed system was evaluated with signals from 28 ships belonging to 8 classes, acquired in a Brazilian Navy acoustic range. Such a system achieves an expressive median value for the area under the detection operation curve of

0.946, for a simulation scenario involving 5 known classes, surpassing the state-of-the-art technique.

- Title: Clustering-and-Bagging-based Ensemble for Novelty Detection in Passive Sonar Systems

Authors: Eduardo Sperle Honorato, Victor Hugo da Silva Muniz, João Baptista de Oliveira e Souza Filho.

Conference: Eighth Latin American Conference on Computational Intelligence (2022).

Abstract: Trained submarine operators identify threats through passive sonar systems by analysing the acoustic waves captured by arrays of hydrophones. Automatic Classification Systems may be quite beneficial to address this task; however, typical operational settings require instruments to identify the occurrences of unknown classes of ships for promptly alerting this specialised crew. This article proposes a new approach for developing an accurate novelty detector of unknown classes of ships. This proposal comprises an architecture composed by a synergistic combination of cluster-specialised and bagging generated novelty detectors, in opposition to a previous solution exploiting a hierarchical class-specialised architecture. The proposed approach is experimentally evaluated using radiated noise from 8 classes of ships, acquired in an acoustic range from the Brazilian Navy. The proposed detector outperforms previous works in 4.4% of AUC (on average), assuming an evaluation scenario composed by five known and three supposedly unknown ship classes. This gain is relevant to this performance-critical application and is related to a more strict definition of class boundaries attained by the proposed strategical clustering and bagging combination.

- Title: Instance-based novelty detection in passive sonar signals

Authors: Victor Hugo Da Silva Muniz, João Baptista De Oliveira e Souza Filho, Eduardo Sperle Honorato

Journal: International Journal of Innovative Computing and Applications

Abstract: In submarines, sonar operators have the main task of identifying

potential threats, named as contacts in the military jargon. The principal tool exploited when dealing with such situations is the passive sonar system. Automatic contact classification models may relieve the huge sonar operator workload but require mechanisms capable of identifying any contact not considered during system development. This paper discusses the development of a hierarchical instance-based detector of unknown contact classes for passive sonar signals, focusing on practical strategies for its hyperparameter tuning and performance assessment. Experimental data exploited in system evaluation comprises the acoustic noise irradiated by 28 ships belonging to 8 classes. These ships were submitted to different operational conditions in several runs conducted in an acoustic range. The kNN algorithm has performed best, achieving a novelty detection rate of 78.0%, associated with an average known case identification rate of 95.0%, considering a five unknown class evaluation scenario.