FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Anomaly Detection in Cyber-Physical Production Systems

Carlos Pinto

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Gil Manuel Magalhães de Andrade Gonçalves

Second Supervisor: Rui Pedro Ferreira Pinto

September 17, 2021

Resumo

À luz da quarta revolução industrial, inúmeros setores da Indústria, outrora centralizados e estáticos, estão a tornar-se cada vez mais adaptáveis e tomam a forma de arquiteturas mais descentralizadas. No entanto, como os sistemas são mais plásticos e dinâmicos, são também muito mais imprevisíveis. Aliás, a complexidade ainda aumenta pela relação entre os dispositivos intervenientes. Embora a interligação entre dispositivos providencie oportunidades, também origina interações imprevisíveis com resultados indesejáveis.

O problema a resolver consiste na deteção de anomalias de forma autónoma e flexível em ambientes industriais, com especial atenção dirigida à deteção de falhas. Em particular, assume-se a existência de um processo físico a monitorizar e pretende-se detetar mudanças de comportamento para um dado comportamento normal. Assim, esta dissertação tem os seguintes objectivos: fornecer uma solução de deteção de anomalias que seja escalável, flexível, e forneça resultados atempados; validar a referida solução num cenário representativo de ambiente industrial.

Esta dissertação enquadra-se na temática dos sistemas imunes artificiais, cuja inspiração advém do sistema imunológico dos seres vivos. Esta área de investigação ainda é pouco investigada, apesar de resultados previamente obtidos, que revelam o seu potencial. Este trabalho ambiciona aplicar os conceitos dos sistemas imunes artificiais à deteção de anomalias, mais especificamente, o projeto pretende detar mudanças no comportamento classificado como normal. De acordo com os requisitos da Indústria 4.0, este algoritmo tem a vantagem de ser aplicado em tempo real, ser mais flexível e compatível com escalabilidade.

Tendo liberdade de escolha para o algoritmo a aplicar, foi desenvolvido um estudo comparativo das soluções mais comuns e atuais no âmbito dos Sistemas Imunes Artificiais. A opção recaiu sobre o Algoritmo de Células Dendríticas, que usa um modelo imunulógico mais moderno que as propostas contemporâneas. De entre as contribuições, foi necessário adaptar o algoritmo de forma a monitorizar continuamente o processo e para flexibilizar a fusão da arquitetura com outros métodos.

A metodologia foi desenvolvida e validada com recurso a um conjunto de dados de acesso público que simula um sistema hidráulico simples. O conjunto de dados tem, associado, um conjunto de resultados de outras implementações. Em termos de resultados, verificam-se melhoramentos em relação às demais implementações atualmente constantes na tabela a nível da sua métrica principal - F_1 *score* - em que o melhor resultado obtido com a implementação desenvolvida foi de 0.72, enquanto que o melhor resultado atual na tabela, usando *deep learning*, obteve uma classificação de 0.79. No entanto, constatam-se também alguns defeitos no algoritmo, nomeadamente, no que toca à sua característica de requerer vários dados para classificar pontos individuais (em média, um dado ponto necessita de 7 pontos para ser classificado), o que leva a um inevitável atraso na sua classificação. O algoritmo foi ainda validado num segundo dataset, de forma a corroborar que, com a nova versão do algoritmo, se torna mais fácil modelar problemas.

Com base nos resultados obtidos, é possível identificar trabalho futuro. A validação do algoritmo num ambiente real será essencial para provar a sua aplicabilidade em ambiente industrial.

Desenvolvimentos futuros incluem: a sua otimização a nível de memória e tempo de processamento; a utilização de outras técnicas para geração de sinais; desenvolver uma forma de classificação imediata; acrescentar uma componente espacial (para além de temporal) e tornar o algoritmo descentralizado.

Abstract

The world is undergoing the fourth industrial revolution, where sectors in the industry migrate from centralized, static approaches, to more adaptable and decentralized architectures. The increase in the dynamics of such systems also renders them more unpredictable. Furthermore, devices are more and more connected, opening opportunities for unforeseen interactions that can lead to undesirable effects.

The problem to be solved is the autonomous and flexible detection of anomalies in industrial environments, with a focus on fault detection. In particular, the existence of a physical process is assumed, where any behavioral changes (relative to normal behavior) are to be detected. Thus, this dissertation has the following goals: provide an anomaly detection solution that is scalable, flexible, and provides timely results; validate said solution in a representative scenario of an industrial setting.

In this thesis, an algorithm from the field of Artificial Immune Systems - a field inspired by the human immune system - is adapted for the problem of anomaly detection, i.e., detecting changes in normal behavior. This field is still overlooked, despite its promising results. With Industry 4.0 in mind, the algorithm is made more apt for real-time operation and is adapted to be more flexible and scalable.

The adapted algorithm was a result of analyzing current solutions in Artificial Immune Systems, and thus, the process of analyzing said solutions is also presented. Then, the algorithm that showed the most potential was chosen - the Dendritic Cell Algorithm - which uses a more modern immunology model than previous approaches. This algorithm was adapted to be able to continuously monitor a process and to be flexible by fusing its architecture with another technique.

The algorithm was then developed and validated using a dataset that emulates a simplified water circulation system. The algorithm shows promising results in what concerns classification results when compared to the remaining algorithms in the dataset's benchmark's main metric - F_1 score - scoring, at best, 0.72, with the current best being achieved by a convolutional with autoencoder neural network at a score of 0.79. There were some identified shortcomings as well, namely on how it can not classify data points in a single iteration, and requires further points (on average, 7 points), leading to a delay in classification. The algorithm was also validated on a second dataset, to corroborate that it is easier to model a problem for the algorithm.

Based on the results obtained, future work can be identified. Validation of the algorithm in a real environment will be essential to prove its applicability in an industrial environment. Future developments include: optimizing its memory and processing time; using other techniques for signal generation; developing a form of immediate classification; adding a spatial (in addition to temporal) component; and making the algorithm decentralized.

Agradecimentos

Depois de cinco anos de vivências, desenvolvimento, dúvidas acompanhadas de autodescoberta, e de amizades que certamente manterei para a vida, esta dissertação representa, para além de um projeto, um sinal de conclusão. Conclusão esta que, ainda que pessoal, se deve a todos aqueles que me apoiaram. A quantidade de pessoas que contribuíram, de uma forma ou outra, é infindável, e, apesar de gostar de incluir uma mensagem pessoa a cada um de vós, infelizmente, apenas poderei incluir algumas.

Agradeço ao meu orientador Gil Gonçalves por ter permitido sequer a possibilidade do desenvolvimento desta dissertação e da dinâmica do laboratório DIGI2. Agradeço ao meu coorientador Rui Pinto que me apresentou esta nova perspetiva de algoritmos baseados em sistemas imunitários, um conceito que me fascinou desde o início da dissertação e, para além disso, ele teve de lidar com dificuldades de comunicação e desenvolvimento da minha parte que, confesso, não desejaria a ninguém. Agradeço também ao João Reis por me ter ajudado no processo criativo, tanto a nível da dissertação, como ao nível cósmico. Naturalmente, agradeço também a toda a gente que esteve nas reuniões da quinta-feira e que me tiveram de ouvir a falar da minha tese, e providenciaram um espaço mais informal de diálogo sobre o desenvolvimento da dissertação.

Obrigado à Leonor por me ter aturado vezes sem fim, e constantemente me ter metido em ordem, tanto no que toca à dissertação como no que toca à minha vida pessoal associada a todo este processo. Apesar de te conhecer à menos de um ano, sinto que pude falar abertamente contigo e serviste de grande esponja de ansiedade, mesmo que não tenhas dado conta. Companheiros de tese até ao fim, Leon!

Obrigado ao Cristiano, por me ter ajudado a pôr em perspetiva a minha posição na vida, a valorizar as coisas que importam, e a perceber que o “bicho de sete cabeças” só está mesmo na nossa cabeça e a perceber que é importante refletir constantemente sobre aquilo que temos como garantido. Uma forte contribuição para manter a minha sanidade mental. Obrigado, Cristiano!

Obrigado ao Paulo, João, Matos, Xavi, Pedro, Sara, Andreia, Rafaela e Filipe. Sei que não fui um amigo fácil de se ter nestes últimos meses mas apesar de tudo sempre me fizeram sentir bem vindo e não poderia pedir amigos mais compreensivos. Não importa quanto tempo ficamos sem falar, sinto que podemos reunir como se nada fosse, sem qualquer problema. Em particular, obrigado ao Paulo por escavar nos pontos que mais me custam falar, mas que bem preciso!

Obrigado à Joana, por ser a minha companhia constante de biblioteca, de lanches de panquecas (e de frustração). Acrescento também os meus demais companheiros de dissertação: Sara, Miguel, Marcelo, Freitas, Vasco e Marco. Nestes momentos, companheiros em suor e lágrimas são tudo. Agradeço também ao Tenente e à Clara que, apesar de não terem sido companheiros diretos, foram companheiros na mesma!

Obrigado ao Dr. Hugo Guimarães, se alguém me ouviu a lamentar sobre os meus problemas, foi esta incrível pessoa! Consigo dizer, com toda a certeza, que sem ele, não conseguiria ter concluído esta dissertação, e, muito menos, ter uma boa perspetiva do futuro que me aguarda!

Obrigado aos meus pais, que durante todo o processo académico compreenderam a minha posição, não me julgaram, e ofereceram toda a liberdade do mundo para fazer as escolhas que eu quisesse, como eu quisesse. Não o digo vezes suficientes: “Obrigado, pais!”. Obrigado aos meus irmãos: ao Kiko por normalizar todo o processo da dissertação e fazer sentir-me um bocadinho menos mal pelos momentos de falta de motivação; e ao JJ pelos momentos aleatórios de *freakness* que, parecendo que não, ajudam a abstrair uma pessoa!

Obrigado aos meus companheiros do Porto: Jorginho, Filipa e Isabel… e a Mica. Confesso que tenho sido um peso nestes últimos momentos, mas nunca me fizeram sentir mal por isso, e a Isabel providenciou momentos de abstração bem preciosos depois de dias de trabalho infindável!

Carlos Pinto

"Compare yourself to who you were yesterday,
not to who someone else is today."

Jordan B. Peterson

Contents

List of Figures

List of Tables

Symbols and Abbreviations

AIN	Artificial Immune Network
AIS	Artificial Immune System
ANN	Artificial Neural Network
ARB	Artificial Recognition Ball
AR	Auto-regressive Prediction
AE	Autoencoder
ARIMA	Autoregressive Integrated Moving Average
AUC	Area Under the Curve
CSA	Clonal Selection Algorithm
CDCA	Cursory Dendritic Cell Algorithm
CPS	Cyber-Physical System
DIA	Danger Inspired Algorithm
DC	Dendritic Cell
DCA	Dendritic Cell Algorithm
DFA	Deterministic Finite Automata
DFT	Discrete Fourrier Transform
DWT	Discrete Wavelet Transformation
DTW	Dynamic Time Warping
FAR	False Alarm Rate
FPR	False Positive Rate
FFT	Fast Fourier Transform
GEP	Gene Expression Programming
GA	Genetic Algorithm
HMM	Hidden Markov Model

I-IoT	Industrial Internet of Things
IoT	Internet of Things
MCC	Matthews Correlation Coefficient
MCAV	Mature Context Antigen Value
MAR	Missed Alarm Rate
MA	Moving Average
MAS	Multi-Agent System
NSA	Negative Selection Algorithm
PAMP	Pathogen-Associated Molecular Pattern
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
SOM	Self-Organizing Map
SOA	Service-Oriented Architecture
SKAB	Skoltech Anomaly Benchmark
SVM	Support Vector Machine
SAX	Symbolic Aggregate Approximation
TLRA	Toll-Like Receptor Algorithm
TPR	True Positive Rate
VAE	Variational Autoencoder
dDCA	deterministic Dendritic Cell Algorithm
iDC	immature Dendritic Cell
kNN	k-Nearest Neighbors
mDC	mature Dendritic Cell
smDC	semi-mature Dendritic Cell

Resumo

Chapter 1

Introduction

1.1 Context

At its core, Cyber-Physical System (CPS) are an integration of computation with physical processes, aligned with communication networks [7]. While traditional embedded systems incorporate computation with real-world interactions, CPS have a particular focus on inter-device communication [7]. They are often integrated into feedback loops where the cyber component (control, communication, and computation in general) and physical components (the physical process itself, measurements by sensors, impacts of actuators) are tightly coupled, just as with embedded systems, but with the added component of subsystem interactions, due to the ability of these subsystems to communicate. This adds complexity and dynamics to such systems that are often difficult to handle in centralized and predefined approaches. As time goes on, CPS tend to have more and more devices connected in a network, more access to the Internet, and less human interaction [8] as CPS are commonly seen as a core ingredient in the concept of Internet of Things (IoT) and the so-called 4th Industrial Revolution [9].

Industry 4.0 is a very attractive concept that was introduced in Germany [9] where the fourth industrial revolution is identified as the incorporation of the Internet and adaptable objects ("smart" [9] machines and products) with the already digitized industry. CPS with access to the Internet are one of the key enablers of the fourth industrial revolution, and this combination is often referred to as the Industrial Internet of Things (I-IoT). The application of IoT in Industry 4.0 allows for devices to be flexible in the way they interface and behave and allows the integration of significantly more complex architectures in the industry, including components that process enormous amounts of data (Big Data), and distributed computation as with cloud computing and edge computing, where lower level devices (i.e. devices closer to the physical process) have increasing processing power and can contribute with more complex computations. This increase in processing power allows CPS components in Industry 4.0 to be independent, such that components organize themselves in a network based on the context that they are inserted in, can monitor their environment, assess potential problems, and be able to adapt, in a way that the system as a whole becomes decentralized [9].

1.2 Motivation

CPS are applied in many different contexts, including, but not limited to, smart grids, water plants, chemical plants, oil, and natural gas distribution systems, transportation systems, medical devices and systems, manufacturing plants, avionics, and automotive control. In many of these contexts, the CPS application is deemed to be safety-critical, in which a change in the standard behavior can lead to drastic economic losses, material damage, if not human lives. Given that attacks have been observed in the past, such as the Stuxnet worm [10], the Ukraine SCADA Attack [11], the Mirai attack [12], Maroochy Water Services in Australia (2000), German Steel Mill Thyssenkrupp (2014), and Norsk Hydro Aluminium (2019) [13], it's important to design these systems to be robust, especially when under faults or attacks.

As the connectivity of CPS increases, and with the I-IoT allowing access to the Internet, so do the attack vectors of such a system. Furthermore, the increase in automation and system dynamics due to complexity also contribute to attacks and naturally occurring faults to be harder to detect, diagnose and repair by a human operator. This leads to a necessity for CPS to detect such attacks or faults and take appropriate measures. Additionally, CPS have real-time requirements, given that a process has to be sensed and processed to prepare any sort of response, and are typically resource-constrained [14].

The field of AIS provides solutions based on immunology and exhibits promising properties, namely adaptivity, scalability, and lightweight solutions. However, it is still largely overlooked by the engineering community. Thus, it is important to assess its applicability for solving the issues stated in the previous paragraph.

1.3 Research Questions and Objectives

Given the presented motivation and context, the following research questions were formulated:

- RQ1: What are CPS and what's their role in industrial environments?
- RQ2: What kinds of Anomaly Detection techniques currently exist in industrial environments?
- RQ3: Which anomaly detection and recovery techniques are more appropriate for CPS in industrial environments?
- RQ4: How can these techniques be tested and validated?

Due to the significant role of CPS in Industry 4.0, and the attractiveness of the latter, this thesis aims to develop an approach to implement anomaly detection as a tool to detect faults. The anomaly detection will focus on the underlying physical process (the plant), as it is often overlooked by existing literature [15]. With the development of the thesis these objectives were further narrowed down to the field of Artificial Immune System (AIS) - a field of algorithms that are inspired by the human immune system - as a largely unexplored field with analogies that fit the problem of anomaly detection, with potential for lightweight, yet effective solutions.

In this context, this dissertation has two main goals:

- Develop a real time process-based anomaly detection technique with CPS resource and time constraints in mind.
- Validate the developed technique using data that is representative of a real world scenario of CPS operation.

1.4 Document Structure

Chapter 2 presents a brief systematic study of literature that resulted in concepts regarding the problem of anomaly detection, and the environment of Industry 4.0, which are given in the same chapter. This chapter further contains state of the art and related work and concludes with a heuristic in literature distribution regarding the problem of anomaly detection.

Chapter 3 elaborates on early work conducted in the field of AIS to elect an algorithm to expand.

Chapter 4 presents the main proposal for this thesis, which consists of expanding on the current algorithm of DCA, which is described in the previous chapter.

Chapter 5 consists of the implementation of the proposed solution on a dataset that resulted from an existing testbed. Results are analyzed considering anomaly detection in the context of Industry 4.0, with conclusions regarding the proposed solution's limitations and strong points.

Chapter 6 concludes the dissertation by summarizing the key contributions provided, summarizing the work done, and providing aspects regarding future work.

Chapter 2

Literature Review

In this chapter, an explanation of how state-of-the-art research was conducted is presented as well as key topics that were identified, followed by the results of said research. Furthermore, the problem characterization and problem definition are also given.

2.1 Systematic Review Approach

Initially, a systematic mapping study was attempted according to the guidelines provided by [16]. Where, in a first step, the key topics to be researched were identified, as research questions:

- RQ1: What are CPS and what's their role in industrial environments?
- RQ2: What kinds of Anomaly Detection techniques currently exist in industrial environments?
- RQ3: Which anomaly detection and recovery techniques are more appropriate for CPS in industrial environments?
- RQ4: How can these techniques be tested and validated?

Given these research questions, the initial approach was to subdivide the research into three categories: Anomaly Detection, CPS, Anomaly Detection in the context of CPS. Both Anomaly detection and CPS are very complex and broad topics, resulting in high numbers of entries, which, given the time frame to conduct state-of-the-art, made a full systematic mapping too time-consuming.

Although a systematic mapping study was not completed, some of the steps of such a study were applied in the last research category: Anomaly Detection in the context of CPS. After a review of articles, by choosing relevant temporary keywords, multiple sources in the literature were read to identify which keywords were more appropriate. These keywords were, then, applied to multiple peer-reviewed databases, namely: IEEE Xplore, ACM Digital Library, Engineering Village (Compendex + Inspec), and Scopus, as shown in Table 2.1. Entries that were dated 2014 or older, were not in English, or did not provide full-text under the licenses provided by the Faculty

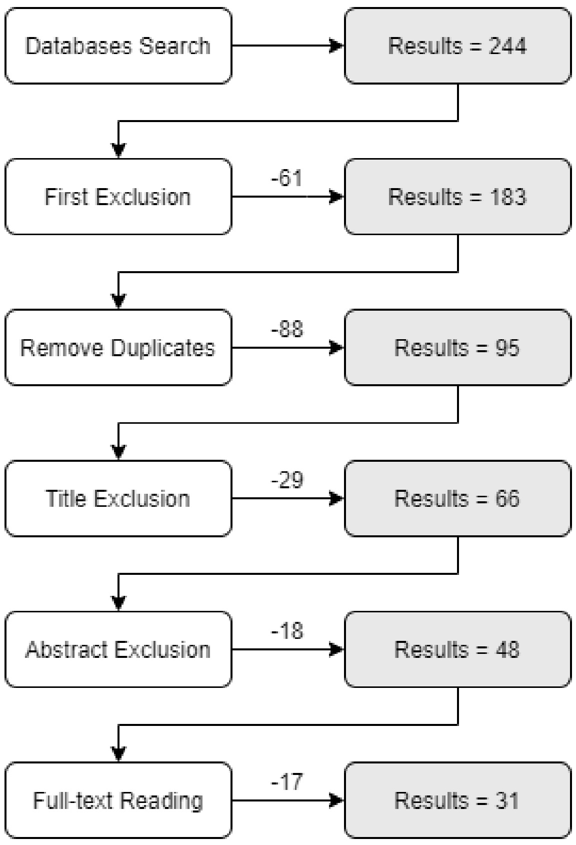


Figure 2.1: Research results, per step.

of Engineering, University of Porto's b-on license, were excluded in what shall be called "first exclusion". The initial results are shown in Table 2.2 as well as results after the first exclusion. All articles were then grouped, and duplicate entries, resulting from the same entry being present in multiple databases, were removed. Then, all entry titles were conservatively evaluated in respect to their relevance to the research questions, such that only entries with clearly irrelevant titles were removed. The same procedure was done to each entry's abstract. The criteria for including articles based on title and abstract were based on the following criteria:

- Offers either insight into requirements, a framework, a survey, a review, or a specific solution with implementation for anomaly detection.
- Is inserted in an industrial context.

The process is summarized in Figure 2.1, along with each step's results. The full (unclassified) literature results are listed in Appendix A.

As previously mentioned, other sources were used, as the systematic mapping study was not successfully applied to all categories. Namely, it would be necessary to apply the same rationale to CPS in industrial environments, as well as Anomaly Detection independently of industrial contexts, to assess which applications could be used in such a context. Furthermore, this methodology was used as source selection, but no actual mapping of studies was done.

Table 2.1: Preliminary systematic study keywords used for each database.

Database	**Search Input**
IEEE Xplore	((("Industrial Environment" OR "Industrial Internet of Things" OR "IIoT" OR "I-IoT" OR "Manufacturing Systems" OR "Industrial Wireless Sensor Network" OR "IWSN" OR "Industry 4.0") AND ("Cyber Physical System" OR "Cyber-Physical-System" OR "Cyber-Physical System" OR "CPS" OR "Distributed Control System")) OR ("Cyber Physical Production System" OR "CPPS")) AND ("Anomaly Detection" OR "Outlier Detection" OR "Anomaly-Detection" OR "Outlier-Detection" OR "Self-Healing" OR "Self Healing" OR "Recovery")
Engineering Village (Inspec + Compendex)	((((("Industrial Environment" OR "Industrial Internet of Things" OR "IIoT" OR "I-IoT" OR "Manufacturing Systems" OR "Industrial Wireless Sensor Network" OR "IWSN" OR "Industry 4.0") AND ("Cyber Physical System" OR "Cyber-Physical-System" OR "Cyber-Physical System" OR "CPS" OR "Distributed Control System")) OR ("Cyber Physical Production System" OR "CPPS")) AND ("Anomaly Detection" OR "Outlier Detection" OR "Anomaly-Detection" OR "Outlier-Detection" OR "Self-Healing" OR "Self Healing" OR "Recovery")) WN ALL)
Scopus	((("Industrial Environment" OR "Industrial Internet of Things" OR "IIoT" OR "I-IoT" OR "Manufacturing Systems" OR "Industrial Wireless Sensor Network" OR "IWSN" OR "Industry 4.0") AND ("Cyber Physical System" OR "Cyber-Physical-System" OR "Cyber-Physical System" OR "CPS" OR "Distributed Control System")) OR ("Cyber Physical Production System" OR "CPPS")) AND ("Anomaly Detection" OR "Outlier Detection" OR "Anomaly-Detection" OR "Outlier-Detection" OR "Self-Healing" OR "Self Healing" OR "Recovery")
ACM Digital Library	((("Industrial Environment" OR "Industrial Internet of Things" OR "IIoT" OR "I-IoT" OR "Manufacturing Systems" OR "Industrial Wireless Sensor Network" OR "IWSN" OR "Industry 4.0") AND ("Cyber Physical System" OR "Cyber-Physical-System" OR "Cyber-Physical System" OR "CPS" OR "Distributed Control System")) OR ("Cyber Physical Production System" OR "CPPS")) AND ("Anomaly Detection" OR "Outlier Detection" OR "Anomaly-Detection" OR "Outlier-Detection" OR "Self-Healing" OR "Self Healing" OR "Recovery")

Table 2.2: Preliminary systematic study results, per database used. Only the unfiltered results using the specified keywords and results after first exclusion are shown.

Database	**Initial results**	**First Exclusion**
IEEE Xplore	32	20
Engineering Village (Inspec + Compendex)	27	24
Scopus	120	79
ACM Digital Library	65	60

2.2 Background

This chapter includes various topics that are relevant for both the developed work throughout this dissertation, as well as for the upcoming section on state of the art (Section 2.3)

2.2.1 CPS and Industry 4.0

CPS are applied in many different contexts, including, but not limited to, smart cities, smart grids, water plants, chemical plants, oil and natural gas distribution systems, transportation systems, medical devices and systems, manufacturing plants, avionics, and automotive control. The ongoing trend is for these applications to be as robust, efficient, and autonomous as possible [17].

Industrial Internet of Things (I-IoT), as IoT applied to industrial manufacturing, is one of the key concepts in Industry 4.0, as sensors, actuators, controllers, etc. (i.e. the "things") are all loose and dynamically connected. The key differences in CPS in the context of Industry 4.0 stem from their Internet access and need for autonomy in the form of the self-x philosophy [18] (e.g., self-healing, self-organizing, self-monitoring and, more generally, self-adapting). In this context, according to [19], CPS desirably possess the following properties:

- Interoperability: all components can exchange information regarding their context and knowledge.
- Virtualization: the physical process is digitized, such that a virtual representation can be used to simulate the said process, to optimize production.
- Modularity: all components are independent enough, that they can be reconfigured in real-time without the need for redesign while keeping their functionalities.
- Distributed Control: components execute independent control, favoring decentralized architectures over centralized ones.
- Flexibility: components can adapt to new contexts.
- Personalization: the system as a whole can respond to human feedback.
- Real-Time Response: the system can respond to contextual changes, as well as changing needs at any point during the production.
- Self-Maintenance: components can respond to attacks or faults by maintaining performance in adverse conditions, without human intervention.

The implementation of CPS in a large-scale environment, with heavily networked components and immense amounts of data, is only possible through certain technologies:

- Communication: many key features of CPS in industrial environments require inter-component communication. This is only possible if components can exchange information between each other respecting timeliness and integrity. A high number of components communicating between each other lead to the need for high-speed connectivity, but also lead to problems such as message collision and incompatible communication protocols [20]. Key technologies that allow this communication to occur include 5G/6G, WiFi, M2M communication and software-defined networking [17] that allow internet access and inter-component communication.

- Processing: with a higher number of components and higher interaction dynamics, processing large amounts of information becomes a concern in Industry 4.0. Data mining and Big data [19] contribute significantly in decision-making based on the immense amounts of data that exist, making use of artificial intelligence, namely, machine learning. On the other hand, the sheer amount of processing power required is provided, not only by the increase of computer performance, but in the distributed philosophy of fog and edge computing [19], where individual components process information that a single, resource-constrained central system could not handle by itself, or make use cloud computing, migrating processing to more capable computers. Concerning virtualization, virtual/augmented reality and digital twins are examples of enabling technologies [19].

These technologies can be incorporated in a convenient architecture that allows for scalability, interoperability while simultaneously accounting for the independent nature of CPS components. P. Zhou et al. [21] distinguish between three categories of architectures: Service-Oriented Architecture (SOA), Multi-Agent System (MAS), and other. SOA defines subsystems as service providers that communicate what they can provide, while some sort of entity is responsible for managing these subsystems based on their provided services; in this perspective, each subsystem is not independent which is the main difference from MAS, where each agent acts on its own based on the context where they are inserted, favoring a more decentralized approach over SOA. The "other" category, that the authors describe as other "aspect-oriented architectures", are mainly concerned with the problem of repurposing or incorporating legacy systems, as developing new CPS that follow the previous architectures over reusing existing devices and protocols can be costly. More traditional architectures make use of centralized architectures, but these are less dynamic and adaptable by nature.

2.2.2 Anomaly Detection

Anomaly detection or outlier detection refers to algorithms that analyze data to identify anomalies or outliers. An anomaly detection algorithm can be based around prior knowledge or by using normal operating conditions to train the algorithm; in either case, a profile of what is to be considered normal is built. The algorithm then receives input data and evaluates if it matches its normal behavior, or if it constitutes an outlier. Anomaly detection has a vast application range, as long as data can be correlated, anomaly detection is possible, as a simple example: a person's IQ test results usually do not drastically change in between IQ tests, and, if it does, an anomaly can be identified.

For this dissertation, the most relevant anomaly detection is time series data anomaly detection, where, as the name implies, the input data is a time series, usually through the use of timestamps in each data instance. This type of anomaly detection is particularly important when discussing anomaly detection in the context of networks or physical processes in real-time, leading to characteristic challenges that are not shared with anomaly detection in data that is not time-dependent. Time series anomaly detection is further discussed in section 2.2.2.4. K. Mehrotra et al. [6] offer a comprehensive introduction into anomaly detection, along with existing algorithms, including the category of anomaly detection in time series. This will be used as a reference for the remainder of this section, regarding anomaly detection, unless otherwise specified.

2.2.2.1 Anomaly Detection Principles

Different algorithms can vary greatly in how they analyze data and draw conclusions, however, there is a set of principles common to all algorithms which will be explained in the following subsections.

Independently of how an algorithm works, it needs to identify data as either anomalous or normal. Anomalies can be manifested in three ways [1]:

- Point anomaly: this is the simplest case in which the individual data instance is anomalous when compared with the rest of the data. For instance, if a data instance presents values that are higher than all other data instances.

- Contextual anomaly: data instance is anomalous regarding its context. The definition of context is problem-specific. For instance, a recorded average yearly temperature of 9ºC in the Amazon rain forest can be seen as a contextual anomaly, because it is abnormal in the context that it is recorded (here, the context is longitude and latitude), but such an average yearly temperature wouldn't be anomalous in all contexts (e.g., Europe).

- Collective anomaly: a related collection of data is anomalous in respect to the entire dataset. As an example, if a periodic behavior that repeats for every constant time interval suddenly changes for a single period, that period alone is not anomalous, nor is any point in the context deviating from the sequence, but the entire period behaves inconsistently when compared to surrounding data points, despite possibly not having any significantly high values or high rate of change.

Figure 2.2 illustrates these three types of anomalies in a time series.

Regarding a particular algorithm, four different cases of data classification can be defined:

- True Positive (TP): data was correctly classified as anomalous.

- True Negative (TN): data was correctly classified as normal.

- False Positive (FP): data was classified as anomalous, but the data is normal.

- False Negative (FN): data was classified as normal, but the data is anomalous.

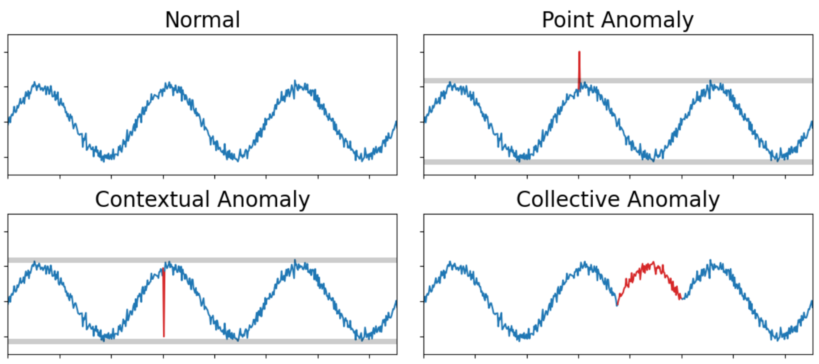


Figure 2.2: Three types of anomalies [1], applied to a noisy sine wave. Anomalies are highlighted in red.

2.2.2.2 Anomaly Detection Metrics

In order to compare different algorithms, their performance can be calculated through evaluation metrics. The commonly used accuracy metric serves anomaly detection evaluation poorly, as, given that most data points constitute normal occurrences, an algorithm that identifies all points as normal will have reasonable accuracy. Three common metrics to evaluate an imbalanced dataset algorithm's performance are precision, recall, and Rank-Power. Additionally, [22] further mentions the use of F_1-score (more broadly, F_β-score). The ROC curve [23] is also used frequently. Less common metrics include balanced accuracy (e.g. used by P. Li et al. [24]) and the Matthews correlation coefficient (e.g. used by M. Pamukov et al. [25]).

- Precision accounts for the ratio of correctly identified outliers to total outliers detected (Equation 2.1). If all outliers that were detected are, indeed, outliers, then the algorithm's precision is 1, even if some outliers were deemed as normal.

$$Precision = \frac{TP}{TP+FP} \tag{2.1}$$

- Recall measures the ratio of correctly identified outliers in respect to all outliers in a given dataset (Equation 2.2). If all outliers that exist in the dataset are identified by the algorithm, then the algorithm's recall is 1, even if some normal occurrences were deemed anomalous. Sometimes it is also called sensitivity.

$$Recall = \frac{TP}{TP+FN} \tag{2.2}$$

- Rank-Power incorporates the notion of ranking anomalies according to their degree of suspicion (Equation 2.3), naturally, this is only possible if the algorithm can rank anomalies accordingly, which is usually possible as most algorithms use some notion of distance or difference to decide if data is anomalous or not. If all correctly identified outliers have the highest degree of suspicion, then Rank-Power will be 1.

$$Rank-Power = \frac{TP(TP+1)}{2\sum_{i=1}^{TP} R_i} \tag{2.3}$$

Where R_i is the rank attributed to the ith correctly identified outlier.

- The F_β-score (Equation 2.4) offers a combined metric of precision and recall, as β increases, precision is favored, while lower values favor recall. The F_1-score (Equation 2.5) gives a harmonic mean between recall and precision, being closer to the smallest of those two values, but favoring none, in particular, being used often to compare between algorithms.

$$F_\beta = \frac{(\beta^2+1) * Precision * Recall}{Recall + \beta^2 * Precision} \tag{2.4}$$

$$F_1 = 2\frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

- Balanced accuracy (ACC) measures the mean value between recall and specificity. Specificity is the opposite of recall, which was described in Equation 2.2, in that it measures the ratio of correctly identified normal occurrences in respect to all existing normal occurrences in a given dataset. Balanced accuracy's calculation is specified in Equation 2.6, where the first term in the numerator is recall.

$$ACC = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2} \tag{2.6}$$

- The Matthews Correlation Coefficient (MCC) measures the correlation between the true values and the predicted values (Equation 2.7). Its values vary between -1 and 1, where 0 means that the algorithm's performance is similar to that of a random classifier, -1 is a completely wrong classification, and 1 means a perfect classification.

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{2.7}$$

- The ROC curve, similarly to Rank-Power, requires a given algorithm to be able to rank its results according to their degree of suspicion. However, unlike Rank-Power, the ROC curve does not result in a single, absolute number, but rather a curve, as the name implies. This curve is a plot of the False Positive Rate (FPR) in respect to the True Positive Rate (TPR), i.e., recall. The FPR can be calculated according to Equation 2.8, while TPR was already shown in Equation 2.2. In algorithms where the outcome is based on continuous numerical outputs, being further classified into a binary result through the use of thresholds, a ROC curve can be constructed by varying said threshold (theoretically from -∞ to +∞) and plotting

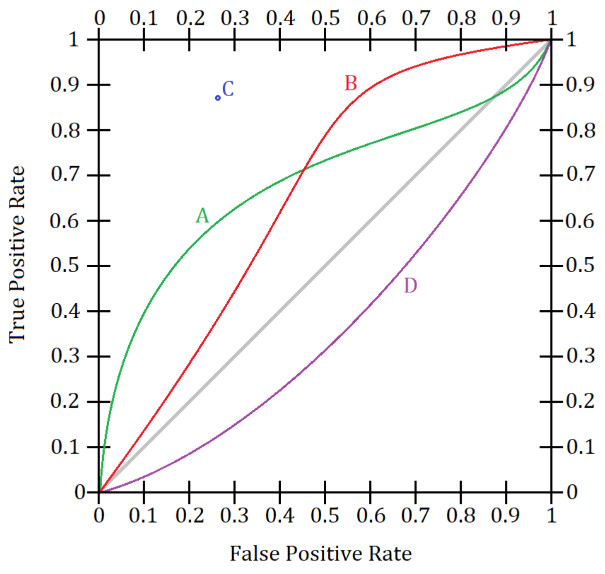


Figure 2.3: Mapping example of 4 different hypothetical algorithms in ROC space.

FPR and TPR. ROC curves are indeed a sum of step functions as varying thresholds change the classification of a given data point discretely, but as the number of data points increases, the step function approaches a continuous curve. For algorithms where a ROC curve isn't possible, the algorithm's results can still be plotted as a single point, with fixed FPR and TPR. However, if the algorithm can be plotted in a curve, the Area Under the Curve (AUC) of the ROC curve can be used as a numerical reference.

As an example for ROC, in Figure 2.3, four hypothetical algorithms are presented. Algorithms A, B, and D have a varying threshold that changes from classifying all data points as negative (bottom-left corner) to classifying all data points as positive (upper-right corner). Algorithm C is mapped as a single point, corresponding to a specific implementation of an algorithm that already classifies points without the use of thresholds. Furthermore, the gray diagonal, where TPR has the same value as FPR, represents an algorithm reference that randomly classifies any data point and its AUC is 0.5. The ideal algorithm lies in the upper-left corner of ROC space and has an AUC of 1, corresponding to an algorithm that never misclassifies any data point. Algorithms that lie below the gray diagonal (such as algorithm D, with an AUC below 0.5) perform poorly, because a random-classifying algorithm would perform better. Algorithm C outperforms both algorithms A and B. Neither algorithm A and B is better than the other (their AUC is similar): for applications where false positives are costly, algorithm A is better, while for applications where positive cases must be identified, algorithm B is better.

$$\text{FPR} = \frac{FP}{FP+TN} \tag{2.8}$$

As a quick example, for a given dataset consisting of 100 data points, where 10 of them are anomalous, an anomaly detection algorithm is applied and detects 20 outliers, out of which, 8 are truly anomalous (12 of the outliers detected are normal data points). The 8 correctly identified outliers occupy positions 1 through 7 and 20, i.e., 7 of the outliers that were correctly detected were considered to be the most suspicious, whereas one of the outliers that were correctly identified was considered to be less suspicious than the remaining 19 outliers detected. In this scenario, the previously mentioned metrics (except the ROC curve) would be given by:

$$TP = 8 \quad FP = 12 \quad TN = 78 \quad FN = 2$$

$$Precision = \frac{8}{8+12} = 0.4$$

$$Recall = \frac{8}{8+2} = 0.8$$

$$Rank-Power = \frac{8*(8+1)}{2*(1+2+3+4+5+6+7+20)} = 0.75$$

$$F_1 = 2\frac{0.4*0.8}{0.4+0.8} = 0.53$$

$$F_2 = \frac{(2^2+1)+0.4*0.8}{0.8+2^2*0.4} = 0.44$$

$$F_{0.5} = \frac{(0.5^2+1)+0.4*0.8}{0.8+0.5^2*0.4} = 0.66$$

$$ACC = \frac{\frac{8}{8+2}+\frac{78}{78+12}}{2} = \frac{0.8+0.86}{2} = 0.83$$

$$MCC = \frac{(8*78)-(12*2)}{\sqrt{(8+12)(8+2)(78+12)(78+2)}} = 0.5$$

In the provided example, precision is low because a significant number of normal data points were misclassified as outliers, recall is higher because most actual outliers were identified, and Rank-Power is not ideal because one of the correctly identified outliers had a low degree of suspicion (the one that was placed in the twentieth position), F_1 is highly impacted by the low precision; that impact increases in F_2, while $F_{0.5}$, which favors recall, isn't impacted as much. Balanced accuracy is just the average between recall and specificity (specificity would be 0.86, in this example) and MCC is 0.5 which indicates this algorithm does significantly better than a random classifier but can be improved substantially. All of these metrics have their limitations, namely: if an algorithm identified a single outlier in total, but it is correctly identified, precision and Rank-Power will be 1, even if an enormous amount of actual outliers were unidentified; if an algorithm determines all data points to be anomalous, the recall will be 1, despite the algorithm being useless for any application. Thus, no single metric provides a good evaluation of performance, and multiple metrics have to be used.

As previously mentioned, anomaly detection attempts to identify data points that are anomalous when compared to what's "normal". A first problem is to define what is normal and anomalous

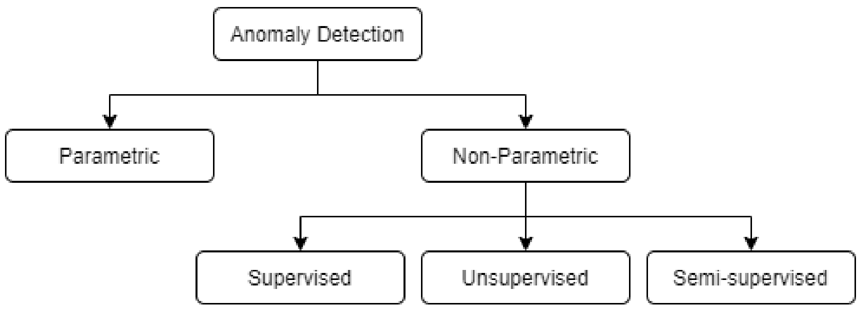


Figure 2.4: Simplified classification of anomaly detection approaches, adapted from [2].

in an objective and computable manner. This definition varies depending on the approach taken by a specific algorithm and will be further discussed in the next section (Section 2.2.2.3).

2.2.2.3 Anomaly Detection Approaches

Three main approaches to what constitutes an anomaly can be identified:

- Distance-based: data points that are distant from others are outliers.
- Density-based: data points that are located in lower density regions are outliers.
- Rank-based: data points whose nearest neighbors have other points as nearest neighbors are outliers.

Furthermore, the classification used by [2], which was derived from previous literature, can be used, where two main groupings of anomaly detection algorithms are distinguished: parametric and non-parametric. Non-parametric algorithms are further separated by [2] as well as [6] into three groups: supervised, semi-supervised and unsupervised (as illustrated in Figure 2.4).

In parametric anomaly detection data has a known distribution and parameters (e.g., normal distribution characterized by mean and standard deviation). In non-parametric approaches, prior knowledge of data may be known, but its distribution is not and, thus, we need to learn it. This can be accomplished through a *supervised* algorithm where all data points used have a known classification through the use of labels (e.g., "anomalous" and "normal"), through an *unsupervised* algorithm where data points have no labeling and data patterns and relationships are learned, or through a *semi-supervised* algorithm using a small amount of labeled data and using that knowledge to attempt learning on further, unlabeled data, where the latter constitutes the majority of all data, which can be accomplished by applying what was learned in a supervised setting to label the unlabeled data, for example.

Examples of anomaly detection approaches are summarized in Tables 2.3, 2.4 and 2.5 for distance-based, clustering-based and model-based approaches, respectively.

One final concept to introduce is the application context in which anomaly detection occurs: whether it is online or offline. Offline anomaly detection refers to the analysis of data that is complete and obtained in an arbitrary moment in the past, which tends to have loose constraints

Table 2.3: Different approach examples of distance-based criteria to identify outliers. Information source: [6]

Approach	Description
Distance to All Points	Computes sum of distance to all other points.
Distance to Nearest Neighbor	Computes distance to closest point.
Average Distance to k Nearest Neighbor	Computes average distance to k nearest points.
Median Distance to k Nearest Neighbor	Computes median distance to k nearest points.

Table 2.4: Different approach examples of clustering and of clustering-based criteria to identify outliers. Information source: [6]

Goal	Approach	Description
Identify different clusters	k Nearest Neighbors	Labels a given point according to the majority of k nearest points' labels.
	k-Means Clustering	Labels a given point according to which cluster centroid cluster it is closest to, updating the cluster centroid that it gets associated with.
	Fuzzy Clustering	Labels a given point according to which cluster centroid cluster it is closest to, updating the cluster centroid that it gets associated with, having less impact the farthest it is from the centroid.
	Agglomerative Clustering	Starts with a high number of small clusters. Clusters merge as more points are processed based on algorithm-dependent criteria.
	Density-Based Agglomerative Clustering	Clusters are defined based on a minimum of points within a specified distance (representative of density), labeling data accordingly.
	Divisive Clustering	Clusters are successively partitioned into smaller clusters until algorithm-dependent criteria is verified.
Detect anomalies through identified clusters	Cluster Membership or Size	Identified clusters containing a small number of data points, or data points that have very low membership values to any cluster are considered outliers.
	Proximity to Other Points	If a data point is significantly distant to other points in the same cluster, is is considered an outlier.
	Proximity to Nearest Neighbor	If a data point is significantly distant to its closest neighbor in the same cluster, is is considered an outlier.
	Boundary Distance	If a data point's distance to its closest cluster boundary, it is considered an outlier.
	When Cluster Sizes Differ	If a data point's normalized distance to its closest cluster boundary, based on cluster size, it is considered an outlier.
	Distances from Multiple Points	Not a method on its own and can be applied to any of the above. E.g., choose multiple neighbors, choose multiple cluster boundaries, choose multiple cluster centroids as reference.

Table 2.5: Different approach examples of model-based criteria to identify outliers, distributions and models for data that depends on time. Information source: [6].

Goal	Approach	Description
Given a model, detect anomalies	Model Parameter Space	If the inclusion of a data point changes some learned model's parameters (.e.g., coefficients of a linear model) by a significant amount, it is considered anomalous.
	Data Space	If a data point diverges from a concrete model (e.g., an equality) it is considered an outlier.
Distributions	Parametric Distribution Estimation	Construct a model based on data metrics (e.g., Gaussian using mean and standard deviation). Points that don't have high likelihood values according to the distribution are considered outliers.
	Regression Models	Construct a model based on regression (e.g., linear regression, artificial neural networks, kernel regression - often used in Support Vector Machines, and splines), usually using gradient descent in respect to error.
Model a Time-Varying Process	Markov Models	Computes probability of outcomes based on past occurrences. Data is modeled as a sequence of events
	Time Series Models	If time spacing between events has significance, time series models can be used (e.g., ARIMA, DFT, Haar Wavelet Transform) as a pre-processing step to numerically model time as a feature.

on used algorithms as they can be more complex and do not need to offer fast response times. On the other hand, online anomaly detection is either real time or near real time and is usually associated with the streaming of data resulting in a continuous process of data acquisition and anomaly detection. A problem with online anomaly detection is that we cannot define what normal behavior is for the entire data, because data is never assumed to be complete; if an algorithm learns the typical behavior of a system at a given time, later on, the system can change behavior which prompts the algorithm to learn this new behavior, which is usually a computationally intensive procedure that cannot be repeated arbitrarily. The problem of online anomaly detection is much pertinent to time series anomaly detection.

2.2.2.4 Anomaly Detection in Time series

In time series data, data order and the spacing in time between data points have a significant impact, as opposed to non-time series where data points can be processed in an arbitrary order without loss of information (although results may change, depending on used techniques). Due to this, time can not be looked at as just another variable to analyze, because of its unique characteristic as a dynamic and contextual indicator. As a quick example of this last statement: a temperature sensor may normally measure 50ºC and 20ºC for a given process at different moments in time, however, if this same sensor measures these two values alternately in a very small time window, then it is behaving in an anomalous way.

Anomaly detection can be applied in a single time series or between time series. In the former, we're interested in detecting anomalies occurring in a time series that differ from the rest of that same time series, while in the latter, we're interested in detecting anomalies regarding a time series as a whole, when compared to other related time series. It is worth noting that the problem of detecting anomalies in a single time series often turns into the problem of detecting anomalies between time series when approaches that subdivide a single time series into subsequences are used (such as sliding window approaches).

2.2.2.5 Time Series Models

In order to use time in approaches previously summarized in section 2.2.2.3, time may be modeled in a way that time series data's dynamics can be numerically used as a common feature. In the interest of reducing the dimension of data, various models aim at capturing time series' specific information, such as long-term trends, periodicity, seasonality, and frequency domain behavior, at the cost of losing information regarding small changes that occur in a small number of points. Some time series models will be discussed, namely 1. Autoregressive Integrated Moving Average (ARIMA) models, 2. Discrete Fourier Transform, and 3. Haas Wavelet Transform.

ARIMA models capture the dependence of a variable based on its past values, using recent values more prominently than older ones, which allows it to model changes in behavior, successfully modeling non-stationary data. Furthermore, it uses a moving average which eliminates noise, to an extent, but includes a term to represent random noise, and a term to represent drift over time. Because ARIMA models are a combination of terms that model different aspects of data, variations to the model can be introduced according to the expected behavior and structure of data, namely, Vector ARIMA (VARIMA) can be used in vector data, Seasonal ARIMA (SARIMA) is particularly well suited for seasonal data, and Fractional ARIMA (FARIMA) is well suited for data with long-range dependence, by reducing the rate at which the impact of older values decay.

Discrete Fourrier Transform (DFT) is the discrete version of the Fourier Transform for continuous signals, and aims to do the same: capture the time-domain signal as a sum of sinusoids of different frequencies and amplitudes. Because sinusoids with frequencies that match the time-domain signal the best will have higher amplitudes, DFT is usually described as offering a frequency-domain description of signals. This allows to numerically represent the shape of signals as an array of complex coefficients, at the cost of ease of access to information regarding temporal localization (although this information is theoretically preserved).

Discrete Wavelet Transformation (DWT) allows the generation of wavelets based on a discrete time series, capturing information regarding frequency as well as time. Haar wavelets, for instance, are obtained by computing how much a signal matches a single period of a square wave averaged at zero shifted over the entire signal, for multiple frequencies (as well as a constant term with non-zero average value to capture signal offset).

2.2.2.6 Detecting Anomalies in Time Series

In order to detect anomalies, we may have a model-based algorithm, which compares predicted values, which can be obtained from models such as ARIMA, and Support Vector Machine (SVM) and compare them with real measurements directly. We can also use a prior transformation of time series, by aggregating data (such as aggregating points by their average values), discretizing them (such as using symbolic variables to represent a range of values), or by applying transformations (such as DFT or DWT) using these transformed versions of data in approaches previously described in section 2.2.2.3.

Some example approaches to numerically measure the distance between time series are introduced in Table 2.6 along with their strengths and weaknesses. Ensemble methods make use of multiple distances obtained through a weighted combination of different methods to compute a combined anomaly score that allows for the use of two different methods to make up for each other's weaknesses. E.g., Symbolic Aggregate Approximation (SAX) with sliding window is not well suited for detecting anomalies with time lagging, generating false positives often, and can be combined with an approach such as Dynamic Time Warping (DTW), which does detect such anomalies, but is not well suited for anomalies consisting in small deviations; these two approaches combined may complement each other well. On a final note, the computational overheads of these methods are not negligible and depending on the application, this might be prohibitive of their usage.

2.3 State of the Art

In this section, state of the art of CPS and anomaly detection techniques will be shortly introduced, to identify promising or unexplored approaches in anomaly detection.

2.3.1 Anomaly Detection in Industrial CPS

As previously explained in Section 2.2.1, Industry 4.0 comes with its challenges. Applying anomaly detection in such a setting implies identifying critical aspects relating to these challenges as well as the typical anomaly detection in time series problems. These challenges depend on the specific context of industrial application, but generally, an appropriate anomaly detection mechanism must accommodate the following characteristics:

1. *Generalization*: it is difficult to predict what kinds of anomalies will occur, and how they manifest in a time series. As an example, a given time series may be highly periodic and approaches that accommodate for seasonal patterns will do very well in detecting deviations, while that same approach in highly unstable time series will perform poorly. A good anomaly detection mechanism must be consistently effective in a way that is agnostic to the time series normal behavior and needs to assume the intended behavior to be dynamic.

Table 2.6: Different approaches to measure distances in time series, with their respective strengths and weaknesses. This table is a reformatting from a table available at [6].

Time Model	**Strengths**	**Weaknesses**
Cross Euclidean Distance	Easy to implement; Computationally Efficient	Lock-step measure; normal series with time lagging cause problems;
Cross Correlation Coefficient-based measure		
Standard Deviation of Differences		
Dynamic Time Warping	Elastic measure; successfully addresses problems with time lagging between sequences	Small deviations may not be detected
Discrete Fourier Transform	Good in detecting anomalies in frequency domain; normal series with time lagging can be solved	Small deviations may not be detected; cannot detect anomalies with time lagging
Discrete Wavelet Transform	Good in detecting anomalies in frequency domain;	Small deviations may not be detected; sensitive to time lagging
Symbolic Aggregate approXimation (SAX) with Sliding Window	Tolerates noise, as long as its standard deviation is small	May not detect abnormal sequence of shorter length than feature window size; normal series with time lagging can result in false positives
SAX without Sliding Window	Tolerates noise, as long as its standard deviation is small	May not detect abnormal sequence of shorter length than feature window size; normal series with time lagging can result in false positives; small deviations may not be detected;
SAX with Bag-of-Pattern	Tolerates noise, as long as its standard deviation is small; normal series with time lagging can be solved	Cannot detect anomalies with time lagging; cannot detect anomalous series with similar frequencies but different shapes

2. *Time Performance*: in industrial applications, anomaly detection must be detected within useful time as a late detection will potentially lead to enormous losses and compromise safety.

3. *Resource Efficiency*: in the concept of Industry 4.0, CPS components are expected to be self-monitoring. As such, components need to perform anomaly detection in a decentralized manner which leads to concerns regarding resource usage, as these components are often resource-constrained, both in themselves and in the use of network infrastructures (i.e. communication itself must be efficient).

4. *Low Need for Labeled Data*: publicly available datasets for industrial settings undergoing attacks are not abundant. For proper learning, many approaches require immense amounts of data, which is not viable in this context. Furthermore, relying on labeled data could eventually lead to approaches that don't respond well if the system changes its behavior intentionally as the model would be tightly coupled with the data it had learned from, which would be against the characteristic mentioned in point 1.

5. *Robustness and Safety*: components need to be able to respond to adverse conditions, by continuing normal operation, without compromising the safety of the equipment and that of human life.

These challenges are non-trivial and, to the best of this dissertation's author's knowledge, no solution attempts to solve all of these challenges simultaneously, tackling only a portion of them.

2.3.1.1 Anomaly Detection Techniques

In this section, anomaly detection techniques that were acquired from the literature review are presented.

A vast majority of techniques make use of deep learning approaches (e.g., [26], [27], [28]), a more detailed analysis of such techniques was conducted by [29]. Deep learning techniques offer great generality, in that they are highly adaptable, however, they tend to have relatively big computation times, high resource usage, and often require significant amounts of data to train the model. These aspects of deep learning are indicators that it is not well suited for a CPS environment, with real-time constraints, resource limitations, and reduced publicly available data. Despite this, a significant portion of literature still pertains to this area.

AIS is one of the most promising modern solutions to the problem of anomaly detection [30]. These techniques have high generality, are usually computationally efficient and typically require less resource use. [31] and [30] offer insight into existing AIS techniques. These techniques are inspired by the mammalian adaptive immune system and are a subclass of artificial intelligence. Most work has been developed according to the Negative Selection Algorithm (NSA) where randomly generated so-called detectors are compared with data generated by the CPSs normal operation, being killed during its maturation period if they match any of that data. After the maturation period, the detectors will be considered to match only with data that is "non-self", corresponding to data that is corresponding to an outlier. These algorithms are highly dynamic due to the detector's lifetime, which makes each detector locally restricted, in time.

The selected papers according to Section 2.1 that present anomaly detection techniques will be shortly presented in the following paragraphs. These are not exhaustive descriptions of each paper's contributions, and as such, it is recommended that the original paper is read by any interested readers.

F. De Vita et al. [32] use Deep Learning techniques to both extract meaningful features and identify anomalies. In an early data analysis phase, the authors used PCA to determine which features to include in the anomaly detection pipeline, visualizing both normal and anomalous data and visually identifying which features differ the most from one case to another. In the anomaly detection pipeline, time series data goes through an Autoencoder (AE) to reduce data dimensionality, followed by PCA to separate meaningful information, and the result is inputted to a K-Means algorithm which outputs if the input is anomalous or not. The authors use two different time series models: for one of the inputs, a fixed time window is used and its mean, maximum and minimum values are calculated and used as features; for the other input, they compute its Fast Fourier Transform (FFT) and use a given number of frequency components. The authors also use a Deep AE for the FFT input, as its dimensionality is high and the relationship between FFT coefficients is highly non-linear. AEs are unsupervised neural networks that work by first compressing input data into a reduced dimensionality format (encoding) and then decompressing it back to the original data (decoding) and learning based on the difference between output and input, effectively learning how to find a so-called "latent" representation of data without the need for labeling. This latent representation can be seen as an abstract feature extraction method that reduces input dimensionality without losing information. In this use case, the decoder is only useful in the learning phase, after which, only the latent representation (i.e. the encoded input) will be used.

G. Bernieri et al. [33] use Deterministic Finite Automata (DFA) to detect abnormal behavior analyzing packets that use Modbus/TCP and CoAP protocols. DFAs analyze an alphabet sequence and produce a single result for a given sequence. The authors extend altered DFAs which learn normal behavior and automatically build a DFA with expected alphabet sequences which are representative of packets; the finite state machine analyzes these sequences which, if operating normally, will follow along with the expected state transitions that were built during training. The authors further extend this approach to include multi-modal DFA that can select appropriate, lower-level DFAs that correspond to a predicted behavior, which allows for more than one behavior to be monitored; furthermore, they implement timing mechanisms that allow for the detection of DoS attacks, even if these attacks follow along with a normal behavior in terms of packet sequences.

G. Bernieri et al. [34] use Variational Autoencoder (VAE) for anomaly detection. VAEs behave the same way as AEs with the difference that the latent representation is expressed as a Gaussian distribution, and the decoder, instead of using the latent representation directly, samples from the learned distribution, which acts as regularization and avoids overfitting at the cost of generally less accuracy. The authors separate Anomaly Detection into three modules: a module that monitors packets between a Corporate Network and a Control Network; a module that monitors packets inside the Control Network; a module that monitors physical side-channel data from the system. These three modules each have their VAE, and their respective analyses are all inputted into a final "Correlation Node" that, based on all three modules, decides if data is anomalous or not.

H. Shin et al. [35] use SVM to learn typical parameters of queue models (based on Queue Theory) of a shop-floor simulation. SVM is a supervised learning technique that learns to split data linearly using hyperplanes that maximize the distance between either classification. Non-linearly separable data can also be classified through SVM by first converting input features into another space where linear classification is possible, through the usage of kernel functions. Although the authors in [35] do not specifically mention it, it can be assumed they used One-Class SVM as they defined a margin parameter and only used "normal" condition data for training, in which case, the technique is unsupervised.

H. Sandor et al. [36] use Dempster-Shafer's "Theory of Evidence", where a given number of states are specified and, given evidence (i.e. measurements), one can assign a probability of belief which specifies the degree that the measurement can be used to infer a certain hypothesis. It is also possible to combine multiple evidences' contributions to a hypothesis into a joint contribution, which allows for data fusion. In the particular case of anomaly detection, the hypotheses set, named frame of discernment, can be made up of two hypotheses such as "anomaly" or "normal" and, if the evidence that supports the "anomaly" hypothesis does so with a high joint probability (higher than a given threshold), the "anomaly" hypothesis can be deemed correct and be detected. Previous work[37] includes more hypotheses to further specify between "physical anomaly" and "cyber anomaly" along with the standard "normal" hypothesis.

P. Li et al. [24][38][39] use clustering algorithms, after which, the authors developed an algorithm to automatically define geometrical convex hulls that encompass each found cluster. After an initial convex hull is created for each cluster, using the quickhull algorithm by C. Barber et al. [40], the algorithm P. Li et al. developed "digs" each hull to be more restrictive of the cluster data, allowing for hulls to be non-convex. They view this as an improvement over clustering algorithms that assume data structure as it makes no assumptions on the structure of each cluster, e.g. the assumption that clusters follow along ellipsoids, normal distributions, or have some sort of symmetry; and it is also an improvement over neighbor-based clustering as these algorithms have to store every single point in memory to classify new points, while hull-based methods require only the hull itself to be stored. The authors don't mention how the algorithm behaves when hollow datasets are used, e.g. whether a donut-shaped normal behavior cluster would consider points that reside inside the hollow part of the cluster to be normal occurrences.

Q. Li et al. [41] use an ensemble algorithm that combines Moving Average (MA), Autoregressive Prediction (AR), Artificial Neural Network (ANN), Gene Expression Programming (GEP) and SVM. The authors recognize each algorithm's shortcomings and linearly combine their results in a weighted sum such that these weights are dynamically updated depending on which algorithm performs best. A downside to their approach is that a lot of resources are used, and there's a possibility that a complex algorithm that doesn't contribute substantially to classifying anomalies has a very low weight attributed to it, yet it still needs to execute.

M. Saez et al. [42] use expert knowledge to build states that describe CPS operation in multiple levels. They use DTW to automatically estimate event starting and end times regarding machine-part interactions. Using these states, the authors claim that anomaly detection can be narrowed down to a smaller scope, and, therefore, anomalies become easier to detect. Their use case uses adaptive threshold limits, SVM, among others, to detect anomalies, but any anomaly detection method could theoretically be used, and changing anomaly detection methods depending on the current state is possible. The definition of states seems to not be scalable, as complex CPS can lead to a "state explosion". In more recent work, M. Saez et al. [43] address this issue, mentioning the possibility of identifying unreachable states to mitigate this problem. In [43], the authors use both data-driven and physics-based models to validate their approach, using expert knowledge in the latter.

R. Pinto et al. [44] use the deterministic Dendritic Cell Algorithm (dDCA) to detect intrusions through anomaly detection. dDCA is part of a broader group of algorithms known as Artificial Immune Systems (AIS), which are algorithms inspired by immunology (this field is further explained in Chapter 3). By identifying features that are indicative of anomalies and normal occurrences, the algorithm applies a stochastic sliding window to determine if data points are anomalous based on previous occurrences. The algorithm needs to be properly modeled according to the application scenario, and the authors show that using the wrong features can lead to poor performance, akin to that of a random classifier.

D. Ramotsoela et al. [2] conducted a survey on anomaly detection applied to intrusion detection, where they include both parametric and non-parametric solutions. As previously mentioned, parametric anomaly detection is not suitable for dynamic systems where normal behavior is usually not known *a priori* and might change over time, therefore, these solutions will not be summarized here. Regarding non-parametric solutions, the authors mention k-Nearest Neighbors (kNN), SVM, ANN, Genetic Algorithm (GA) and hybrid solutions. The authors point out that practical feasibility is often overlooked, and the majority of research does not consider resource constraints.

A. Bagozi et al. [45] apply anomaly detection in fault detection through cloud computing. The authors perform two steps of clustering: first, they apply a variant of a clustering algorithm that is appropriate for streaming data[46] obtaining synthesized data, which is then, secondly, supplied to an X-means algorithm[47] that does the final clustering. The system designer needs to predefine warning and error boundaries, but the algorithm potentially detects state changes before these boundaries are crossed by computing relevance (distance between cluster centroids, distance between points, and difference in the number of clusters) between clusters that result from different time steps, improving anomaly detection efficacy.

L. Kaupp et al. [48] use AE to gather features from mixed data, i.e. “numerical, categorical and temporal attributes”. Unlike the approach previously mentioned in this paper ([32]), AEs are not used to reduce data dimensionality by using the latent space representation as features, but rather the AE itself is used as an anomaly detection algorithm, by analyzing the error in reconstruction of the input. The rationale is that, as normal data was used as training for the AE, anomalies will lead to bad reconstructions of input, since they differ significantly from training data.

A. von Birgelen et al. [49] use a Self-Organizing Map (SOM) which is an ANN that is usually used for dimensionality reduction by representing an arbitrary number of features as an N-dimensional map (most often two-dimensional for ease of human visualization). The authors use it as an anomaly detection tool by looking at the quantization error, i.e., how far off the data points map to any neuron in the SOM. This approach isn't the paper's contribution, it is the estimation of where the anomaly originated that is the main contribution of the paper- The authors use reverse mapping (from SOM output to the original input, which effectively works as an estimator) to discover which signal or sensor most likely originated the anomaly by, once again, mapping the anomalous input to the SOM and measuring the distance between the signals and the weights of the neuron it was mapped to. The underlying principle is that signals with higher distances are more likely to be the root of anomaly.

M. Caporuscio et al. [50] survey multiple aspects of "Smart Troubleshooting" namely prevention, detection, recovery, and adaptation. For this dissertation, focus will be given to the literature review the authors conducted on detection, which they classified as log analysis or model-based analysis. In log analysis, they name Deep Learning, SVM and clustering as the used methodology for anomaly detection. In model-based analysis, they identified Hidden Markov Model (HMM), Petri Nets, and automaton, regarding the latter, automata learning is emphasized.

2.3.1.2 Literature Distribution

Based on the state of the art that was conducted, key technologies of anomaly detection were identified. For each key technology, a custom query was designed to be applied in multiple databases (the same databases shown in Table 2.2). The results were then recorded and plotted as shown in Figure 2.5. Further results can be found in Appendix B

Note that these results do not replace a full literature review, and serve merely as a heuristic to obtain some insight into how much researchers invest in each technology. Following, are the main identified limitations to the methodology used here:

- duplicate results between databases were allowed;
- duplicate results within the same database were allowed (from different queries);
- no analysis of the results' titles or any sort of content was conducted;
- only technologies that were discussed in Section 2.3.1.1 were considered, therefore it is likely that there are other fields in anomaly detection or fault detection that were not considered;
- the results obtained were not subjected to any filter (e.g. time-window filter to avoid results that were too old).

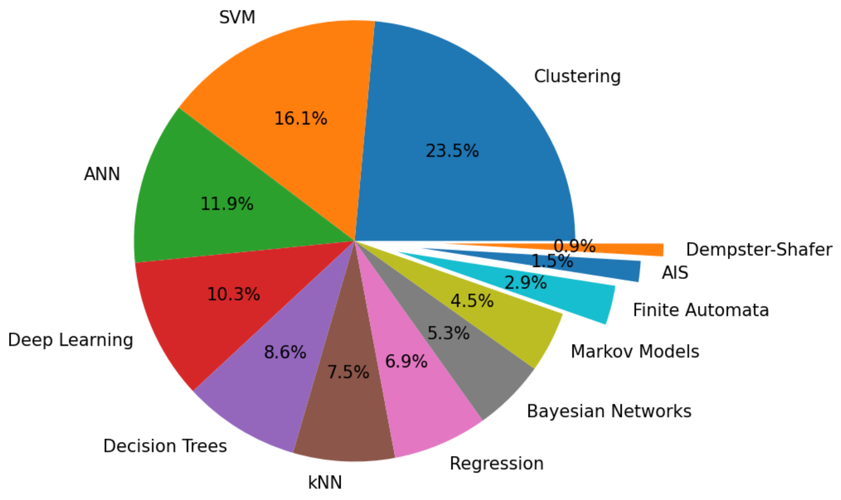


Figure 2.5: Relative results for all databases. Each database's results were summed and then the final percentage was computed by dividing each categories results with the full number of results for all categories. These results were last updated 10th August, 2021.

Despite these limitations, it was assumed that the results are somewhat representative of the main technologies being researched, and were deemed fit considering this is a heuristic approach. From Figure 2.5, it is possible to highlight clustering techniques as the most often researched, which can be justified by the fact that most technologies achieve clustering, namely SVM, which comes in second, and ANN and deep learning techniques with binary outputs, which come in third and fourth, respectively. ANN and deep learning techniques are about the same in prevalence, which can be attributed to the fact that deep learning techniques necessarily implement ANNs. Decision trees (including forest-like algorithms) and finite automata are good solutions for simple problems, but degrade in performance as the problem becomes more and more complex due to its reductive nature, and necessity of storing high quantities of data, and both suffer from a "state explosion" problem with the increase in complexity. kNN and other "nearest neighbors" are usually inefficient in what concerns memory, as data points need to be stored, some way or another. The least researched topic considering all databases is Dempster–Shafer's theory, which is used mainly for data fusion (sensor fusion, in particular) with its applicability in anomaly detection being mostly associated with the problem of intrusion detection. AIS is the second least researched topic across all databases analyzed, despite its intuitive applicability in anomaly detection (explained in Section 3.0.0.1), which can be due to its rather immature state [51] and the lack of an algorithm from AIS that has both the required efficacy and efficiency to be applicable in large scale scenarios, although some algorithms from AIS were deemed to complex, this field is still largely unexplored.

Given the lack of research into AIS, its analogy with anomaly detection, and the room for improvement it offers, the scope of this dissertation was further restricted to AIS algorithms, which will be discussed in the following chapter.

Chapter 3

Artificial Immune Systems Overview and Analysis

In this chapter the main AIS approaches will be introduced with a higher level of detail. Section 3.0.0.1 introduces the field of AIS. Section 3.1 discusses the most basic algorithms of each branch of AIS, including their biological inspiration and algorithm description, and Section 3.2 offers early results between these algorithms applied to anomaly detection, to explore each algorithm's potential and where they might be improved upon. A summary of the chapter is given in Section 3.3.

3.0.0.1 Introduction to Artificial Immune Systems

Artificial Immune Systems (AIS) are algorithms that are inspired by the field of immunology, being applied to computer science. Its main goal is to use known theories from immunology for problem solving where a parallel between a given problem and the immune system can be established. Its maturity as a field is still relatively new, and as previously stated in Section 2.3.1.2, research in this field as an anomaly detection solution is still comparatively recent. AIS is bio-inspired, as it draws analogies from biological functions, and progress in the field is closely coupled to research in biology.

One underlying principle of any AIS algorithm is that the (biological) immune system, mainly the human immune system, as a system that has been tried and tested naturally over the years, can be leveraged to produce working algorithms for multiple applications. However, this also means that the main goal is to produce, not a replica of the immune system, but a solution to a problem, and, thus, abstractions are implemented for the sake of practicality. Furthermore, the usage of immunology as inspiration can be considered problematic, as wrong theories or the inability to explain certain immune phenomena can translate to wrong abstractions in algorithms.

Applications of AIS include computer security, fault detection, fault-tolerant control, pattern recognition, clustering, and optimization.

The main identified branches of AIS are further explained in the upcoming chapter.

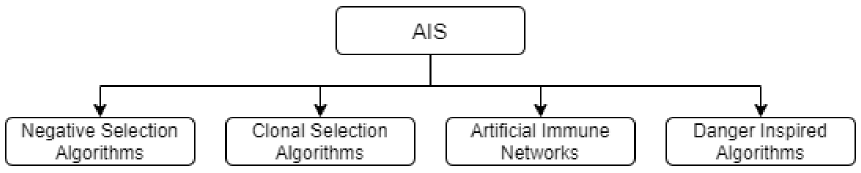

AIS
Negative Selection Algorithms
Clonal Selection Algorithms
Artificial Immune Networks
Danger Inspired Algorithms

Figure 3.1: AIS taxonomy used in this thesis.

3.1 Algorithm Branches Overview

As a relatively new research field, AIS has no universally agreed-upon division into sub-fields. In this section, the division presented in Figure 3.1 will be assumed, but producing an AIS taxonomy is out of the scope of this thesis, and this taxonomy will only be used for the structure. Each sub-field pertains to a particular biological model of the immune system, which will be briefly explained, albeit simplified, with specific scientific terms and biological phenomena that are not directly relevant to the respective algorithms reduced to a minimum (e.g. cellular organelles, specific substance names such as peptides, chemical interaction details such as the binding of antigens to cells). Note that the described algorithms do not intend to enforce a basis to which all and any algorithm in the sub-field must follow, but rather a generic algorithm that is an attempt at generalizing each algorithm according to multiple specific implementations, and, thus, as long as the underlying biological theory is the same, an algorithm that does not follow the presented generic structure can be developed under the same sub-field without loss of value.

3.1.1 Negative Selection Algorithm

The Negative Selection Algorithm (NSA) was first proposed in the work of S. Forrest et al. [52], in 1994. It is based on the self-non-self theory and was first used in computer virus detection through anomaly detection.

3.1.1.1 Biological Background: Self/Non-self

The main basis behind Self/Non-self theory [53] is that the immune system identifies organisms that are foreign to the body (non-self) and that the presence of such organisms leads to an immune response. In the vertebrate immune system, the theory hypothesizes that this is accomplished by a process of negative selection in the thymus, where immature T cells migrate to, to mature (step *a* in Figure 3.2). The thymus is a protected space where, ideally, only "self" antigens exist, i.e., it is highly unlikely that intruder organisms infiltrate the thymus. T cells that react to "self" antigens, i.e., recognize the body's own signature antigens, are sent an apoptotic signal and undergo apoptosis (programmed cell death) (step *b* in Figure 3.2). This way, matured T cells that leave the thymus, are only able to recognize "non-self", theoretically resulting in a distributed system of agents that can tolerate the system itself and detect intruders (step *c* in Figure 3.2).

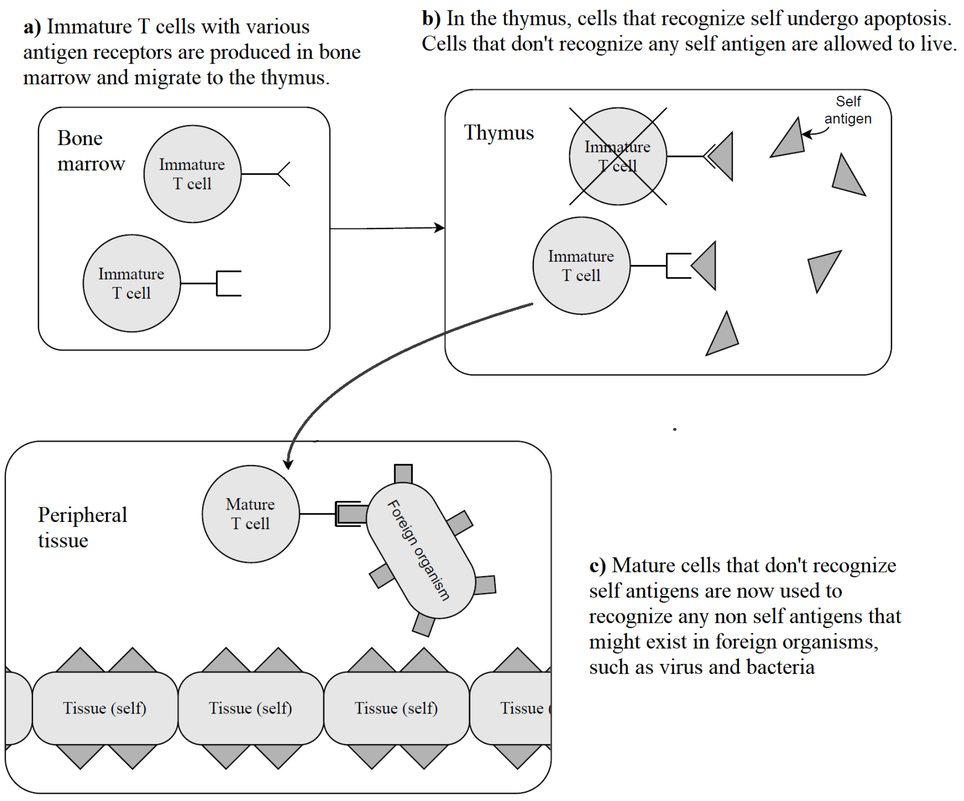

a) Immature T cells with various antigen receptors are produced in bone marrow and migrate to the thymus.
b) In the thymus, cells that recognize self undergo apoptosis. Cells that don't recognize any self antigen are allowed to live.
Bone marrow
Immature T cell
Immature T cell
Thymus
Immature T cell
Self antigen
Immature T cell
Peripheral tissue
Mature T cell
Foreign organism
c) Mature cells that don't recognize self antigens are now used to recognize any non self antigens that might exist in foreign organisms, such as virus and bacteria
Tissue (self)
Tissue (self)
Tissue (self)
Tissue

Figure 3.2: Simplified diagram of Self/Non-self theory. Note that self antigens are immensely varied, as are non-self antigens. This figure is for illustrative purposes only, and illustrates all self antigens as identical.

3.1.1.2 Algorithm Overview

NSA is inspired by the Self-Non-self theory [54], more specifically the negative selection of T cells. The analogy is rather straightforward for intrusion detection through anomaly detection, leading this application to be how the first instance of this algorithm was used by S. Forrest et al. [52]. The algorithm works by generating so-called "detectors" which are analogous to T cells. These detectors can be generated in any way, but are mostly generated randomly inside a search space that is problem-dependent. Then, an appropriate metric has to be implemented to compare these detectors to data points from known normal behavior. Detectors that can detect data points from this normal behavior are discarded, while detectors that do not detect normal behavior are saved for later use. Upon established stop criteria, detectors stop being created, and the detector set that is now "mature" is used for future classification of anomalies, under the assumption that if these detectors detect any instance of data, then it is likely an anomaly.

Algorithm 1 describes how NSA's detectors are generated.

To implement NSA the following has to be defined, according to the target application scenario:

- **Detectors:** how to represent detectors. E.g. real-valued arrays, strings. The nature of the detectors is also coupled to the input data that is to be classified. Depending on the nature of detectors, the following also has to be defined:
  - **Detection:** procedure that, given a detector and a point to classify, results in a binary decision of detection ("detected" or "not detected"). E.g. common n-contiguous bits/elements, euclidean distance compared to a threshold.
  - **Detector generation:** procedure that allows the generation of new detectors. E.g. random generation of detectors.
- **Stop criterion:** when to stop generating detectors during training phase. E.g. *n* number of detectors reached; estimated search space coverage.

Algorithm 1: General Negative Selection Algorithm Detector Set Generation

```
Input: $S_{self}$ set of points that are normal
Output: $D$ detector set
while not stop_condition() do
    $self\_detected = True$;
    while $self\_detected$ do
        $d = generate\_new\_detector()$;
        foreach $s_i$ in $S_{self}$ do
            if $detect(d, s_i)$ then
                break;
            end
            $self\_detected = False$;
        end
    end
    $D \leftarrow d$
end
return $D$
```

3.1.2 Clonal Selection Algorithm

The first instance of a Clonal Selection Algorithm (CSA) was proposed in the work of L. de Castro et al. [55], in 1999. It is based on the clonal selection theory, which aims to explain acquired immunity, and was first used in pattern recognition.

3.1.2.1 Biological Background: Clonal Selection

Clonal selection theory[54] aims to explain how B cells are selected and favored in response to their affinity towards antigens. B cells are responsible for secreting antibodies that can fight or inhibit pathogenic organisms, and T cells regulate the B cells' activity. Foreign organisms have their antigens collected and presented to B cells. Those B cells that have receptors that recognize these antigens (step *b* in Figure 2) will be split into two groups (step *c* in Figure 2). B cells react to antigens that they match with, in two simultaneous ways: on one hand, they differentiate into short-lived plasma cells that secrete the antibodies required for an immune response (step*d.1* in Figure 3.3); on the other hand, they clone themselves, generating long-lived memory cells (step*d.2* in Figure 3.3). The memory cells are clones of their parents but are subjected to hypermutation (mutation at high rates) to potentially create children that have a higher affinity than their parents. Throughout different kinds of antigen exposure, this leads to a pool of B memory cells that are proportionate to their affinity towards the most common antigens that the body found in the past, and these memory cells can readily respond to future occurrences of the antigen that lead to its proliferation, with even higher efficacy than their parents due to the hypermutations they were subjected to.

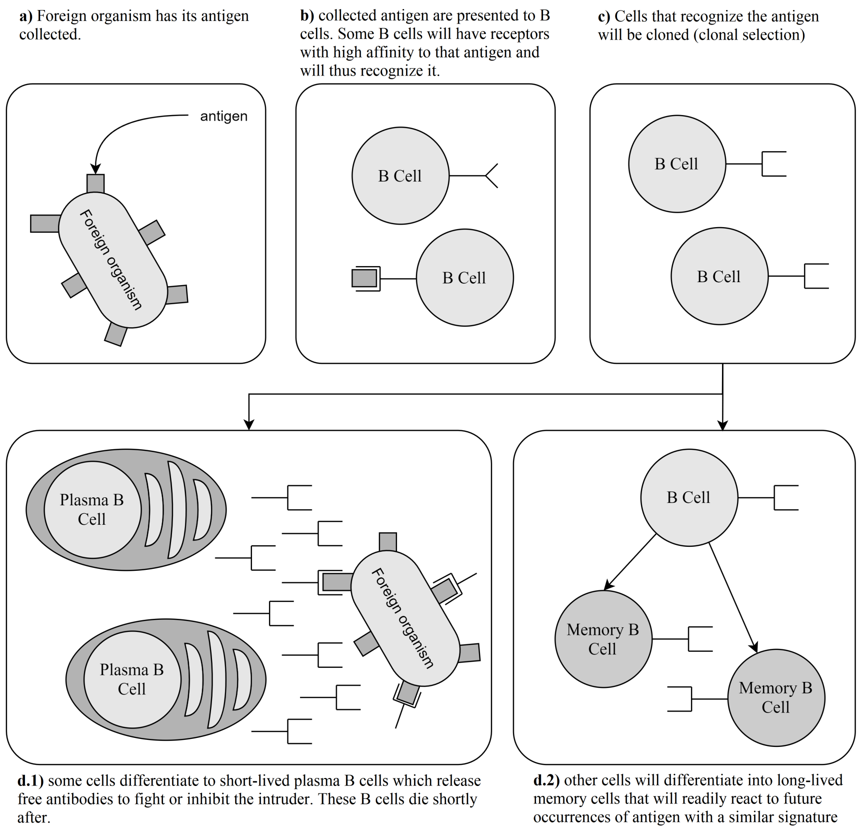

a) Foreign organism has its antigen collected.
b) collected antigen are presented to B cells. Some B cells will have receptors with high affinity to that antigen and will thus recognize it.
c) Cells that recognize the antigen will be cloned (clonal selection)
antigen
Foreign organism
B Cell
B Cell
B Cell
B Cell
Plasma B Cell
Plasma B Cell
Foreign organism
B Cell
Memory B Cell
Memory B Cell
d.1) some cells differentiate to short-lived plasma B cells which release free antibodies to fight or inhibit the intruder. These B cells die shortly after.
d.2) other cells will differentiate into long-lived memory cells that will readily react to future occurrences of antigen with a similar signature

Figure 3.3: Simplified diagram of clonal selection theory.

3.1.2.2 Algorithm Overview

CSA is inspired by the clonal selection theory, more specifically the selection of B cells according to their affinity to antigen, the selective cloning of selected B cells, and their subsequent mutation and differentiation into memory cells. The differentiation into plasma cells, albeit important in the context of the clonal selection theory itself, is not implemented in algorithms of this subgroup. The main principle of these algorithms is to pick, among candidate solutions (analogous to B cells, or, more directly, the antibodies they produce), those that have the highest (or lowest) affinity, storing them in a pool, and create new candidate solutions that result from small changes to the previous best solutions. The worst solutions are discarded keeping only the best candidates, although new candidates can be generated to provide more exploration to the algorithm. CSA is close to GA in that they both implement a Darwinian philosophy of "survival of the fittest" where the strongest solutions are the ones that carry over to the next generation, but they differ in how CSA clones and mutates the solutions in between generations. As the algorithm aims to maximize/minimize the affinity of the overall population, algorithms from this branch are mostly coupled to function optimization and pattern recognition [56].

Algorithm 2 describes how CSA's population is gradually generated and selected.

To implement CSA the following has to be defined, according to the target application scenario:

- **Antibodies:** how antibodies represent candidate solutions in the problem domain. Because CSA has many applications, it is difficult to contextualize precisely what is meant with "candidate solution". E.g. bit-string to match to patterns, set of parameters to be applied to a problem (such as hyperparameters for a machine learning model, or scheduling times for scheduling problems). Given the antibody structure, the following also has to be defined:
  - **Affinity:** how to rank candidate antibodies for selection. Overall, higher affinity means better solutions, although, in problems of minimizing a cost function, a higher affinity can be associated with worse solutions, instead. E.g. number of matches of specific bit-string to target set of bit-strings, the accuracy of machine learning model using a specific set of hyperparameters.
  - **Mutation:** how to slightly change antibodies. E.g. bit-flips in bit-strings, adding/multiplying a random real-valued number to a single random hyperparameter of a machine learning model.
  - **Generation:** how to produce new antibodies. E.g. random bit-strings, random sampling of hyperparameters from a parameter sweep (also known as grid search) search space.
  - **Selection:** how to select which antibodies survive. Selection can happen in multiple ways in the algorithm, namely: which antibodies will be chosen to be cloned; which antibodies will be chosen to survive, and, conversely, which will be replaced by newly generated ones. This is mostly accomplished by simply picking the highest-ranking ones, according to affinity, but random new antibodies can be left to survive, or be subjected to cloning, to potentially find other optimal solutions, and avoid converging to local optima.
  - **Cloning:** how to distribute clones. The cloning process itself consists in creating copies, but the number of copies to create, and subsequently mutate, can be managed

in many ways. E.g. create clones proportional to affinity rankings; create clones of x best antibodies; create clones of the entire population.

- **Stop criterion:** when to stop generating solutions. E.g. hard limit on iteration number, decline in affinity over generations.

Algorithm 2: General Clonal Selection Algorithm, adapted from [57]

```
Input: $N$ population size
$n$ number of antibodies to clone
$d$ number of new antibodies to be generated each iteration
Output: $P$ best population
1  $P \leftarrow generate\_antibodies(N)$
2  while not stop_condition() do
3      foreach $p$ in $P$ do
4          $update\_affinity(p)$
5      end
6      $P' \leftarrow select(P,n)$ foreach $p'$ in $P'$ do
7          $C \leftarrow clone(p')$
8      end
9      foreach $c$ in $C$ do
10         $hypermutate(c)$
11     end
12     foreach $c$ in $C$ do
13         $update\_affinity(c)$
14     end
15     $P \leftarrow insert(C,n)$
16     $P_{new} \leftarrow generate\_antibodies(d)$
17     $P \leftarrow replace(P,Pr)$
18 end
```

3.1.3 Artificial Immune Networks

Early work in Artificial Immune Network (AIN) was proposed independently both in the work of L. de Castro et al. [58] and J. Timmis et al. [59], in 2000, for clustering data. It is based on the immune network theory, which aims to explain acquired immunity as a deeply dynamic phenomenon.

3.1.3.1 Biological Background

The immune network theory or idiotypic network theory [60] proposes that the immune system is more dynamic than previously suggested. Rather than waiting for external organisms to invade and stimulate a response, the immune system is constantly stimulating itself. The theory suggests that immune cells not only recognize antigen from foreign organisms but also recognize each other (either through each other's receptors or through free antibodies), creating a loose and dynamic

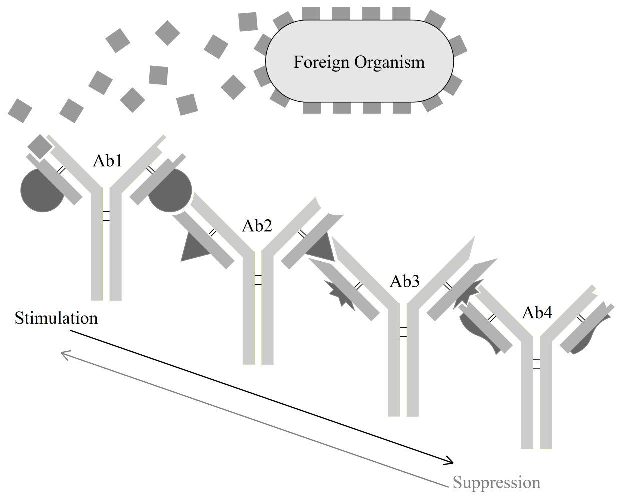


Figure 3.4: Richter's model [3] of an antibody chain. Each antibody is labeled Abx and the antigens are pictured as dark squares. This example showcases how an antigen can trigger a cascade of reactions. The presence of the antigen is initially recognized by $Ab1$, which increases in number. This leads to $Ab2$ to start recognizing $Ab1$, which prompts $Ab2$ to increase in number and so on. As the antibodies increase due to stimulation, they also inhibit the antibodies lower on the chain, regulating the response. An increase in $Ab4$ will lead to a reverse cascade to gradually suppress the subsequent antibodies.

network of immune cells. The essential idea is that the immune system learns over time, not exclusively by the stimulation of individual cells by antigens, but rather by the network of immune cells as a whole. This implies that the immune system tolerating a given antigen, may not mean that the antigen was not recognized, but instead, it was recognized by some immune cells, which, in turn, were recognized by other cells, which suppressed the former cells' response or even eliminated some of these cells. Conversely, an antigen may be recognized by some immune cells, and this can further be potentiated by other immune cells recognizing the former cells. In this perspective, any given cell can stimulate or suppress matching cells that it recognizes, indirectly affecting how the system responds to antigens, and thus, the immune network theory can be seen as an expansion on the clonal selection theory explained in Section 3.1.2.1. Note that the immune network theory is not as intuitive as the remaining theories explained in this chapter, even to immunologists, and, according to G. W. Hoffman, the "proposal initially seemed to many immunologists to make a complex subject even more complex, in fact unmanageably so." [61]. Multiple models attempt to describe the network's dynamics, e.g. Figure 3.4 showcases the Richter model [3], more specifically, a scenario that exemplifies how multiple B cells (and their antibodies) can react to antigens, even if a majority of those cells do not directly react to the specific antigens that began the response.

3.1.3.2 Algorithm Overview

An AIN explores the interconnectivity of immune cells, mainly their mutual suppression/stimulation mechanism. The main focus of these algorithms is to create populations that are dynamically linked over generations. AIN can be distinguished from CSA as the evolution of the population is no longer individualized, and there is always some sort of interaction between the population's participants. Algorithms under this category are vastly different in how they interpret biological theory to solve problems. In@articleKnight2001,this thesis, special attention is given to clustering approaches. As an example, for clustering, usually, the populations are represented as graphs, where nodes symbolize the cells, and edges symbolize the stimulation/suppression of a given cell to other cells it "recognizes". This recognition is abstracted as affinity, usually by some measurement of similarity, such as Euclidean distance [58]. Through this affinity, each cell will selectively clone itself and potentially mutate itself, or die off, which, in theory, will lead to clusters of cells that represent the clusters of the original data. Each of these clusters is named Artificial Recognition Ball (ARB) and is made up of multiple B cells. Algorithm 3 shortly introduces AINE [62], where each ARB includes a data representation, number of B cells, and stimulation values.

3.1.4 Danger-Inspired Algorithms

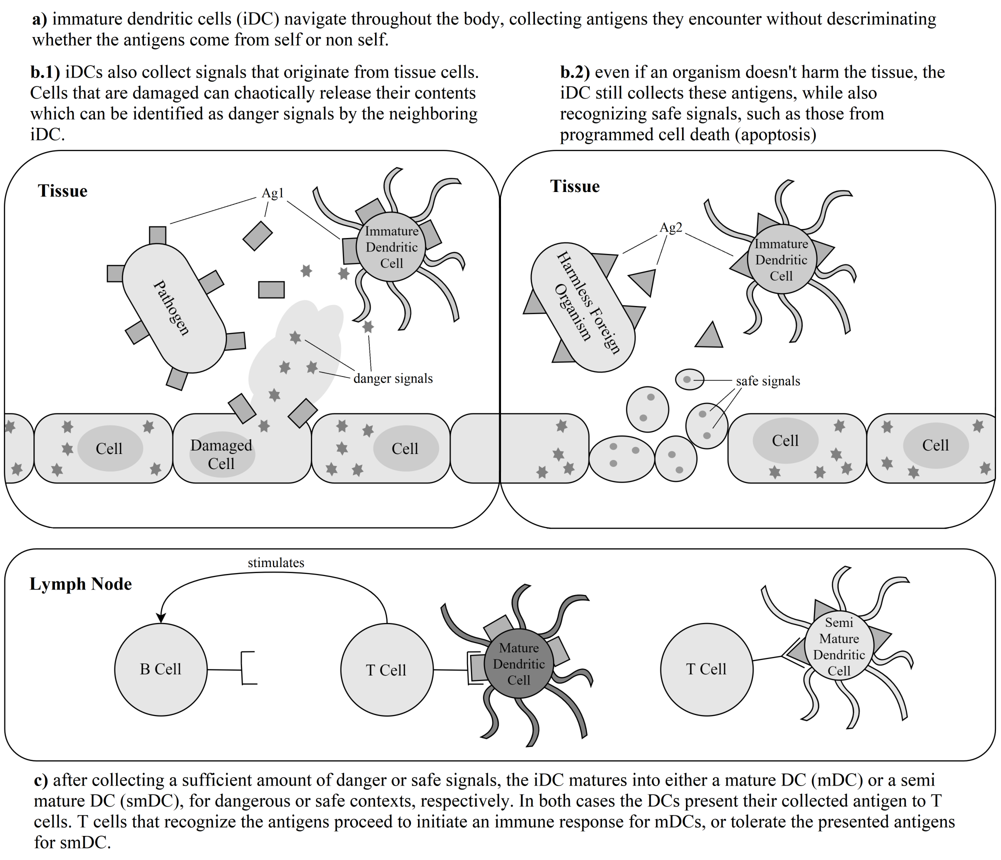
A Danger Inspired Algorithm (DIA) is an algorithm that draws inspiration from the danger theory [53]. This subfield is relatively new with the most prominent algorithm being DCA originally proposed by J. Greensmith et al. [63], implemented for intrusion detection through anomaly detection.

3.1.4.1 Biological Background

Danger theory [53] addresses key shortcomings identified in Negative Selection. It aims to explain how the human immune system can start an immune response to things that are self, such as cancer cells, and avoid immune responses to things that are non-self, but beneficial or harmless, such as gut microbiota. The theory proposes that, although the immune system can identify self and non-self, it is not this phenomenon that triggers immune responses. Rather, a specific kind of cell - the Dendritic Cell (DC) - functions as a biological detective that navigates body tissue. These DCs start their life cycle as being immature, a period during which they collect any antigens they find, and also recognize signals from the tissue itself and Pathogen-Associated Molecular Pattern (PAMP) - molecules that function as signatures of pathogenic presence. The origin of these signals is not fully known, but the theory proposes that tissue cells can communicate if they are in distress (e.g. in the event of the tissue's cells being killed or presence of PAMP) or if they are safe (e.g. the cell died naturally through programmed cell death). As the immature DC collects antigens and identifies the tissue signals it eventually differentiates into either a mature Dendritic Cell (mDC) or a semi-mature Dendritic Cell (smDC) if they come from a "dangerous" context, or a "safe" context, respectively. Then, they migrate to a nearby lymph node, where T cells are found, and present their collected antigens to T cells that recognize them. For mDC, T cells that

Algorithm 3: A specific AIN implementation for generating an ARB population, based on AINE [62]

```
Input: $A$ input data (antigens)
$N$ population size
$NAT$ network affinity threshold
$R_m$ mutation rate
$N_c$ number of clones on each ARB
Output: $S_{ARB}$ set of all ARBs
$S_{ARB} \leftarrow initialize\_population()$
$C_{ARB} = \{\}$
while $True$ do
    foreach $arb$ in $S_{ARB}$ do
        foreach $a$ in $A$ do
            $present\_antigen(p)$
        end
    end
    foreach $arb$ in $S_{ARB}$ do
        $update\_stimulation(arb)$
    end
    foreach $arb$ in $S_{ARB}$ do
        $allocate\_b\_cells(arb)$
    end
    if $stop\_condition()$ then
        break
    end
    $C_{ARB} = clone\_and\_mutate(S_{ARB})$ foreach $arb$ in $C_{ARB}$ do
        $S_{ARB} \leftarrow arb$
    end
end
return $S_{ARB}$
```


a) immature dendritic cells (iDC) navigate throughout the body, collecting antigens they encounter without descriminating whether the antigens come from self or non self.

b.1) iDCs also collect signals that originate from tissue cells. Cells that are damaged can chaotically release their contents which can be identified as danger signals by the neighboring iDC.

b.2) even if an organism doesn't harm the tissue, the iDC still collects these antigens, while also recognizing safe signals, such as those from programmed cell death (apoptosis)

c) after collecting a sufficient amount of danger or safe signals, the iDC matures into either a mature DC (mDC) or a semi mature DC (smDC), for dangerous or safe contexts, respectively. In both cases the DCs present their collected antigen to T cells. T cells that recognize the antigens proceed to initiate an immune response for mDCs, or tolerate the presented antigens for smDC.

Figure 3.5: Simplified model of danger theory. Note that immature dendritic cells collect a vast variety of antigens, not just one specific type, which is not illustrated in this figure.

recognize its antigens may start an immune response; conversely, for smDC, T cells that recognize its antigens will tolerate the antigens presented. Figure 3.5 illustrates the interaction between DCs, the organism's tissue, and T cells in the perspective of danger theory.

3.1.4.2 Algorithm Overview

Danger-inspired algorithms draw inspiration from danger theory's principle that immune responses are triggered by the damaging effects of pathogen on tissue, rather than by directly identifying the pathogen. In the case of Toll-Like Receptor Algorithm (TLRA) [64], the algorithm defines what is meant by PAMP, and defines a T cell matching metric. In the case of DCA [63] four signals are available for definition: safe, danger, PAMP and inflammatory cytokines. A more detailed description of DCA is given in Section 3.2.6.

3.2 Algorithms Analysis for Anomaly Detection

In this section, some algorithms are implemented by this dissertation's author for a simple benchmark dataset and compared for application in anomaly detection. These implementations are not

Table 3.1: Summary table of random circle dataset's features.

Feature Order	Feature	Data Type
1	ID number	Integer
2	Label	
3	X coordinate	Real Valued
4	Y coordinate	

optimized for performance and are an attempt at evaluating these algorithms comparatively, to identify potential weaknesses and strengths to select one which to improve on in this thesis. First, the overall procedure is explained in Section 3.2.1, then the two used datasets are introduced in Section 3.2.2. Each algorithm used is explained in Sections 3.2.3 through 3.2.6. Finally, the results of applying these algorithms to the two datasets are discussed in Section 3.2.7 with limitations of this comparative study being shown in Section 3.2.8

3.2.1 Procedure

The implemented algorithms are applied to two datasets (described in the next section). All algorithms were implemented in Python, using only the built-in Python packages, to avoid differences in performance due to one problem being easier to adapt to a package with good performance, in which case the difference in performance would be due to the package itself, and not the algorithm being somehow faster in nature. Based on Section 2.2.2.2, three metrics are used: precision, recall and F_1 score. Furthermore, to evaluate the applicability of the algorithms in real time scenarios, the time per data point will also be measured. Finally, for scalability, the effect on relative time due to problem complexity will be analyzed.

3.2.2 Target Datasets

For each algorithm, two datasets will be used. The first dataset (Section 3.2.2.1) is a simple dataset that aims to validate that the algorithm works. The second dataset (Section 3.2.2.2) is a well-known benchmark dataset used in many machine learning approaches, adapted here for anomaly detection.

3.2.2.1 First Dataset: Random Circle

This dataset is a randomly generated dataset where multiple two-dimensional points were generated between 0 and 1 (for both dimensions). If the point's distance to the center point with coordinates $(0.5, 0.5)$ is lesser than 0.2 the point is considered normal, and if it is greater than 0.2, the point is considered anomalous. This dataset's simplicity is meant to ease the first validation for the algorithms in the following sections. Table 3.1 summarizes each feature. There are 800 points labeled as normal, and 200 points labeled as anomalous.

3.2.2.2 Second Dataset: Wisconsin Breast Cancer Diagnosis Dataset

The Wisconsin Breast Cancer Diagnosis Dataset [65] is a well-known two-class dataset that consists of 32 features obtained from multiple breast cancer images. Table 3.2 summarizes each feature. The dataset consists of 357 benign data points and 212 malignant data points, for a total of 569 points. From the dataset's source images, 10 features were extracted, and then, for each feature, the mean, the standard deviation, and the mean between the three worst values were computed, effectively tripling the features to 30. Then each point contains its unique identification number and its classification for a total of 32 features.

This dataset was not used in its raw form. First, the labels of 'B' for benign and 'M' for malignant were replaced by numerical values: 0 and 1, respectively. Then, the data were scaled in the range of $[0, 1]$ across all its data features (i.e. features 3-32), and this was saved as a separate dataset. Additionally, the original dataset was standardized (i.e. its means were made null and its standard deviations were made unitary across all data features), after which, Principal Component Analysis (PCA) was applied for 1 component, 2 components, and so on, up to 29 components, such that each result was saved as a separate dataset, for a total of 29 datasets. These 29 datasets will be used to test each algorithm's capacity for scalability. PCA is a technique that allows the projection of m-dimensional numeric data into n-dimensional data, such that $n < m$. This is accomplished by finding n orthogonal unit vectors that follow along with the directions that best fit the original data's variance, and then project that original data into the new-found vectors, effectively reducing the dimensions of data while attempting to lose the least amount of information possible. Figure 3.6 is an example of applying 1, 2, and 3 component PCA (i.e. reducing data from 30 dimensions to 1, 2, and 3 dimensions, respectively) to the standardized data. Although this was not the initial purpose of applying PCA, this allows us to have an initial grasp on the data's distribution, showing that there is a separation in data in what concerns benign cells and malignant cells (represented in blue and red, respectively, in Figure 3.6).

3.2.3 Real-Valued Negative Selection Algorithm

NSA originally used bit-strings and metrics such as Hamming distances and n-contiguous bits to compare antigen and T cell (detector). Z. Ji and D. Dasgupta [66] developed the first real-valued NSA, which was chosen to be implemented, as real-valued data is often more adequate for the task of monitoring physical processes. Following the requirements previously described for these sorts of algorithms in Section 3.1.1.2, the following was defined:

- **Detectors:** Real-valued arrays. Each array represents a point in space with the same dimensions as the input data.
  - **Detection:** if a given point is closer than a given distance (Euclidean) to one of the detectors, it is considered anomalous. This way, each detector can be seen as an hyper sphere of "non-self". A point is considered normal if it matches no detector.

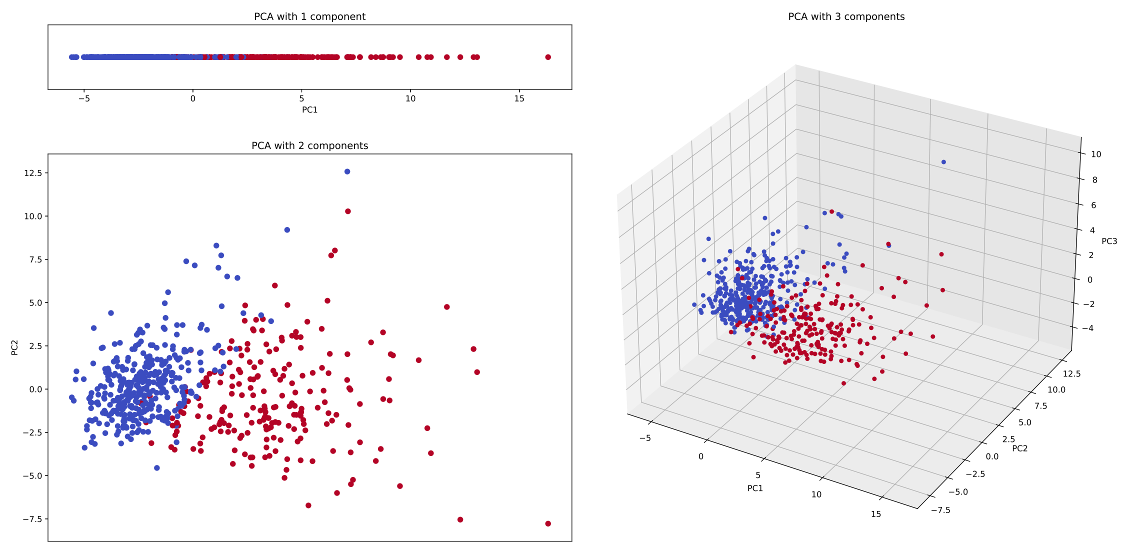


Figure 3.6: PCA applied with 1 component (upper left), 2 components (bottom left) and 3 components (right). In the two leftmost plots, it is possible to visualize the projection of the 2 components PCA into the single component PCA plot, notably in the rightmost outlier point (isolated point in red).

Table 3.2: Summary table of UCI's Wisconsin Breast Cancer Diagnosis dataset's features.

Feature Order			**Feature**	**Data type**
1			ID number	Integer
2			Diagnosis	Character
Mean	St. dev	Worst		
3	13	23	radius (mean of distances from center to points on the perimeter)	Real Valued
4	14	24	texture (standard deviation of gray-scale values)	
5	15	25	perimeter	
6	16	26	area	
7	17	27	smoothness (local variation in radius lengths)	
8	18	28	compactness $\left(\frac{perimeter^2}{area} - 1\right)$	
9	19	29	concavity (severity of concave portions on the contour)	
10	20	30	concave points (number of concave portions of the contour)	
11	21	31	symmetry	
12	22	32	fractal dimension ("coastline approximation" - 1)	

Table 3.3: Parameters for the NSA algorithm version implemented in this thesis.

Parameter	Description
N	Number of detectors to be generated in training
$dist_{threshold}$	Distance threshold which is to be used both in eliminating detectors during training phase, and to be used during evaluation to classify self and non-self (normal points and outliers, respectively)

  - **Detector generation:** detectors are generated randomly using the input data's bounds as limits.

- **Stop criterion:** generate detectors until a specified number is reached.

First, to train the model we need to generate detectors. Thus, we take a subset of the points under the assumption that this subset is normal (no outliers exist) as training data. For each potential candidate detector, d_i, a random array with the same dimension (i.e. number of features) as the input data is computed; d_i is compared towards all data points in the normal subset using the Euclidean distance as described in Equation 3.3, if this distance is lesser than a predetermined value, $dist_{threshold}$, d_i is discarded and a new detector undergoes the same procedure, if the distance is greater than $dist_{threshold}$, the detector is stored in a detector set, D. The training procedure finishes when the set D has a predefined number of detectors, N.

After training, for each point to be classified, x_i, with m features (Equation 3.1), we evaluate its distance to all detectors in D (which now has a structure according to Equation 3.2) using Euclidean distance and classify it as anomalous if its distance to any single. Table 3.3 summarily describes all user parameters for the version implemented in this section.

$$X = \{x_0, x_1, \dots, x_n\},\ x \in \mathbb{R}^m \tag{3.1}$$

$$D = \{d_0, d_1, \dots, d_N\},\ d \in \mathbb{R}^m \tag{3.2}$$

$$distance(x_i, d_i) = \sqrt{\sum_{a=0}^{m} ((x_i^a)^2 - (d_i^a)^2)} \tag{3.3}$$

3.2.4 Clonal Selection for Optimization

CSA was originally used for pattern recognition and function optimization, and, as such, its potential for anomaly detection is not clear. Some solutions implement a hybrid where CSA is used to optimize the suitability of another algorithm's parameters (e.g. NSA), but it is the latter algorithm that is responsible for anomaly detection. To the best of this dissertation's author's knowledge,

Table 3.4: Parameters for the CSA version implemented in this thesis.

Parameter	Description
N_A	Population size at the end of each iteration of the algorithm
N_C	Number of clones that will be produced at each iteration
N_{new}	Newly generated antibodies produced at each iteration
I_{max}	Maximum number of iterations the algorithm goes through
Aff_{target}	Affinity value to be achieved, after which the algorithm stops

no implementation of CSA alone achieves anomaly detection. Nevertheless, the optimizing properties of CSA were tested towards optimizing NSA's parameters. Following Section 3.1.2.2's requirements:

- **Antibodies:** parameter pairs (N and $dist_{threshold}$) to apply to NSA.
  - **Affinity:** performance of NSA for the antibody. This is evaluated by promoting higher F_1 scores and estimated area coverage, and penalizing algorithm running time.
  - **Mutation:** for a given antibody, randomly select one of its parameters and multiply it by a random value inside $[2/3, 3/2]$.
  - **Generation:** randomly select parameter pairs inside a given range of values.
  - **Selection:** select the N_A highest affinity antibodies.
  - **Cloning:** clone the N_C highest affinity antibodies.
  - **Stop criterion:** max iterations, I_{max} reached, or target affinity, Aff_{target}, reached.

For the optimization of NSA, first, a static set of "self" points is stored for training. For each antibody 3.4's affinity, the NSA algorithm runs for a test dataset, and its F_1 score (previously described in Section 2.2.2.2) and detector estimated coverage are calculated, and its time is averaged between each data point. The highest affinity N_C antibodies are cloned, mutated, and re-evaluated, and the worst antibodies are eliminated and replaced by N_{new} new ones, such that the population has N_A antibodies. The entire procedure is performed for either a max number of iterations, I_{max}, or until the best affinity in the population achieves a predetermined value, Aff_{target}.

The detector estimated coverage is determined by generating evenly spaced points along with the data space and computing the ratio of non-self points to self points that result after running the algorithm for all points (Equation 3.6). The area, F_1 score, and execution time are all fused into a single affinity value as described in Equation 3.5. This equation always leads to affinity values between 0 and 1 and only works if all factors are also between 0 and 1, such that the factor of all these values will be highly penalized by the lowest of them. To normalize the impact of the execution time, a custom sigmoid function is applied, such that low values of time will lead to values close to 1, and high values of time will lead to values close to 0. This customized sigmoid is shown in Equation 3.7, where $midpoint$ is the value of x for which the function evaluates as 0.5, and $slope$ is the slope of the function at the midpoint. This customized sigmoid allows us to define a soft limit on desired execution time such that values well over this limit are very close to 0, and, therefore, lead to an affinity that is also very close to 0 and undesirable. The *slope*, in this context, has to be a negative number and defines how steep the transition between 1 and 0 is, such that a value close to 0 leads to a very soft curve, and a value approaching $-\inf$ resembles a reversed unit step function.

$$A = \{a_0, a_1, \dots, a_{N_A}\},\ a_i = \{N^i, dist^i_{threshold}\} \tag{3.4}$$

$$affinity(a_i) = area \times F_1 \times sigmoid(time_{execution}) \tag{3.5}$$

$$area = \frac{TP+FP}{TN+FN} \tag{3.6}$$

$$sig(x) = \left(1 + e^{4\times slope \times (x - midpoint)}\right)^{-1} \tag{3.7}$$

3.2.5 Immune Networks

AINs can be used as anomaly detectors by learning normal data distribution through its clustering properties, and, somehow, quantify how much a given point belongs to a cluster. Let this metric of how much a point belongs to a cluster be named "normality score". It is then possible for points to be classified as normal points if their normality score is high (above a certain threshold), and be classified as abnormal in the opposite scenario. It is not the main scope of this thesis to conduct a thorough exploration of all AIS algorithms. This work is merely indicative of which algorithm to pursue to adapt/improve for real time anomaly detection. This, aligned with some limitations, lead to the decision of not further exploring this family of algorithms, but their applicability for anomaly detection is not discarded. The following reasons are presented:

- Unintuitive: the algorithm's interactions are hard to understand, variations in its parameters are not very clear in their effect, and the algorithm's evolution is highly unpredictable. The biological theory that inspires it is loosely coupled to the algorithms in this family, which further hinders the ease of understanding the algorithm.
- Complex: as it contains a highly networked and dynamic structure, its time complexity discourages its application in real time scenarios.
- Scope: as mainly a clustering algorithm, its suitability for anomaly detection is not clear, although, as was described earlier, it is possible to still be applicable for this family of problems. To the best of this dissertation's author's knowledge, this was never done.

3.2.6 Dendritic Cell Algorithm

J. Greensmith's thesis [67] proposed DCA for anomaly detection applied to intrusion detection as a DIA. This algorithm draws direct inspiration from DCs as cells that indiscriminately collect data (antigens). The algorithm, as implemented in this section, is listed in Algorithm 4. DCA has no training phase, instead, when implemented, danger, safe, PAMP and inflammatory signals need to be extracted from data. The way these signals are extracted, currently, rely on expert knowledge. Each signal represents degrees of danger (as a non-negative real value or integer) in the problem domain:

- PAMP signals indicate the presence of danger with great confidence;
- Safe signals indicate the absence of danger with moderate confidence;
- Danger signals indicate the presence of danger with low confidence;
- Inflammatory cytokines increase the magnitude of all previous signals, increasing their effects.

As previously mentioned, the modeling of these signals is coupled with the problem, i.e., the anomalies to detect. E.g. in intrusion detection, the PAMP signal can be associated with an increase in incoming connections [68], error messages [67] and danger signals can be packet counts, message size, and also a fusion of multiple features [44].

Each data point is processed in the algorithm as antigen-signal pairs. The signals were described in the previous paragraph, the antigens are unique identifiers and will ultimately be the target for detection. Thus, antigens are what we want to classify, and signals are what we use for that classification.

DCA collects these data points over time, resulting in classifications that take into account the context of recent and future data. A single data point with a large danger signal will probably be considered safe if surrounded by safe-signaling data points. The way classification is accomplished is by distributing data points among immature Dendritic Cell (iDC)s. Each iDC will store the antigen (i.e. the identifier) and use its signals to update its state in two ways: it will update its context value, k, which indicates danger (positive values) or safety (negative values) and increases with PAMP and danger signals, and decreases with safe signals; it will update its costimulation value, csm, which denotes how much it has processed signals, increasing with PAMP danger and safe signals. Inflammatory cytokines increase the magnitude of all three signals, effectively increasing the rate at which both k and csm change. Equation 3.9 is a matrix adaptation of J. Greensmith's method for calculating k and csm signals from safe, danger, PAMP and inflammatory signals, while Equation 3.10 is the same, but in iterative form, which is effectively the one used in the implemented algorithm in this section. When the iDC's csm reaches a given threshold, the iDC matures into either mDC or smDC, depending on whether k is greater than 0 or not, respectively. A given antigen can be classified as normal or anomalous according to its Mature Context Antigen Value (MCAV). The MCAV value is computed according to Equation 3.8, where $\#mDC_{Ag_i}$ denotes the number of times that a given antigen, Ag_i, was found in a DC that matured into a mDC and $\#DC_{Ag_i}$ is the number of times that a given antigen, Ag_i, was collected by any DC. If the MCAV is greater than a predefined threshold the antigen is considered anomalous, else it is considered normal.

Note that in this implementation, the antigen pertains to the specific data point, but this is not a necessity (e.g. the antigen can be a process ID which originated data points, in which case it is the process ID, and therefore the process, that gets classified, not the individual data points).

$$MCAV(Ag_i) = \frac{\#mDC_{Ag_i}}{\#DC_{Ag_i}} \tag{3.8}$$

$$\begin{bmatrix} csm \\ k \end{bmatrix} = \begin{bmatrix} w_{P,csm} & w_{D,csm} & w_{S,csm} \\ w_{P,k} & w_{D,k} & w_{S,k} \end{bmatrix} \begin{bmatrix} \sum_i P_i * (1+I_i) \\ \sum_i D_i * (1+I_i) \\ \sum_i S_i * (1+I_i) \end{bmatrix} \tag{3.9}$$

$$\begin{bmatrix} csm_i \\ k_i \end{bmatrix} = \begin{bmatrix} csm_{i-1} \\ k_{i-1} \end{bmatrix} + \begin{bmatrix} w_{P,csm} & w_{D,csm} & w_{S,csm} \\ w_{P,k} & w_{D,k} & w_{S,k} \end{bmatrix} \begin{bmatrix} P_i \\ D_i \\ S_i \end{bmatrix} * (1+I_i) \tag{3.10}$$

3.2.6.1 Signals and Antigens

Given the nature of DCA, the description of the implementation would not be complete without the functions that allow the extraction of signals from data points. For each dataset presented in Section 3.2.2, different signals will be used. In each case, the antigens were the unique IDs of each data point, as the goal is to classify each data point.

For the first dataset, it is known that the data points follow a perfect circle centered in point $(0.5, 0.5)$. Thus, we can model each point's safety as the distance of the point to the circumference if it is inside the circle, and the danger signal as the same distance, if it is outside the circle. The distance metric is the same as previously shown in Equation 3.3. Equations 3.11 and 3.12 show the signal functions used, where p_i is the point whose signals are to be calculated, and p_0 is the center point at $(0.5, 0.5)$. Recall that the radius of the circle in this dataset is 0.2.

$$danger(p_i) = max\,[\,(distance(p_i, p_0) - 0.2)\,,\,0] \tag{3.11}$$

$$safety(p_i) = max\,[\,(0.2 - distance(p_i, p_0))\,,\,0] \tag{3.12}$$

The second dataset needs to be examined. To use the data that is most meaningful for the label, the Pearson's correlation coefficient between each feature and the label were computed (Equation 3.13, such that y is the label's numerical value of 0 or 1). Note that Pearson's correlation coefficient does not apply to categorical data, but in the case of encoded binary categorical data (as is the case), it is still applicable. These results are used for two purposes: the most highly correlated value with the label will be used for both PAMP and safe signals; the remaining values will be used for the danger signal which will be obtained as a weighted sum of each feature's difference to the mean of normal data points only (Equation 3.16, where $\overline{X_j^{normal}}$ denotes the mean

Algorithm 4: DCA algorithm implemented in this section. This algorithm is adapted from J. Greensmith [67]

```
Input: $Ag$ input data (antigens)
       $N$ DC population size
       $c$ number of times any antigen will be presented to distinct DCs
       $t_{migration}$ migration threshold after which DC matures
       $t_{mcav}$ MCAV threshold above which the antigen is considered anomalous
Output: Classification of antigens as normal or anomalous

foreach $ag$ in $Ag$ do
    $signals = compute\_signals(ag)$
    $DC_{sample} = sample\_random(c)$
    foreach $dc$ in $DC_{sample}$ do
        /* update csm and k of each cell and store the antigen in
           their repertoire                                          */
        $expose(dc, ag, signals)$
        if $dc.csm > t_{migration}$ then
            if $dc.k > 0$ then
                /* this will increment the count of mDC for all
                   antigens that this cell had                       */
                $mature(dc, MATURE)$
            else
                $mature(dc, SEMIMATURE)$
            end
            $reset(dc)$
        end
    end
end

foreach $ag$ in $Ag$ do
    $mcav = \frac{mDC\_count(ag)}{c}$
    if $mcav > t_{mcav}$ then
        $classify(ag, OUTLIER)$
    else
        $classify(ag, NORMAL)$
    end
end
```

of feature X_j considering only benign data points.), in which the weights will be the respective correlation coefficients divided by the sum of all coefficients, as shown in Equation 3.15 to prevent the danger signal from being too high as a result of summing too many values. The PAMP and safe signals are, in this case, mutually exclusive, and are given in Equations 3.17 and 3.18, where $\overline{o}$ corresponds to the mean of the most correlated feature (the texture mean, feature order 4 in Table 3.2) across its normal occurrences, only, ignoring the anomalous (malignant) data points, and t_o denotes the greatest distance between any normal point and $\overline{o}$.

$$\rho(X,Y) = \frac{\sum_{i=0}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=0}^{n}(x_i-\overline{x})^2\sum_{i=0}^{n}(x_i-\overline{x})^2}} \tag{3.13}$$

$$p_i = \{x_i^3, x_i^4, \ldots, x_i^{32}\} \tag{3.14}$$

$$w_{X_i} = \frac{\rho(X_i,Y)}{\sum_{j\in\{3,5,\ldots,32\}}\rho(X_j)} \tag{3.15}$$

$$danger(p_i) = \sum_{j\in\{3,5,\ldots,32\}} w_{X_i} \times (x_i^j - \overline{X_j^{normal}}) \tag{3.16}$$

$$PAMP(p_i) = max\ (\ |x_i^4 - \overline{o}| - t_o\ ,\ 0) \tag{3.17}$$

$$safe(p_i) = max\ (\ t_o - |x_i^4 - \overline{o}|\ ,\ 0) \tag{3.18}$$

With these signals defined, the previously described Algorithm 4 can be applied.

3.2.7 Results and Discussion

All specific algorithms mentioned in the previous sections (Real-Valued NSA, CSA for optimization and DCA) were *implemented* (not *designed*) by me, using Python for faster prototyping. In this section, the validity of the implemented algorithms is evaluated, and their performance and parameter effect will also be analyzed. Section 3.2.7.1 offers initial insights into each algorithms' advantages and shortcomings, while Section 3.2.7.2 focuses on evaluating performance for different data dimensions (as a representation of scalability) and aims to further validate the algorithms' performance in data that is not so clearly separable.

3.2.7.1 Random Circle Dataset

For all algorithms, the random circle dataset was split into 200 normal occurrences for training, with 600 normal occurrences and all 200 abnormal occurrences being used for testing. Note that DCA has no training phase, thus, to simulate this, the mentioned procedures for feature means in normal occurrences were applied only to this training subset, instead of all normal occurrences. In

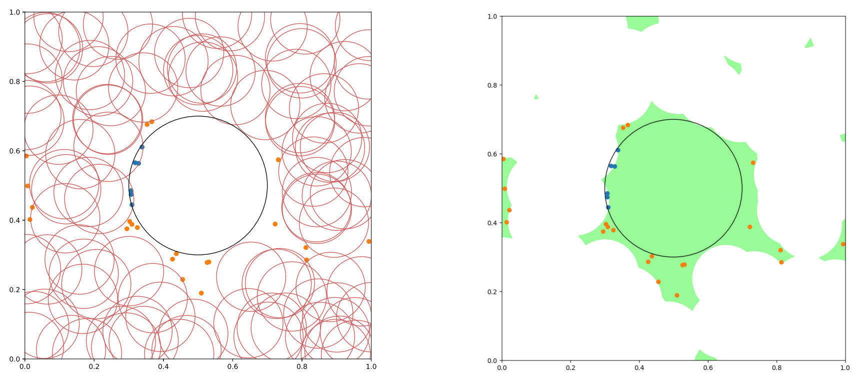
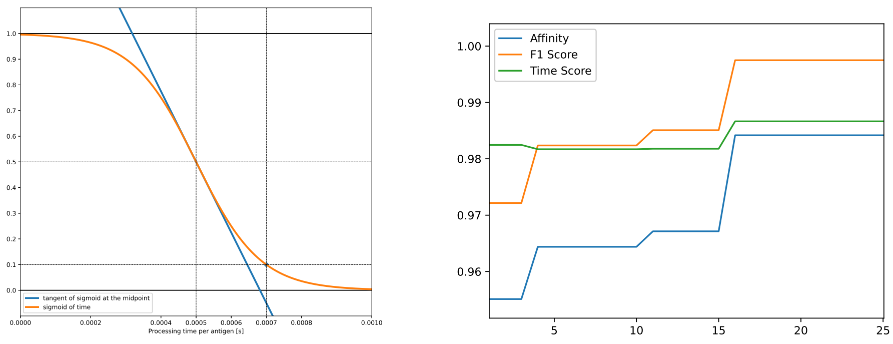

(a) Detectors representation as circumferences (b) Detectors representation by self space

Figure 3.7: Two equivalent representations of detectors. (a) represents the detectors directly as circumferences where points residing inside said circumferences are considered anomalous. (b) illustrates the opposite, i.e., normal space where points inside it are considered normal. In both cases, points in orange are anomalies that were misclassified as normal, and blue points are normal points misclassified as anomalies.

any case, the 200 training normal occurrences were never used to evaluate the performance of any algorithm, only the remaining 800 test data points were used.

For NSA, with 100 detectors and a threshold distance of 0.1, the model could achieve an F_1 score of 0.93 with 6 false positives and 21 false negatives. As the dataset is two dimensional, the detectors themselves can be visualized, and the false positives can be visualized as shown in Figure 3.7, where a known problem of NSA becomes visible, which is the problem of non-self gaps: areas that are uncovered by any detector but which represent anomalous space, leading to false negatives. All false positives are data points that were too close to the normal-anomaly boundary circle and were not included in the training. Some shortcomings are identified, but the algorithm is validated.

Regarding CSA, NSA was optimized regarding the two main parameters, as described in Section 3.2.4. Before using the algorithm, the customized sigmoid function previously described in Equation 3.7 needs to be tuned to a goal. As the processing time per antigen, the midpoint was established as 0.5 ms/antigen, and the sigmoid was tuned such that at 0.7 ms the score would be 0.1, leading to a sigmoid with $slope = 2750$ and the $midpoint = 0.5 \times 10^{-3}$ (Figure 3.8a). This can be interpreted as defining 0.5 ms as a soft limit and 0.7 ms as a hard limit. As the affinity of each antibody is calculated by factoring in three factors that are in the range of 0 and 1, we can conveniently plot the algorithm's progress across generations in a single plot as shown in Figure 3.8b. In this particular case, the best parameters found for NSA were 79 detectors with a threshold distance of approximately 0.168 for an affinity of about 0.984 resulting from an F_1 score of 0.9975 and just under a 0.1 ms per antigen processing time. Because CSA has to execute NSA



(a) Customized sigmoid function.

(b) Affinity and scores over generations.

Figure 3.8: Customized sigmoid function used to normalize time scores (a) and evolution of both F_1 score and time score as well as the resulting affinity over multiple generations.

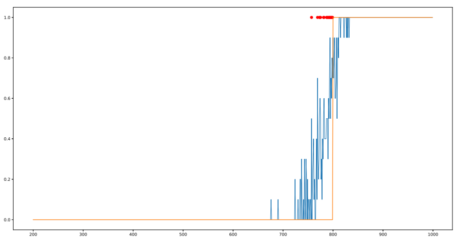
for all points every time a new antibody is produced, the algorithm is slow, taking approximately 3 minutes using only a population size of 10 with 4 clones and 3 antibody additions per iteration, with a max iteration limit of 25. However, the NSA resulting from this implementation was better than that of the previous paragraph. Nonetheless, after full implementation, the applicability for anomaly detection in real time for CSA could not be found.

For DCA, the implementation was executed with 100 cells, with each antigen being captured by 10 random unique cells; the MCAV threshold was set to 0.4 and the migration threshold was set to 1.0. Applying the algorithm in the dataset directly resulted in a F_1 score of 0.49 with 20 false positives and 105 false negatives, despite using danger and safe functions that model the data's distribution perfectly. This is because DCA is designed to be implemented in time series by leveraging the time locality of data points, with the assumption that outliers do not occur in isolation. By ordering the dataset, such that all normal points have lower indexes, DCA achieves an F_1 score of 0.954 with 19 false positives. Creating 4 partitions of anomalies, each with 50 consecutive outliers, DCA achieves an F_1 score of 0.90. These two last results are shown in Figure 3.9. The only uncertainties come from transition areas, where DCA's "sliding window" effect is noticeable.

3.2.7.2 Wisconsin Breast Cancer Diagnosis Dataset

In this scenario, only DCA and NSA will be compared. For each algorithm, all 29 datasets of PCA components were used, and the original, normalized version, for a total of 30 datasets used on each algorithm.

First, DCA needs to be revalidated for the signal functions (danger, safe, and PAMP) described in Section 3.2.6 as these are not the same used in the previous dataset. The normalized version of the dataset is fully ordered, and the signals that result from each antigen are plotted, as well as the final classification results. Figure 3.10 shows all three signals and the MCAV, the same parameters


(a) DCA applied to fully ordered data.

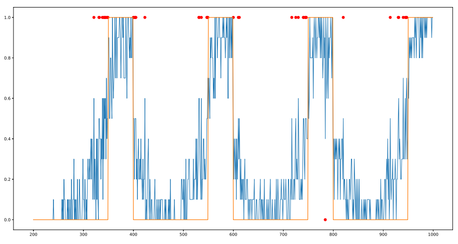

(b) DCA applied to 4 partitions of ordered data.

Figure 3.9: DCA results for both fully ordered (left) and partitioned (right) datasets. The orange line represents the true classification, the blue plot is the MCAV of each antigen, with higher values implying danger and lower values implying safety. The red points represent points that were misclassified, with their position being the classification that DCA attributed to them.

were used as with the first dataset (random circle), and the F_1 score in this instance was 0.986 with 6 false positives and no false negatives, and thus, the algorithm is considered validated.

For each PCA reduced dataset, DCA was adapted for the new correlation coefficients and means, but the parameters remained the same. In NSA the parameters were also the same. Figure 3.11 reveals that DCA is faster than NSA, but when normalized, both algorithms seem to follow a $O(n)$ progression (n being the number of features of each data points). When analyzing the scores of Figure 3.12, NSA seems to fall drastically as the number of features increase, while DCA remains effective. The disproportion between precision (near 1) and recall (near 0) of NSA are signs that the algorithm is evaluating all points as negative (as normal) leading to no false positives, but many false negatives.

In the case of data dimensions increase, it stands to reason that the same number of detectors does not cover the same amount of space, relative to the full problem space. As an attempt to find useful values for NSA, another test was made where the number of detectors increases with each added dimension, by doubling the previous number of detectors. This lead to an increase in recall for the smaller dimension, but the algorithm still declines greatly for both recall and precision values with dimension increase, although now, the processing time per antigen increases exponentially as well. Thus, NSA scales poorly with the number of dimensions. With this in mind, DCA seems to hold potential as an anomaly detection algorithm and will be pursued in the following chapters.

Further results not shown in this section can be found in Appendix C

3.2.8 Limitations

This study was a superficial study to gain insight into each algorithm's potential as an online anomaly detection technique. However, it is important to note the limitations of the work developed in this chapter:

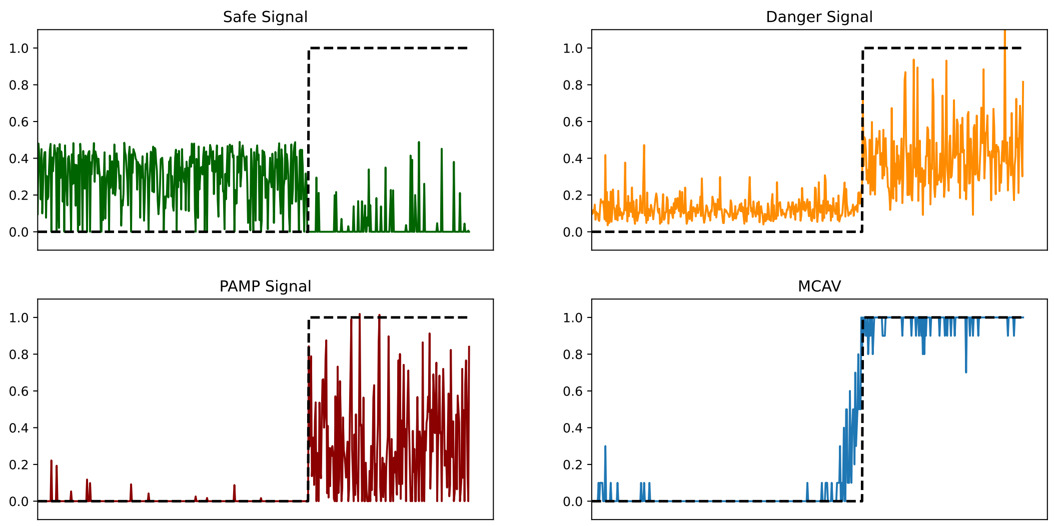


Figure 3.10: All three used signals (inflammation isn't used) and the resulting MCAV. The black dashed line represents the original label from the dataset (0 or 1). All signals were scaled up to double their original value for better visualization.

- AIN was not analyzed other than conceptually, i.e., it was not implemented and fully understood before being discarded;
- CSA was not implemented as an anomaly detection algorithm, as a way to do this was not easily achieved without merging CSA with other algorithms;
- The used datasets are fairly simple and are not time series, which leads to a lack of understanding on how these algorithms scale to real use cases, namely in real-time scenarios;
- Optimized versions of these algorithms exist, which were not explored.

3.3 Summary

In this chapter the main four families of AIS were introduced, namely, those based on: self non-self theory, clonal selection theory, immune network theory, and danger theory. A simplistic implementation of one algorithm from each family was implemented and their applicability for anomaly detection was evaluated:

- NSA requires no knowledge or modeling of the problem, being conceptually simple which makes it easy to implement. The intuition behind T cells (detectors) allows visualization of the detectors themselves and, consequentially, of the "normal space". However, the algorithm was found to scale poorly with data dimension, and detectors very often overlap and leave gaps between them that fail to detect anomalies.

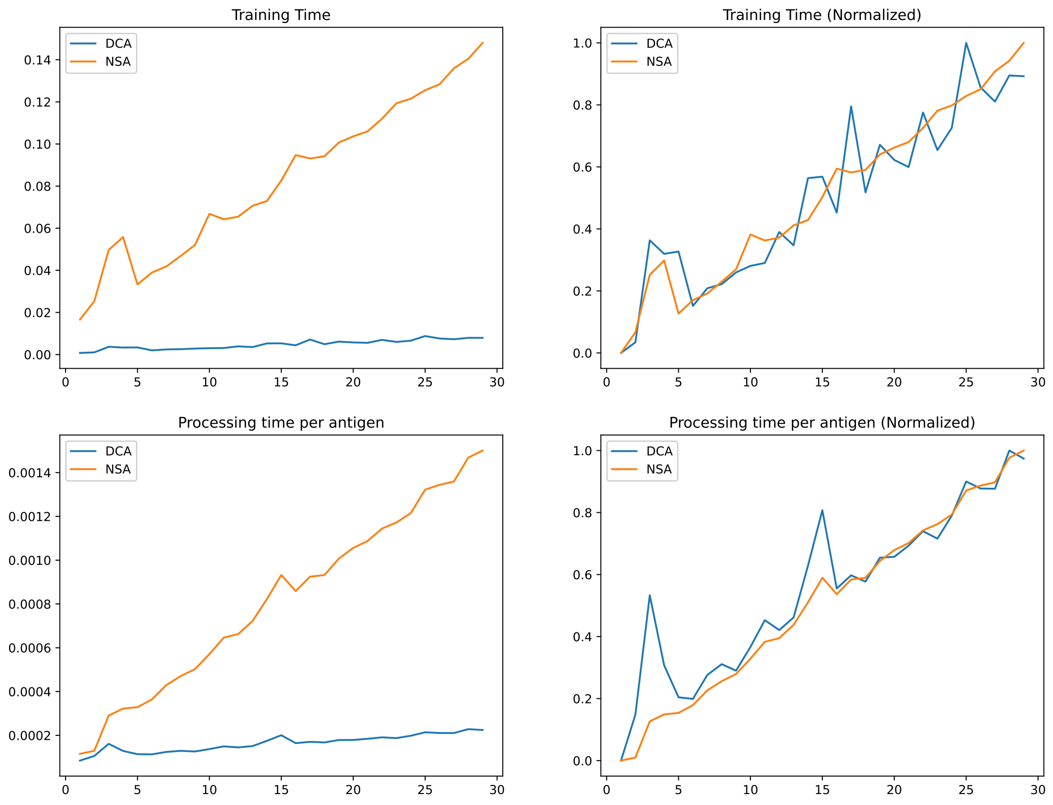


Figure 3.11: Plots of times (y-axes) over each dataset with different numbers of features (x-axes). The left plots show the original time values, while the right plots are normalized linearly between 0 and 1.

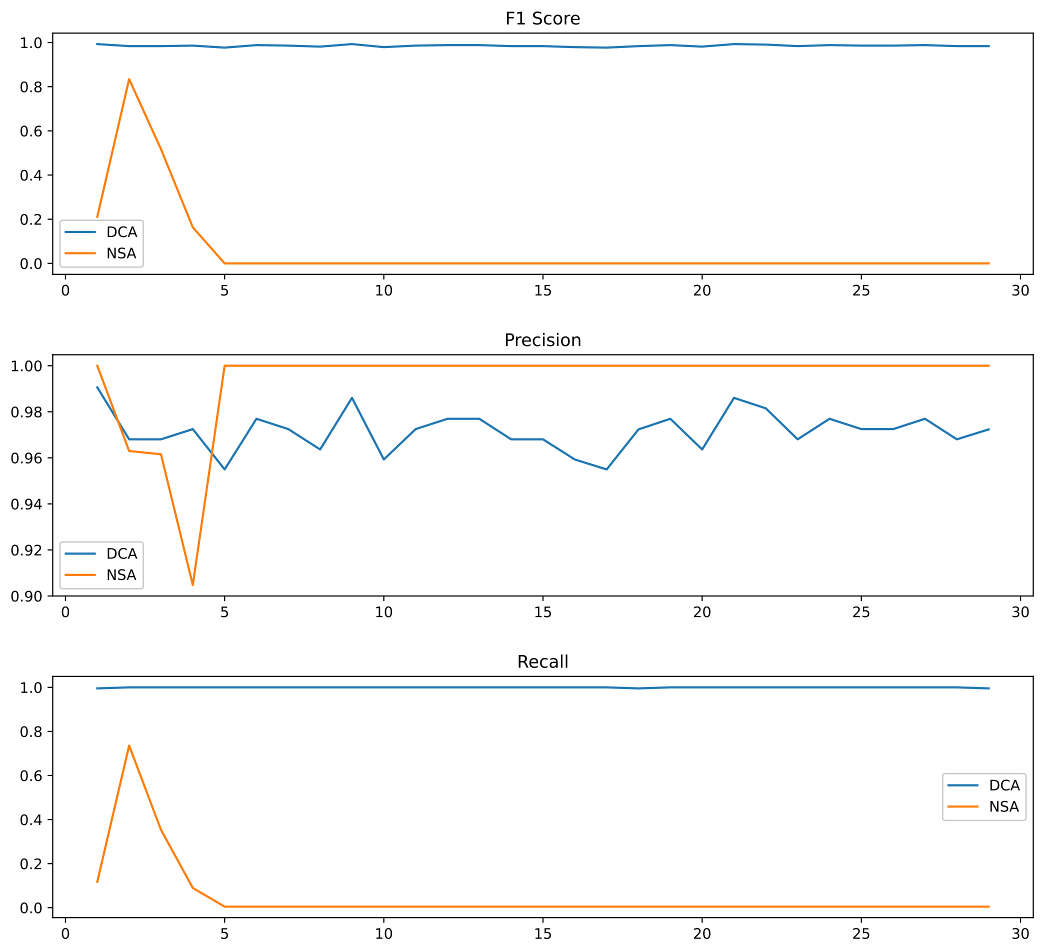


Figure 3.12: Plots of precision, recall and F1 scores (y-axes) over each dataset with different numbers of features (x-axes).

- CSA was only shortly explored, as its application in anomaly detection was not clear. The algorithm succeeded in tuning NSA's parameters, but the problem domain was simple and the execution time of NSA in the second dataset made it prohibitive to attempt CSA in that context.
- AIN, as a networked population algorithm was deemed unintuitive in its implementation, too complex, and, as primarily a clustering algorithm, was left aside. This does not mean that the algorithm does not apply to anomaly detection.
- DCA is relatively lightweight in what concerns execution time. Its scalability is also reasonable, maintaining high metrics for increasing data dimensions. Two identified drawbacks from this algorithm are that extensive modeling of the problem was required, and the algorithm currently needs for all cells that collect a given data point to mature before classifying said data point, which leads the algorithm as it is to be unfit for online anomaly detection.

Chapter 4

Improving DCA for Online Anomaly Detection

In the previous chapter, two main shortcomings were identified for the application of DCA for online anomaly detection:

1. The voting mechanism implies that, in the worst-case scenario, all data has to be processed before classification, which is not a problem in offline detection.

2. The modeling of the algorithm is extremely coupled to the problem at hand. This means that a model of DCA in a problem is not transferable to another problem, and significant effort is required in each problem.

Section 4.1 explains how a modular structure was implemented to make experimentation in the upcoming sections easier. In Section 4.2, a procedure of adapting DCA to function as an online anomaly detection algorithm is proposed. In Section 4.3 fusing DCA with unsupervised techniques to benefit from more data-oriented solutions is proposed.

4.1 Modular Dendritic Cell Algorithm

In this section, a different architecture for DCA design was implemented. The motivation behind this is that different alterations to the algorithm would inevitably lead to great effort in reprogramming the implementation. Thus, a modular architecture was implemented as shown in Figure 4.1. This allows for more experimentation, without the need to adapt the entire software structure to accommodate changes. This includes the following modules:

- **Interface**: to keep the API consistent, the interface module implements the same functions, even if the algorithm changes. It is capable of receiving antigens and outputting their classification (if available). Internally, it sends the received antigens to the signal extractor and the antigen repertoire.

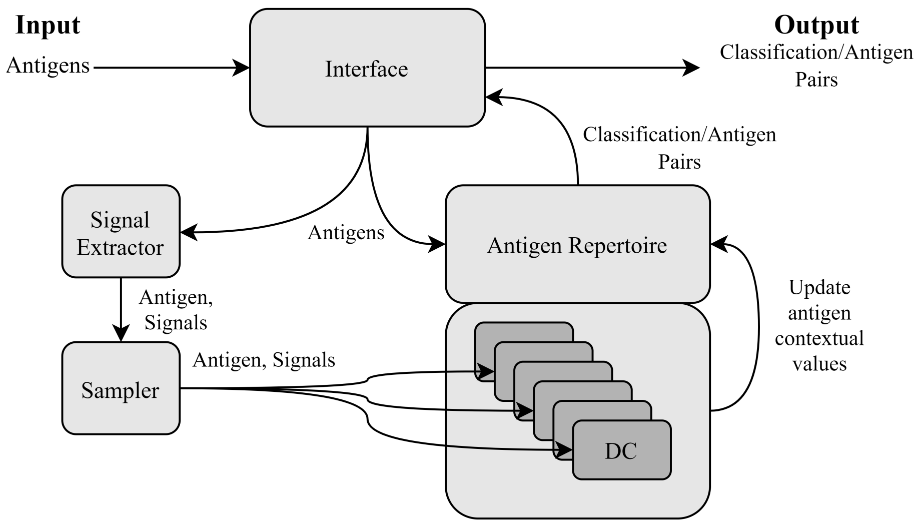


Figure 4.1: Modular DCA model from the perspective of data flow.

- **Signal Extractor**: the way signals are inferred from data. It is capable of receiving antigens and extracting safe, danger, PAMP and inflammation signals from them. It outputs the inferred signals as well as the antigen it received, sending it to the sampler.

- **Sampler**: how the antigens will be distributed among the DC population. It receives a single antigen-signals pair and decides which DCs receive it, if they get copied, or any other intermediate processing procedure.

- **DC Population**: the DC population collect antigens, storing them locally. Any DC has access to the antigens it collected and can change them in the repertoire.

- **Antigen Repertoire**: where the data to be classified is stored. It can function as a passive database or can be implemented to have behavior, such as changing the antigens it collects in some way.

As long as the previous description of each module is implemented, each module can be replaced and tested with different approaches.

4.2 Adapting Dendritic Cell Algorithm for Online Anomaly Detection

In this section, an adaptation of DCA into a more suitable online anomaly detection algorithm is proposed. To refer to this new implementation, let CDCA refer to the online adapted version of DCA.

4.2.1 Deterministic Dendritic Cell Algorithm

This adaptation is partially based on the work of J. Greensmith et al. [69], after the original DCA, which attempted to implement a version of DCA with fewer parameters, that the authors called deterministic Dendritic Cell Algorithm (dDCA) [69]. dDCA was evaluated in its aptitude for intrusion detection in industrial scenarios, by Pinto et al. [44]. These are the main differences between dDCA as described in [69] when compared to the original DCA as implemented in the previous chapter:

- The exposure of antigens to cells is done as a sliding window of cells, rather than randomly picking n cells
- The signals were reduced to only safe and danger signals, and the weights were set as predetermined, rather than as a user parameter.
- A new classification metric is proposed, that takes into account all signals in the data, but this is only possible if the signals are known beforehand.

The signals in dDCA are calculated as indicated in Equation 4.1 (compare with Equation 3.10), where D_i and S_i stand for danger and safe signals associated with the ith collected antigen, respectively.

$$\begin{bmatrix} csm_i \\ k_i \end{bmatrix} = \begin{bmatrix} csm_{i-1} \\ k_{i-1} \end{bmatrix} + \begin{bmatrix} D_i + S_i \\ D_i - 2S_i \end{bmatrix} \tag{4.1}$$

4.2.2 Online Adaptation

The main impediment for online detection is the fact that the MCAV assumes all DCs already matured. However, this dissertation's authors notice that it is possible to classify a point before all DCs collect it. Thus, it is proposed that each time a new antigen is collected, the signals of each cell are updated and an intermediate MCAV value is calculated, under the assumption that the antigen has already been collected by all cells that are to collect it. Figure 4.2 summarily represents how this is achieved: each antigen will be stored as a data structure, with three possible states: "Inconclusive", "Dangerous" and "Safe". The "Inconclusive" state is undesirable as it means that for some iterations, the data point has no classification, however, this approach allows "Dangerous" points to be identified much sooner when compared to "Safe" points that still have to wait for all cells that collected the antigen to mature. The effect of the new "inconclusive" state can be observed in Figure 4.3, where the same instance of CDCA is executed and the progress is captured in four different moments in time.

To further validate that CDCA works as intended, it was applied to the same dataset as the previous version in Section 3.2.7.2. An F_1 score of 0.945 was obtained, considering all inconclusive points to be false positives or false negatives. This implementation, despite being adapted from dDCA, still includes PAMP signals and Inflammation signals, but a null weight can be given to them if desired.

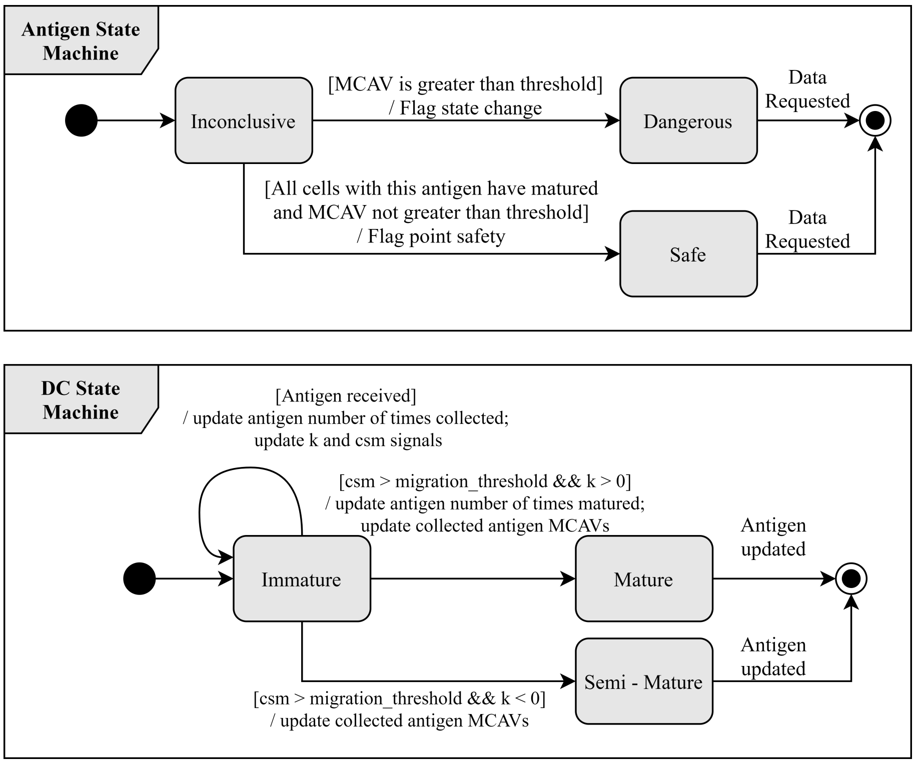

Antigen State Machine
Inconclusive
[MCAV is greater than threshold] / Flag state change
Dangerous
Data Requested
[All cells with this antigen have matured and MCAV not greater than threshold] / Flag point safety
Safe
Data Requested
DC State Machine
[Antigen received] / update antigen number of times collected; update k and csm signals
[csm > migration_threshold && k > 0] / update antigen number of times matured; update collected antigen MCAVs
Immature
Mature
Antigen updated
Semi - Mature
Antigen updated
[csm > migration_threshold && k < 0] / update collected antigen MCAVs

Figure 4.2: Finite state diagram of each DC and antigen. Antigens are stored until classified, after which their classification is flagged and the antigen is erased.

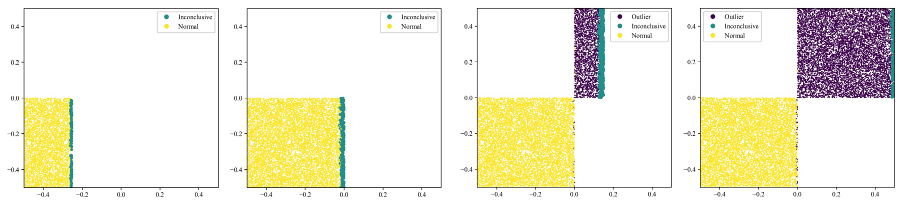


Figure 4.3: The same execution of CDCA over four different moments in time (the points are inputted to the algorithm from left to right). The lifespan of each cell was increased significantly to allow an easy visualization of inconclusive data points (shown in green). Normal points are shown in yellow, with purple points representing anomalies.

In the modular approach presented in Section 4.1, this implied switching the antigen repertoire to store intermediate values for MCAV computation and changing the DC population module to incrementally update the MCAV every time it collects antigens and matures, rather than letting this procedure be done at the end of the algorithm's execution.

With CDCA being suitable for online anomaly detection, the following section explains how it was attempted to be made easier to model.

4.3 Adapting Dendritic Cell Algorithm for Unsupervised Anomaly Detection

Regarding the problem of modeling, the bulk of the effort is in deciding the signals. The remainder of the algorithm's performance is relative to parameter tuning and requires no expert knowledge of the problem domain. Thus, the extraction of signals is proposed to be based on unsupervised techniques in an attempt to overcome this problem. The chosen unsupervised technique(s) needs to:

- **Functional requirements**:
  - be capable of processing multi-dimensional data;
  - be able to process data online, which also implies that it needs to be able to process previously unseen data;
  - provide real-valued information regarding the data (to be used as signals);
- **Performance requirements**:
  - be scalable in what concerns memory storage;
  - be scalable in what concerns execution time per data instance.

In addition to these requirements, it is beneficial if the technique can be expanded, which would allow for DCA to adapt to new environments (e.g. device changed place, now exhibits new normal behavior which needs to be relearned; new device connects to the factory floor, changing the interactions in the system to a new normal). These requirements were considered with flexibility and real time response in mind (two of the factors for CPS described in Section 2.2.1) additionally to being adaptable to DCA.

In the modular approach presented in Section 4.1, this section is mostly restricted to the signal extractor module, to encapsulate an unsupervised technique as a signal extractor.

4.3.1 Clustering

As previously described in Section 2.2.2.3, clustering can be used for anomaly detection. Using *sklearn* [70] as a reference for potential candidates, the following cluster techniques are considered: 1. K-means 2. Affinity propagation 3. Mean-shift 4. Spectral clustering 5. Ward hierarchical clustering 6. Agglomerative clustering 7. DBSCAN 8. OPTICS 9. Gaussian mixtures 10. BIRCH

Techniques 4 through 8 are all transductive clustering algorithms, meaning they only work under the assumption that all data is accessible, thus, to classify new data, the algorithm needs to run over the entirety of the data again, as if analyzing a new problem. Affinity propagation, Mean-shift, and Gaussian mixtures have poor scaling [70], with affinity propagation having a time complexity of $O(N^2T)$ and memory complexity of $O(N^2)$, and Mean-Shift having $O(N^2)$ time complexity. The mentioned time complexities are in total, and time complexity of $O(N)$ is desirable for online usage (such that each data point will be processed in $O(1)$ time). BIRCH scales relatively poorly to high dimensional data but might be applicable.

4.3.1.1 Adapting K-Means

K-Means attempts to find K clusters that minimize the variance within each cluster. Each cluster is described by a single point - the centroid - in the same space as X which represents the mean of the values that pertain to the cluster, and this centroid is iteratively moved until the difference in how much it moves is below a certain threshold. Classification of future points is done by simply computing which centroid is closer to it. K-Means suffers in irregularly shaped data, whose clusters are not successfully separated by how distant they are to centroids. To apply K-Means to DCA, rather than using the categorical classification it provides, the distance to centroids can be used. The proposed approach uses specific training data to calculate the centroids of the training data (this is the same as the original K means). From this procedure, the centroids are stored for later use. Additionally, for each centroid, the distance to the furthest point belonging to that centroid's cluster is also stored. Then, a parameter $f_{tol} \in (0, +\inf)$ allows tuning how much to increase/decrease the radius of each centroid (higher values should increase true negatives, at the potential cost of false negatives, and vice versa). Algorithm 5 presents how danger and safe signals can be extracted from the collected centroids and radii.

Algorithm 5: K-Means danger and safe extraction algorithm.

```
Input: $x$ data point
$R, C$ sets of centroids and radii, respectively
Output: safe and danger signals

1  $signal_{min} = +\inf$
2  foreach $c_i, s_i$ in $C, R$ do
3      $signal = distance(x, c_i) - r_i$
4      if $signal < signal_{min}$ then
5          $signal_{min} = signal$
6      end
7  end
8  if $signal_{min} < 0$ then
9      $danger = signal_{min}$
10 else
11     $safe = -signal_{min}$
12 end
```

Applying this version of K-Means we get the heat map for danger and safe signals shown in Figures 4.4 and 4.5, for different cluster number and tolerance values, respectively.

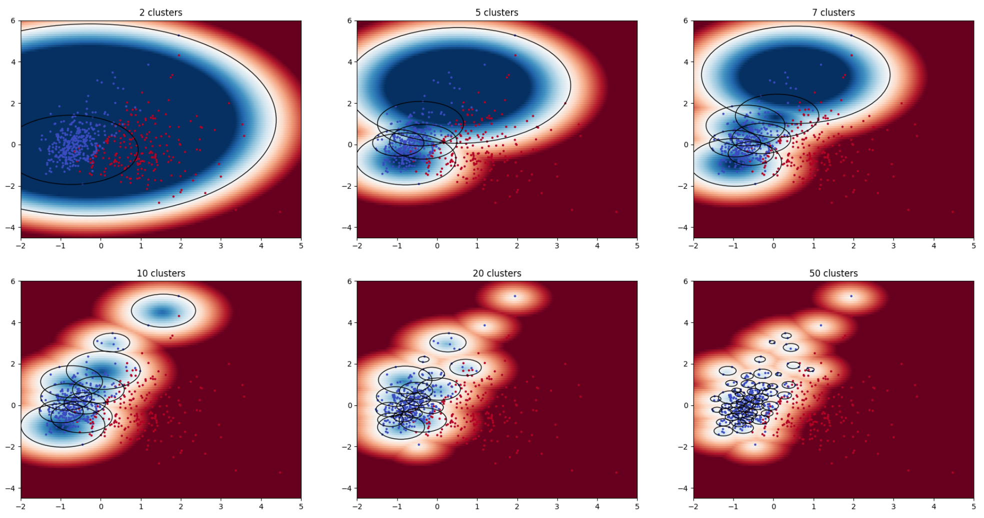


Figure 4.4: Heat maps for different values of k clusters and 1.0 tolerance. Low numbers of clusters create unrealistically large areas leading to a big amount of false negatives. Overly big values of k lead to diminishing returns and, although not the case here, an increase in false positives.

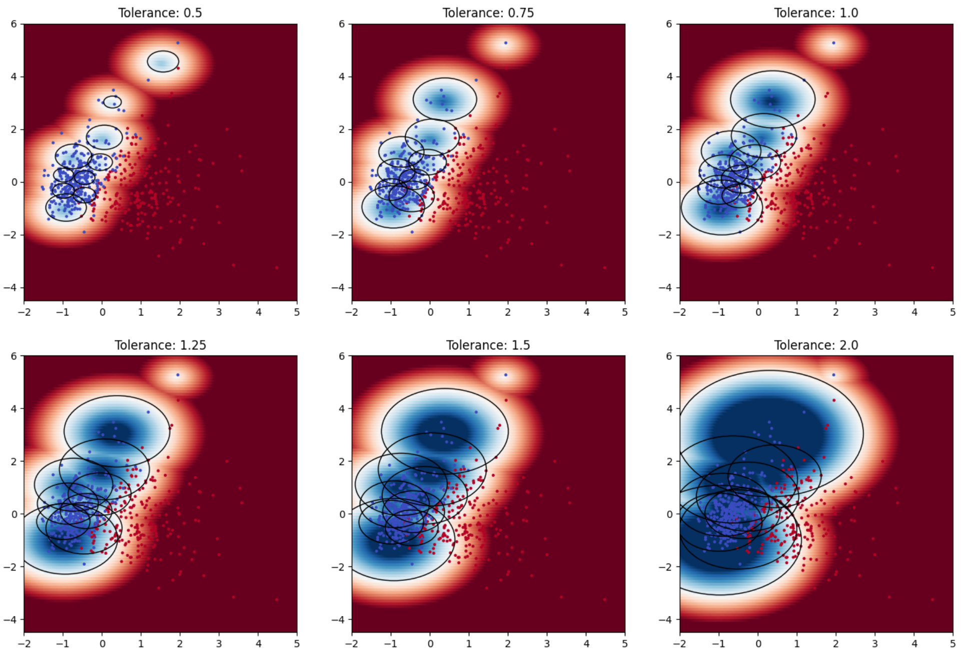


Figure 4.5: Heat maps for different values of *tolerance* clusters for 10 clusters. Low numbers of tolerance increase the chances that an anomaly will lie in the "dangerous" area, but some normal points will also be included. Increasing tolerance will have a non-linear effect, as doubling the radius will increase covered problem space by that increase to the power of data dimension.

Chapter 5

Results and Discussion

In this chapter, two datasets are introduced, in which CDCA is applied with the goal of validating it in two aspects. In Section 5.1 the Skoltech Anomaly Benchmark (SKAB) dataset is introduced, as the dataset to be used to validate the approach in regards to its performance. Then, in Section 5.2 the proposed solution in this dissertation is compared with another DCA approach, validating if its performance is competitive, despite its modeling effort being decreased. All results were obtained using the Python programming language, executing the tests on a computer with the following characteristics:

- Intel(R) Core(TM) i7-6700HQ CPU with 4 physical cores and 8 logical cores
- CPU speed of 2.60 GHz
- 16.0 GB of Random Access Memory
- Windows 10 Operating System

5.1 Skoltech Anomaly Benchmark Dataset

The dataset used is the Skoltech Anomaly Benchmark (SKAB) dataset (dataset can be found in [4]). This dataset was chosen due to the exclusivity to physical process measurements (no network packets, CPU usage, etc.) and the anomalies are caused by faults, rather than by intentional attacks, which goes in accordance with the scope of the thesis (anomaly detection for fault detection). It contains a collection of data points to be used for training (anomaly free collection of points) and anomalies are grouped, i.e., only collective anomalies are present (recall the concept of collective anomalies in Section 2.2.2.2), which aligns with the assumption that anomalies usually occur together in physical processes. It is also inserted in the context of I-IoT and the benchmark has a GNU AGPL v3.0 license, making it accessible. Finally, as a benchmarking solution, the proposed approach can be compared with both current and future solutions using the same benchmark.

Table 5.1: Distribution of SKAB dataset. Number of trials refers to how many times the experiment was done and recorded as a separate file. In case of multiple trials, the number of points and number of anomalous points is the average among trials.

Anomaly Description		**Number of Trials**	**Number of points**	**Number of anomalous points**
Closing of the valve at the outlet of the flow from the pump		4	1078	379.25
Closing of the valve at the inlet of the flow from the pump		16	1135.125	394.3125
Rotor imbalance	Sharp behavior	1	968	398
	Linear behavior	1	1153	410
	Step behavior	1	1147	401
	Dirac delta function behavior	1	1091	346
	Exponential behavior	1	1147	403
Increase in the amount of water in the circuit	Slow increase	1	1145	402
	Sudden increase	1	1326	587
Draining the water from the tank until cavitation		1	1191	451
Two-phase flow supply to the pump inlet (cavitation)		1	1141	401
Water supply of increased temperature		1	1079	339

5.1.1 Assumptions

The assessment of the algorithm's performance is made on the following assumptions:

- An observable physical process exists, whose normal behavior exhibits patterns that are different in the observable data, in respect to the patterns that originate from abnormal behavior;
- The physical process is described by real valued data (e.g. sensor measurements, time between events);
- anomalies occur only sporadically, and thus the implementation can not use *a priori* information regarding the nature of anomalies. This also implies that an implementation can assume the system to start by behaving normally, as the majority of observations ought to be normal;
- anomalies do not occur in isolation, i.e., a single point of data, if anomalous, has other anomalous points of data in close proximity in time.

5.1.2 Description

The testbed used for the production of the SKAB dataset is a water circulation system and is shown in Figure 5.1 (this figure was obtained from [4], with slight alterations in the numbering).

Furthermore, not shown in Figure 5.1, the system contains two vibration sensors, a pressure meter, a flow meter, and a thermocouple. The system communicates through OPC-UA and the data collected from the system is stored in a MySQL database. The data is processed from the database and stored in the `.csv` format files which constitute the dataset. Currently, the anomalies present in the dataset are summarized in Table 5.1 (previous page).

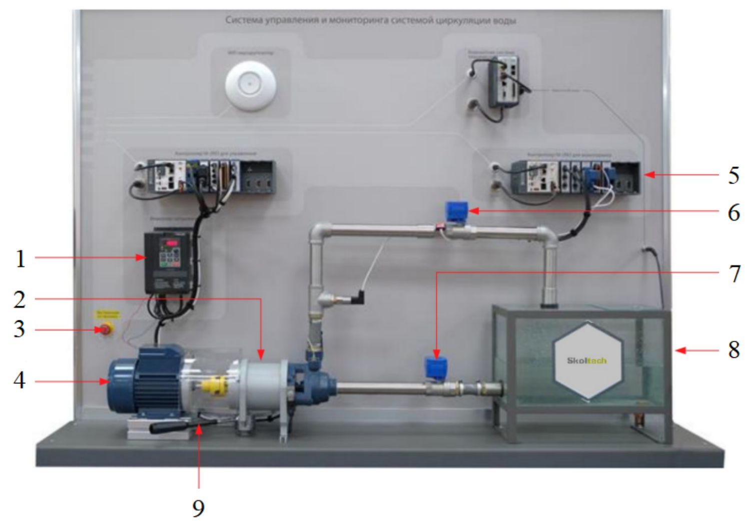


Figure 5.1: Testbed used for the production of the SKAB dataset. Figure obtained from [4]. The numbered components are (from [4]): 1 - inverter; 2 - water pump; 3 - emergency stop button; 4 - electric motor; 5 - compactRIO controller; 6 and 7 - solenoid valves; 8 - water tank; 9 - mechanical lever for shaft misalignment.

5.1.3 Early analysis and preprocessing

To infer if the current features enable anomaly detection, a first step was to visualize the data. As high dimensional data, it is not possible to conveniently plot the entire information, so PCA with two components was used. Figure 5.2 illustrates these results, where there is an indication that more features may be required. Note that PCA leads to loss of information and this analysis is superficial, however, if PCA would reveal clear data separation, then the original data would also be easily separable.

In the following are some attempts at gathering more data as a preprocessing step (note that these features are to be used in addition to the original features, not as a replacement): 1. Moving average; 2. Derivative; 3. Absolute derivative; 4. Moving average of derivative (or absolute derivative); 5. Difference from moving average; 6. Squared distance from moving average.

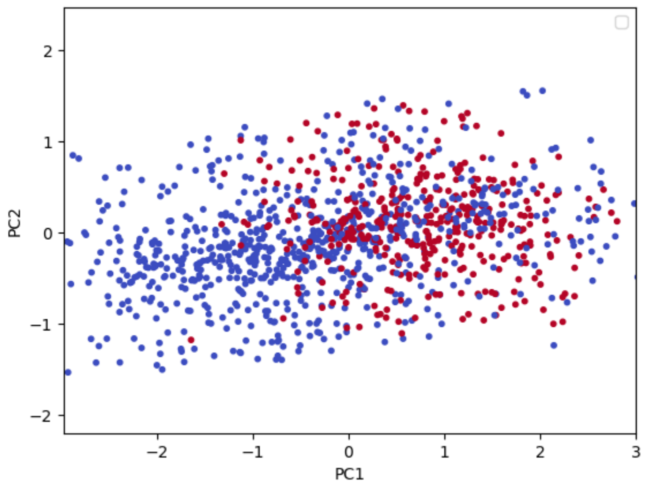


Figure 5.2: Visualization of fault introduced in valve 1 (cf. Figure 5.1). No clear data distribution is perceptible although outliers seem to tend to lie in a cluster in the positive direction of PC1, overlapping with normal values.

Note that applying any of these procedures effectively doubles the number of features (using two of these procedures triples the number of features, and so on). Thus, it is desirable to apply only one or a few more to reduce the complexity. Two of these feature extraction methods were used: moving average and moving average of derivative (signed derivative). The usage of moving average smoothes the otherwise noisy curves, which is particularly relevant for the derivative, when applied to noisy signals of small intervals, the derivative can achieve high values that change significantly from one point to the next, however, the information of whether the signal is increasing or decreasing was considered valuable and thus a moving average of the derivative was used. Given the physical nature of the process, it was considered that values don't drastically change from one point to the next, and thus, metrics of difference from average were considered less informative. Table 5.2 summarizes all original and added features of the dataset. The effect of adding these features will be evaluated in the next section.

Table 5.2: Summary of features used from the SKAB dataset. Features 11-26 were added as a preprocessing step. Features 1 and 3-10 are the original features from the dataset. The anomaly field is not a feature, but was included in this table as it belongs to the dataset and is used for scoring results.

Order	Feature	Description	Data Type
1	datetime	Date and time of day when the value was written to the database	String
2	anomaly	Whether a particular point is anomalous (=1) or not (=0)	Real value
3	Accelerometer1RMS	Vibration acceleration from accelerometer 1 [g]	
4	Accelerometer2RMS	Vibration acceleration from accelerometer 2 [g]	
5	Current	Current in the electric motor [A]	
6	Pressure	Pressure in the fluid loop, at the outlet of the water pump [bar]	
7	Temperature	Temperature of the engine body [ºC]	
8	Thermocouple	Temperature of the fluid in the circulation loop [ºC]	
9	Voltage	Voltage of the electric motor [V]	
10	Volume Flow Rate RMS	Circulation flow rate of the fluid inside the loop [l/min]	
11-18	Moving averages	Moving averages of features 3 through 10 (same units)	
19-26	Moving average of time derivatives	Moving averages of derivatives over time of features 3 through 10 (same units)	

5.1.4 Results and Discussion

Table 5.3: Results for all existing algorithms in SKAB and CDCA as well as the adapted version of K Means. Results for CDCA and K Means are an average of the results of 10 repeated test instances.

Algorithm	**F1**	**FAR [%]**	**MAR [%]**
Perfect detector	1	0	0
Conv-AE	0.79	13.69	17.77
MSET	0.73	20.82	20.08
`CDCA_100_10_20`	0.72	37.95	5.71
`CDCA_100_5_20`	0.72	38.27	5.75
`CDCA_50_10_20`	0.72	38.33	5.80
`CDCA_20_10_20`	0.72	37.74	6.41
`CDCA_100_1_20`	0.72	38.76	5.57
`CDCA_100_10_5`	0.72	29.18	15.24
`CDCA_100_10_20_original`	0.71	28.27	15.61
LSTM-AE	0.68	14.24	35.56
T-squared+Q (PCA)	0.67	13.95	36.32
MSCRED	0.64	13.56	41.16
LSTM	0.64	15.40	39.93
`K Means w/ danger signals`	0.61	25.63	35.76
`CDCA_100_10_20_full`	0.61	19.40	41.11
LSTM-VAE	0.56	9.13	55.03
T-squared	0.56	12.14	52.56
Autoencoder	0.45	7.56	66.57
Isolation forest	0.40	6.86	72.09
Null detector	0	0	100

SKAB [4] includes a scoring system that allows comparing results with the currently implemented solutions, using F_1 score, False Alarm Rate (FAR) and Missed Alarm Rate (MAR). FAR is numerically the same as the FPR (Equation 2.8 in Section 2.2.2.2) and MAR is the complement to recall, i.e., it can be calculated as $1 - Recall$. As SKAB is relatively new, only *ad hoc* implementations from the authors currently populate the public score table.

Six versions of CDCA with K means are presented, with different parameters, denoted as `CDCA_cells_copies_k` such that `cells` refers to the number of cells in the population, `copies` is the number of copies of each antigen created and distributed to the population, and `k` is the number of clusters used for K means. Furthermore, two versions of preprocessing were attempted for `CDCA_100_10_20`, to evaluate the effect that the preprocessing step had on the results, these test instances are denoted in the same way as the previous instance with `_original` appended meaning that no preprocessing was used, and `_full` for all feature alterations initially discarded in Section 5.1.3 being included as well. One final version is the usage of adapted K means' danger and safe signals directly, by classifying points as anomalous if a danger signal is extracted from K means, and classifying them as normal if a safe signal is extracted from K means, using the preprocessed data as was used for the majority of CDCA. Effectively, in this version, we're using adapted K means alone, without the voting mechanism of CDCA. This last test aims to evaluate if CDCA is effectively improving and using the temporal locality of signals, or if the results are a result of the K means signal extractor component, and it could be used by itself.

In all cases, inconclusive points were considered false positives or false negatives, to obtain a conservative result to represent a baseline. Table 5.3 shows all results obtained, averaged over ten instances for each test, ordered by descending F_1 score.

Regarding the preprocessing method, using the original features, with no further feature extraction, has positive results, unlike what was indicated by PCA in Section 5.1.3. Preprocessing the original data with further moving average and moving average of derivative had little effect on F_1 score, having a higher impact on both FAR and MAR, improving the former and worsening the latter. Adding all preprocessing methods degraded the performance of CDCA. This can be attributed to the fact that a higher number of features represents more potential for variation, and the learned centroids will tend to be larger, which, in turn, leads to more points being considered normal. This is corroborated by the increase in MAR and decrease in FAR.

The proposed K means for danger signals without CDCA has significantly worse performance regarding F_1 score, when compared to all instances of CDCA, except for the previous instance with full features. This indicates that CDCA is leveraging the effects of time locality to improve the results yielded by K means.

In what concerns the parameters of CDCA, all alterations consistently lead to similar F_1 scores, with significant differences in FAR and MAR, namely, CDCA seems to be consistently worse than all other algorithms in terms of its FAR, but consistently better in what concerns MAR. In other words, CDCA suffers from too many false positives but excels in capturing all anomalies. The most noticeable impact in changing parameters is in the smaller number of clusters leads to larger clusters, which leads to lower FAR and higher MAR. Thus, the number of clusters regulates the tradeoff between false positives and false negatives, with fewer clusters leading to more true positives at the cost of an increase in false negatives, and more clusters having the opposite effect. Both population and number of copies seem to have little impact on the classifications, with larger populations resulting in slightly less MAR and larger copies sampled resulting in slightly less FAR.

Regarding performance, the main concern is the time of classification, i.e. between receiving an instance of data and classifying it. SKAB does not evaluate the time of execution or classification, thus, only CDCA's timings are presented in Table 5.4 for outlier classification times only, as these timings are most crucial (for times concerning classification as normal, see Appendix E). Currently, CDCA's biggest limitation is its cycle delay in classification: the algorithm as is, can not classify a data point in the same iteration it receives it, in the majority of cases (some exceptions include data points that have a very high danger or safe signals and using very low migration thresholds).

In Table 5.4 the classification cycle delay is understood as the number of iterations between the iteration where CDCA received a point, and the iteration it outputted its classification. Ideally, if CDCA can output a classification the moment it receives a new point, this would imply a classification cycle delay of 0. The population size seems to have little impact on the time it takes to classify anomalies. This is an unexpected result, as it would be logical for bigger populations to lead to longer classification times, as it would be less frequent for any given DC to be exposed to antigens. A possible explanation is that the increase in population, with the same MCAV threshold (recall that MCAV threshold is the threshold of the value of the proportion of DC cells that define an antigen as dangerous to the total population of DC with that antigen), leads to cells migrating faster, but with less consistent results as they are being exposed to antigens much faster, and thus the MCAV might not achieve the threshold significantly faster.

Regarding the number of clusters, a lower number of clusters leads to longer times for anomalies to be detected, which is to be expected, for the same reason as explained before: a smaller number of clusters means bigger clusters, which implies that more points will be considered safe, and hence it will be harder for cells to migrate as dangerous (and those that do, take longer to do so). The last parameter evaluated is the sample copies of antigens from each sample, and a decrease in antigen copies consistently leads to classification delays on average, but with highly irregular results, meaning the average case improves (in what concerns its delay) but the worst-case scenario worsens. One final observation is that, although the results from using all preprocessing methods initially discarded yield worse results in what concerns F_1 score, FAR and MAR, the anomaly classification cycle delay improves, when compared to using the original features. Conversely, using no preprocessing might yield similar metrics in the SKAB score table, but leads to significantly worse delays, likely due to the less distance between outliers and the learned K means centroids (in higher dimensions, Euclidean distance tends to increase in value).

Table 5.4: Performance results for CDCA with multiple parameters. Results are sorted by ascending order of anomalous classification cycle delay means (first column after test instance), smaller values are better.

	Classification Cycle Delay		Classification Delay [ms]	
Test Instance	Mean	Standard Deviation	Mean	Standard Deviation
CDCA_100_1_20	7.684	10.290	1.537	2.097
CDCA_20_10_20	7.758	8.797	2.124	3.242
CDCA_100_10_20	7.818	8.654	2.112	2.288
CDCA_100_10_5	10.081	10.889	2.662	2.854
CDCA_100_10_20_full	11.893	9.298	3.189	2.676
CDCA_100_10_20_original	14.055	15.185	3.801	4.450

Lastly, the raw throughput of this CDCA implementation is analyzed. This metric is used as a way to measure the potential that this algorithm has for analyzing data online if no classification delay cycles are achieved. Table 5.5 shows different parameter combinations and their resulting processing capacity in antigens per second. The most impactful parameter in what concerns raw throughput is the number of copies of each antigen, given that each time the antigen is collected these copies have to be made, distributed among DCs, and their signals propagated, triggering

MCAV updates of other antigens, in case any of these DCs migrate. The population size is irrelevant for the raw throughput, as the impacted DCs are purely determined by the number of copies of the antigen being sampled, and accessing each DC is done in $O(1)$ time, as are the antigens as they are, in this implementation, represented in a hash table. The number of clusters also has reduced impact as calculating the Euclidean distance is only carried out once per antigen, even if the number of copies is higher (the signals themselves are also copied). In this particular scenario, an increase in the number of clusters from 20 to 50 resulted in a 10 % reduction in throughput

The decrease in throughput from the original dataset (8 features) to the fully preprocessed dataset (56 features) is an indicator that this implementation is scalable to problem complexity.

Table 5.5: Results obtained for raw throughput, i.e., without considering classification and classification delay.

Test Instance	**Processing Capacity [Antigen/s]**	**Relative Processing Capacity [%]**
CDCA_100_1_20	14375.849	177.7
CDCA_100_10_20_original	8578.720	106.0
CDCA_1000_10_20	8127.645	100.5
CDCA_100_10_20	8090.653	100.0
CDCA_100_10_5	8024.461	99.2
CDCA_100_10_20_full	7957.220	98.4
CDCA_20_10_20	7548.657	93.3
CDCA_100_10_50	7227.560	89.3
CDCA_100_100_20	1720.764	21.3

In all cases, the fact that the algorithm takes more than one iteration to classify the points is a significant limitation that renders it inapt for time-critical scenarios, in its current state. Given that in this dataset each data point is separated by approximately 1 second, this implies that most anomaly classifications take, at least, 7 seconds, even though the algorithm requires less than 4 *ms* of execution time (i.e., disregarding the delay between receiving one point and the next). This implementation, as is, can still be useful in an Industry 4.0 scenario due to its scalability. The algorithm also proves to work well in combination with other techniques, easily adapting techniques from other fields. Note that the modeling required for K Means is universal and not tied to this problem. This will be corroborated in the following section.

5.2 M2M using OPC UA Dataset

M2M using OPC UA Dataset [5] is an intrusion detection validation dataset. Although the dataset pertains to intrusion detection and not fault detection, which is the scope of this dissertation, it was, to the best of this dissertation's author's knowledge, the only dataset used to validate a DCA

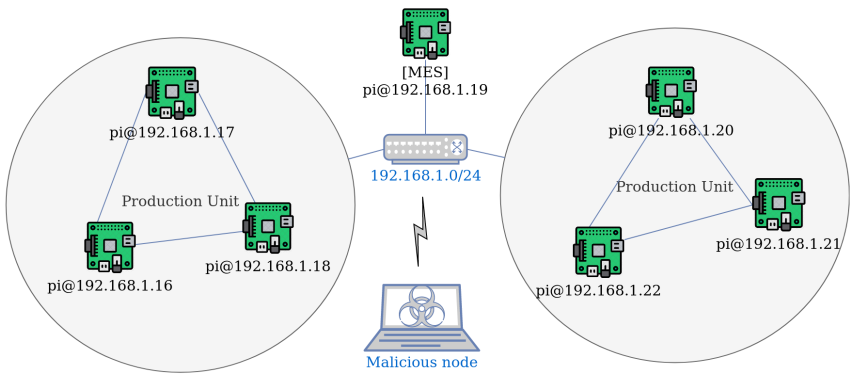


Figure 5.3: Illustration of the testbed used for the generation of the M2M using OPC UA Dataset. Figure obtained from [5].

implemented in an industrial scenario, in [44]. In this section, the main goal is to prove that CDCA with K Means is easily adaptable to different problems, without expert knowledge.

5.2.1 Description

The dataset was obtained from a testbed of multiple connecting devices, as shown in Figure 5.3. The system was attacked by sniffing, injecting, and modifying packets, and then OPC UA packets were captured to form the dataset. Data is labeled with whether it is normal or anomalous, and further, if anomalous, which kind of attack it relates to: Denial of Service, Eavesdropping or Man-in-the-middle, and Impersonation or Spoofing attacks. The dataset's constitution can be found in Appendix F.

5.2.2 Preprocessing

The only preprocessing that was done in the dataset was the removal of all categorical features (except for the label, which will be used for evaluation), and the removal of timestamp features (timestamps of flow start and end).

5.2.3 Results

Applying CDCA with adapted K Means with 100 population size and 10 copies of each data points sampled, and using 20 clusters, the instance was trained using the first 8000 data points, of the total of 107.634 data points, of which 33.567 are normal instances. The AUC of the obtained ROC is 0.978, in comparison to the value of 0.99 obtained in [44]. It is noteworthy that both the AUC of ROC of both approaches is too close to a perfect 1.0, which might indicate that the dataset might not be reliable as a benchmarking dataset for anomaly detection.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

With the objectives that were established, this thesis can be considered partially successful. A new algorithm was developed that expands on the previous work of the Artificial Immune System algorithm - the Dendritic Cell Algorithm - adapting this algorithm to work in real-time as an online anomaly detection tool, leading to the Cursory Dendritic Cell Algorithm, which uses the same principles of the original version, but iteratively updates the classification metric, and keeps track of said metric for individual points. This leads to near real-time classification but opens the possibility of interim "inconclusive" points - points that are awaiting classification. The algorithm was further adapted to be more flexible in its implementation, by first implementing a modular structure that easily allows replacing sub-components, namely how data is collected, how it is preprocessed, how values of abnormality are extracted from it, how those values are mapped to the algorithm's inherent population, and how the population behaves. Consequentially, this modular approach allows faster development.

An adaptation of K Means, an unsupervised clustering algorithm, was implemented to allow easing the implementation process of the Dendritic Cell Algorithm to make it substantially easier to model to any given real valued problem. The hypothesis is that using adapted K Means to extract values of abnormality, both algorithms can help each other: DCA has the means to be more flexible and adapt to different problem contexts, while K Means' classification is made more robust for detecting collective anomalies, which are closely located in time.

The results were validated in the SKAB [4] dataset, which was deemed representative of an I-IoT scenario, albeit on a smaller scale. The results revealed some strong points and some shortcomings in the implemented algorithm. Namely, the usage of signals from adapted K Means effectively increases the performance of the algorithm, by leveraging anomalies as phenomena that tend to happen with proximity in time, corroborating the hypothesis that the algorithms benefit each other. The current implementation also outperforms a significant amount of results found in the same benchmark in what concerns the chosen metric. The algorithm was further tested on another dataset [5], with the main focus of corroborating that it is easy to adapt the algorithm to different scenarios, with comparative results to a similar algorithm of DCA. However, shortcomings of the algorithm are also identified:

- Strengths:
  - The algorithm does not require offline implementation, being able to process data *point-wise*;
  - It exhibits competitive performance in what concerns F_1 score when compared to state-of-the-art approaches, such as deep learning approaches;
  - The algorithm was made modular, such that it can be easily adapted to different problems without a full redesign;
  - Signals can come from other data-oriented techniques, and the algorithm can successfully leverage time proximity to achieve better results and improve the capacity of detecting collective anomalies;
  - The algorithm prioritizes anomalies over normal points, classifying the former faster than it does the latter;
  - The processing capacity of the algorithm can achieve values of 14k data points per second, for data with 56 real valued features.
- Weaknesses:
  - Data points are rarely classified in the same instance as they are received by the algorithm, presenting potentially unacceptable delays;
  - Single point anomalies, be it point anomalies or contextual anomalies, are usually not detected;
  - The algorithm has significant false positive rates;
  - The effect of the population size parameter is unclear.

6.2 Contributions

- Comparative work of AIS, mainly pertaining NSA and DCA, in terms of their potential for online anomaly detection;
- Developed a new modular structure to implement DCA in a more decoupled and extensible way;
- Developed Cursory Dendritic Cell Algorithm (CDCA), a version of DCA that is more apt for anomaly detection;
- Adapted K Means as a flexible tool to extract signals from data, attenuating the problem of modelling related to DCA;
- Validated a version that implements CDCA with the developed modular structure, using the adapted K Means as signal extractor, using the SKAB dataset.

6.3 Future Work

Significant work is still left to do in the development of the proposed solution. Firstly, the main drawback of the algorithm as is, is its delay in classification, which needs to be further refined to offer immediate classification. Furthermore, its application in a real test case is critical before it can be validated as a real-time implementation. The association with other scalable techniques for the generation of signals, other than K Means, is also a promising point to explore. Finally, the algorithm was not built with time optimization in mind, giving priority to achieving a first working prototype, and, thus, optimizing the algorithm for performance is also a promising avenue. With the distributed nature of Industry 4.0, another interesting proposal is to investigate the algorithm's potential to leverage not only time locality, but also spacial locality, such that devices can identify their processes as antigens, send them to the cloud, and thus DCA could identify, not only when the anomaly happened, but also where, in a decentralized manner.

Lastly, an article is being written on the developments of this dissertation, which, if approved, will further validate the potential of DCA.

Appendix A

Systematic Review Results

Table A.1: First part of articles that resulted from the queries shown in Table 2.1 applied to the research steps illustrated in Figure 2.1. Remaining results can be found in Table A.2.

Reference	Title
[33]	AMON: an Automaton MONitor for Industrial Cyber-Physical Security
[34]	KingFisher: an Industrial Security Framework based on Variational Autoencoders
[36]	Cross-Layer Anomaly Detection in Industrial Cyber-Physical Systems
[42]	Anomaly Detection and Productivity Analysis for Cyber-Physical Systems in Manufacturing
[71]	Protecting Cyber Physical Production Systems using Anomaly Detection to enable Self-adaptation
[24]	A Geometric Approach to Clustering Based Anomaly Detection for Industrial Applications
[41]	Safety Risk Monitoring of Cyber-Physical Power Systems Based on Ensemble Learning Algorithm
[38]	On the Identification of Decision Boundaries for Anomaly Detection in CPPS
[43]	Context-Sensitive Modeling and Analysis of Cyber-Physical Manufacturing Systems for Anomaly Detection and Diagnosis
[72]	Nowhere to Hide Methodology: Application of Clustering Fault Diagnosis in the Nuclear Power Industry
[32]	A Novel Data Collection Framework for Telemetry and Anomaly Detection in Industrial IoT Systems
[44]	Attack Detection in Cyber-Physical Production Systems using the Deterministic Dendritic Cell Algorithm
[21]	A Comprehensive Technological Survey on the Dependable Self-Management CPS: From Self-Adaptive Architecture to Self-Management Strategies
[73]	A framework for Model-Driven Engineering of resilient software-controlled systems
[74]	A New Concept of Digital Twin Supporting Optimization and Resilience of Factories of the Future

Table A.2: Continuation of Table A.1.

Reference	**Title**
[75]	Artificial intelligence in cyber physical systems
[76]	A scalable specification-agnostic multi-sensor anomaly detection system for IIoT environments
[77]	A simulation-based platform for assessing the impact of cyber threats on smart manufacturing systems
[2]	A Survey of Anomaly Detection in Industrial Wireless Sensor Networks with Critical Water System Infrastructure as a Case Study
[45]	Big Data Summarisation and Relevance Evaluation for Anomaly Detection in Cyber Physical Systems
[78]	Dawn of new machining concepts:: Compensated, intelligent, bioinspired
[79]	Designing Context-Based Services for Resilient Cyber Physical Production Systems
[80]	Detecting cyber-physical attacks in Cyber Manufacturing systems with machine learning methods
[81]	Improving Security in Industrial Internet of Things: A Distributed Intrusion Detection Methodology
[82]	Machine Learning Quorum Decider (MLQD) for Large Scale IoT Deployments
[27]	Non-convex hull based anomaly detection in CPPS
[48]	Outlier Detection in Temporal Spatial Log Data Using Autoencoder for Industry 4.0
[83]	Real-Time Outlier Detection and Bayesian Classification using Incremental Computations for Efficient and Scalable Stream Analytics for IoT for Manufacturing
[49]	Self-Organizing Maps for Anomaly Localization and Predictive Maintenance in Cyber-Physical Production Systems
[50]	Smart-troubleshooting connected devices: Concept, challenges and opportunities
[35]	SVM-Based Dynamic Reconfiguration CPS for Manufacturing System in Industry 4.0

Appendix B

Literature Destribution

Table B.1: Search inputs used for each technology. To all these inputs, "`AND ("anomaly detection" OR "fault detection")`" was appended, which is not shown in this table, as it is common to all technologies.

Technology	**Search Input**
Clustering	("Clustering" OR "Cluster based")
SVM	("SVM" OR "Support Vector Machine" OR "Support Vector Machines")
ANN	("artificial neural network" OR "artificial neural networks")
Deep Learning	("Deep Neural Network" OR "Deep Auto Encoder" OR "Recurrent Neural Network" OR "Convolutional Neural Network" OR "Long Short Term Memory" OR "Generative Adversarial Network" OR "Self Organizing Map")
Decision Trees	("Decision Tree" OR "Random Forest")
Regression	("Gaussian Process Regression" OR "Logistic Regression" OR "Linear Regression" OR "Kernel Regression" OR "fuzzy Regression")
Nearest Neighbors	("Nearest Neighbors" OR "Nearest Neighbor" OR "k Nearest Neighbors" OR "k Nearest Neighbor")
Bayesian Networks	("Naive Bayes" OR "Bayesian Network")
Markov Models	("Markov Model")
Finite Automata	("Finite Automata" OR "Finite State Machine" OR "FSM")
Dempster-Shafer	("Dempster Shafer" OR "Theory of Evidence" OR "Evidence Theory")
AIS	("artificial immune system" OR "artificial immune systems")

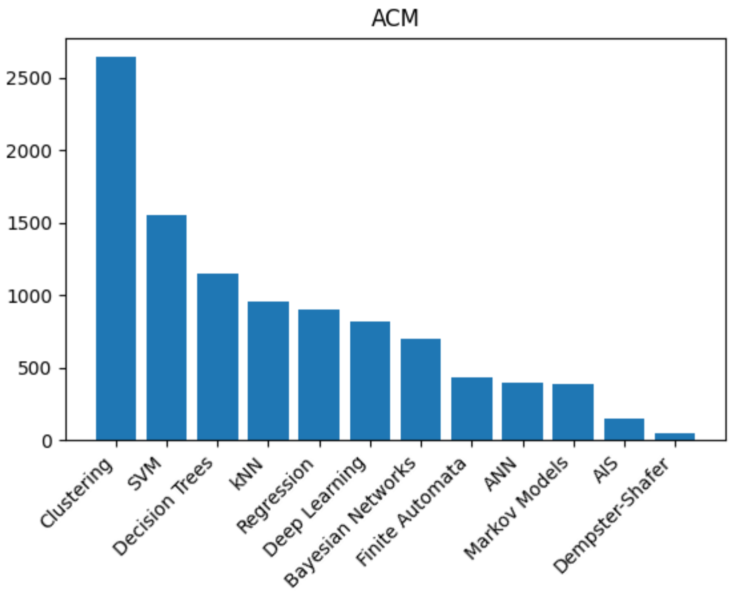


Figure B.1: Obtained results for each query shown in Table B.1 for the ACM database.

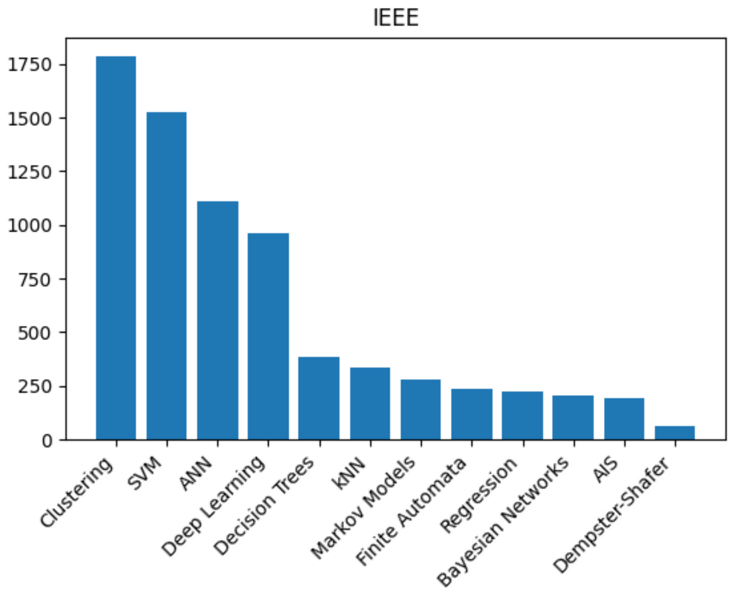


Figure B.2: Obtained results for each query shown in Table B.1 for the IEEE Xplore database.

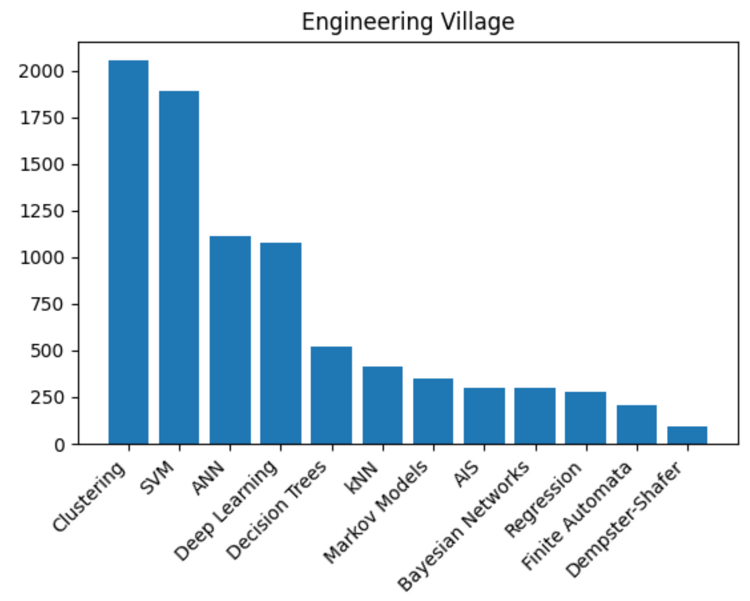


Figure B.3: Obtained results for each query shown in Table B.1 for the Engineering Village/Ei Compendex database.

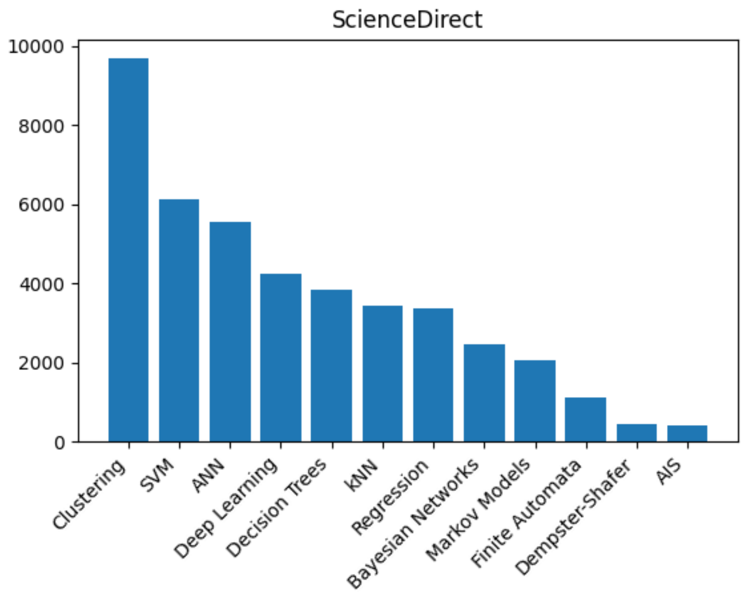


Figure B.4: Obtained results for each query shown in Table B.1 for the ScienceDirect database.

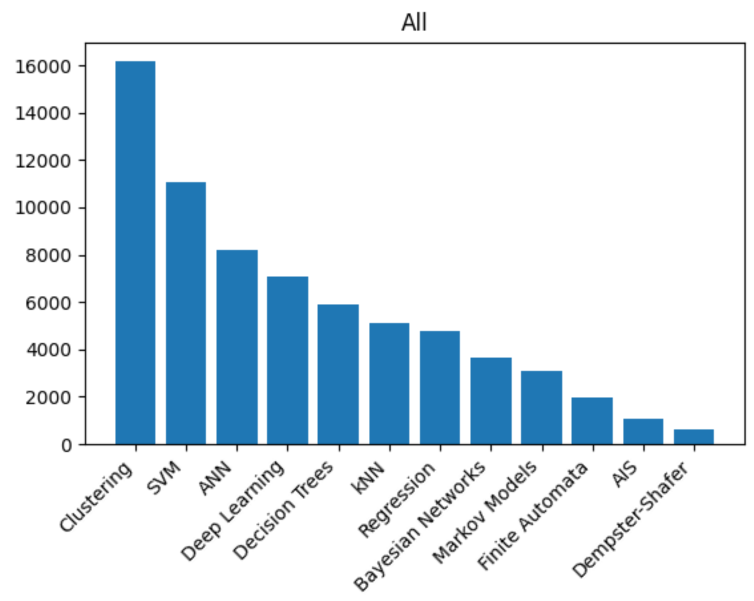


Figure B.5: Sums of all results obtained from previous queries on all databases (shown in Figures B.1 through B.4).

Appendix C

NSA and DCA - Scalability

Table C.1: All timings for both NSA and DCA using static parameters.

Number of Features	NSA			DCA		
	Train Time [ms]	Test Time Mean [µs/antigen]	Test Time Standard Deviation [µs/antigen]	Train Time [ms]	Test Time Mean [µs/antigen]	Test Time Standard Deviation [µs/antigen]
1	15.4206	110.4164179	36.15842196	0.863	78.20191898	39.80410716
2	27.525	146.1882729	95.2403787	2.3802	109.769936	52.2004483
3	45.9291	256.4950959	121.3121175	1.6195	139.673774	76.10419481
4	30.8797	276.4108742	79.26623534	1.8557	105.9375267	45.6309226
5	40.074	386.8230277	101.7576802	1.8481	137.3861407	64.21182807
6	43.2855	444.5304904	83.13679141	3.2272	135.224307	53.32507108
7	57.2805	467.3012793	113.9640066	3.5085	134.2579957	72.15207462
8	53.3295	573.9671642	146.9023116	2.4656	160.3957356	70.63717119
9	52.9023	589.0609808	147.849037	2.6942	155.0624733	72.50998033
10	66.0244	647.7466951	148.8625107	5.8144	158.2266525	73.84350019
11	61.4916	613.6317697	94.55780958	3.9377	142.3579957	52.2316838
12	83.8024	644.928145	116.2694898	3.96	145.3366738	63.0614612
13	74.9727	799.7513859	143.2211761	3.6993	168.4236674	60.54158418
14	84.5941	800.1059701	165.2805933	5.898	163.3884861	68.68373321
15	102.6854	940.0264392	195.5337202	5.7558	184.9151386	68.16204
16	87.4177	852.3498934	158.0440345	4.217	161.6876333	64.14844327
17	88.8748	955.4614072	196.4661175	6.543	179.4646055	67.71293733
18	105.3135	982.0121535	213.4399225	4.7622	178.3187633	71.58413642
19	103.9708	1095.571642	211.4217401	5.1133	196.8066098	78.17688009
20	107.9244	1172.669723	243.0811088	5.3845	205.8891258	79.77440967
21	107.6563	1171.376119	267.0046773	7.6665	201.5648188	76.58415879
22	113.1483	1175.895736	200.1156448	5.7128	201.7899787	75.08656376
23	118.0297	1165.921962	120.4800836	6.0512	188.5042644	55.20345256
24	156.9367	1283.895309	257.7124156	8.571	213.9550107	73.20204322
25	127.9077	1312.912154	169.7027835	7.2064	205.4601279	54.20276196
26	143.3915	1358.764606	171.2863473	9.9758	212.8036247	60.4261722
27	138.5992	1477.208742	197.6399264	6.9152	229.4761194	68.67986641
28	145.1598	1444.865672	172.8795407	7.4708	224.0002132	74.65199271
29	146.0911	1451.658209	108.260966	7.9112	216.2070362	56.64597622

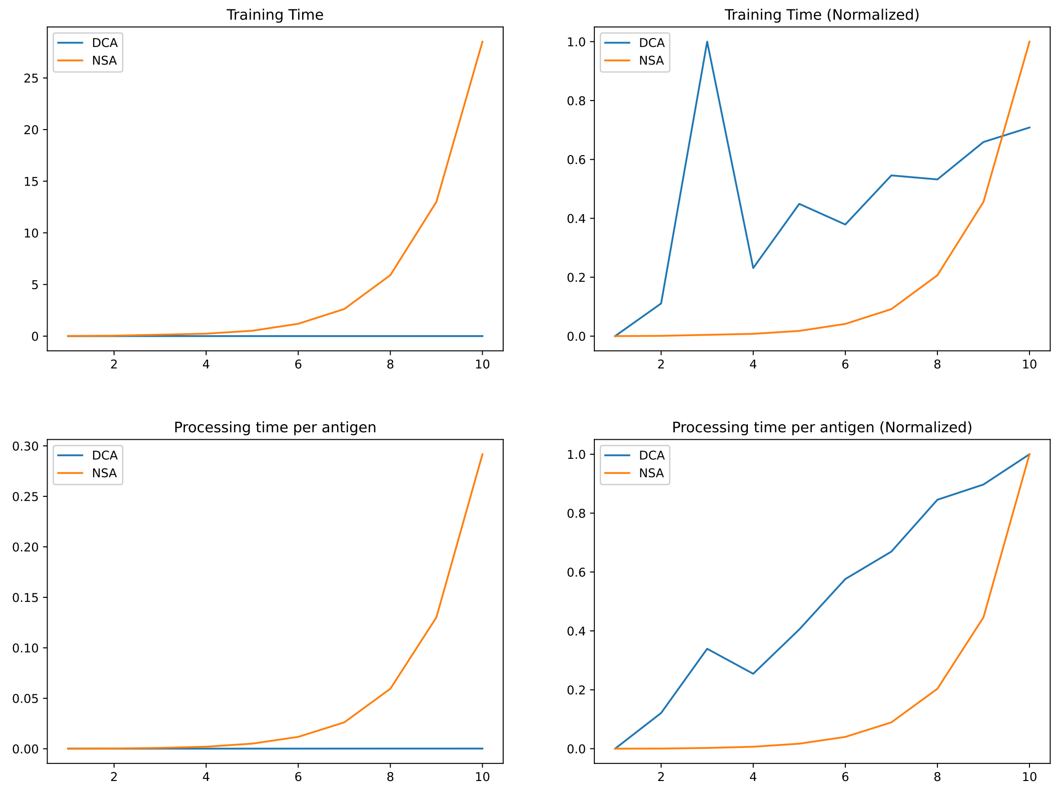


Figure C.1: Plots of times (y-axes) over each dataset with different numbers of features (x-axes). The left plots show the original time values, while the right plots are normalized linearly between 0 and 1. The number of detectors for NSA is doubled every iteration.

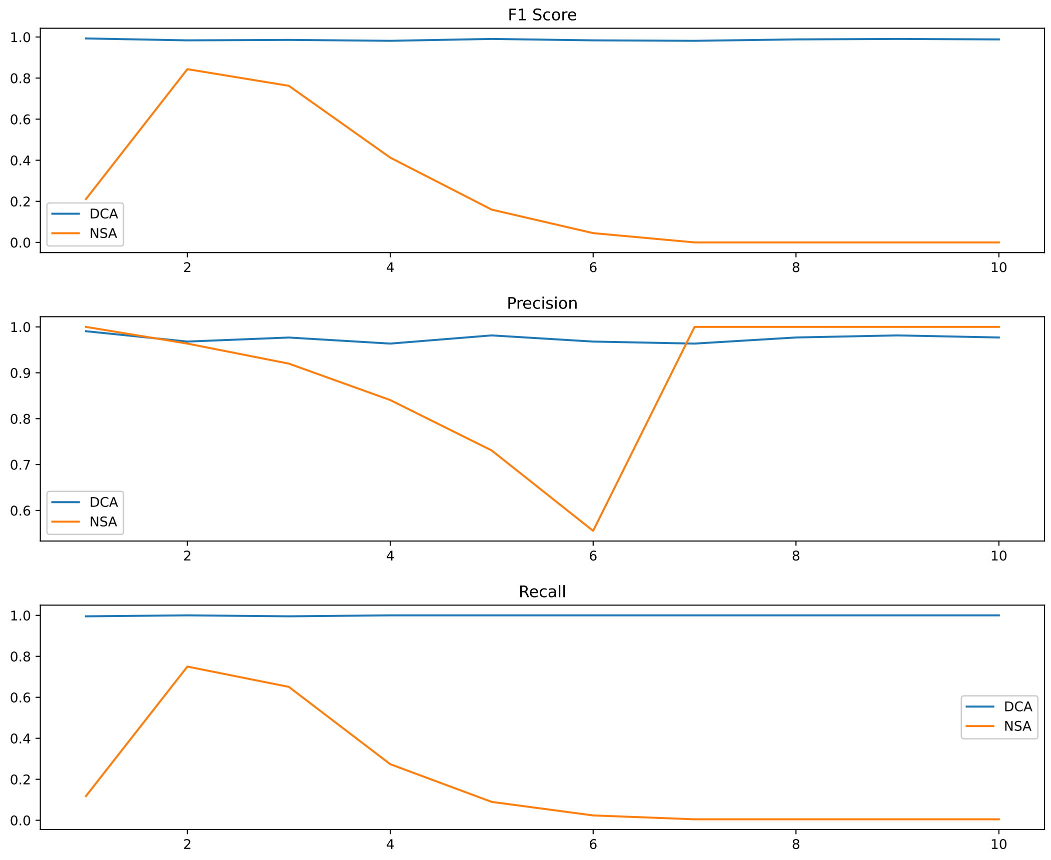


Figure C.2: Plots of precision, recall and F1 scores (y-axes) over each dataset with different numbers of features (x-axes). The number of detectors for NSA is doubled every iteration.

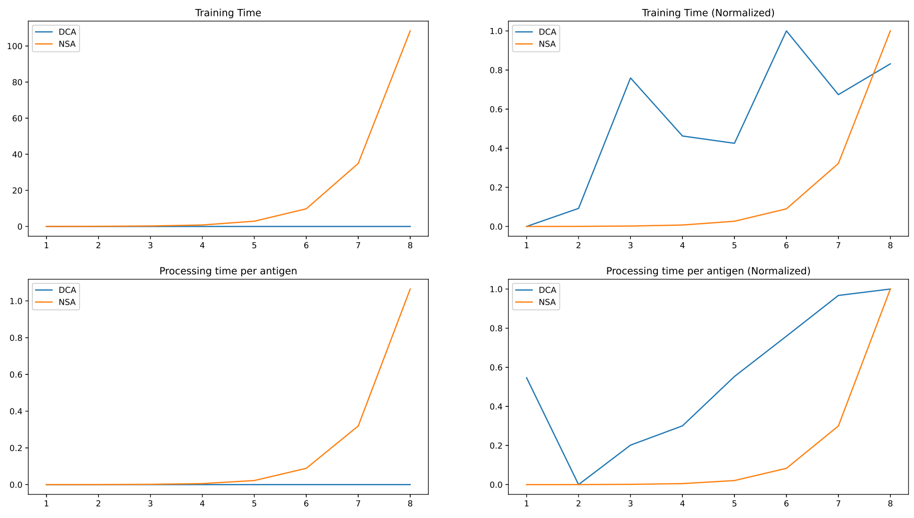


Figure C.3: Plots of times (y-axes) over each dataset with different numbers of features (x-axes). The left plots show the original time values, while the right plots are normalized linearly between 0 and 1. The number of detectors for NSA is tripled every iteration.

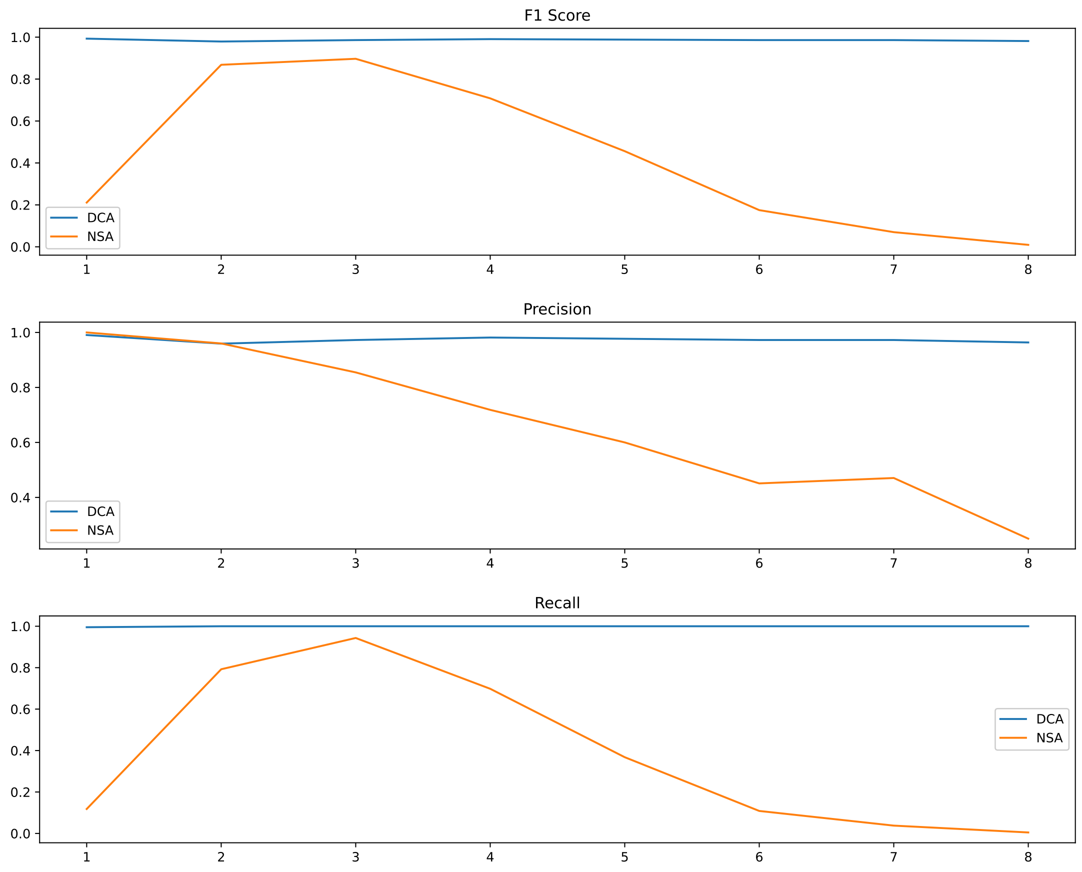


Figure C.4: Plots of precision, recall and F1 scores (y-axes) over each dataset with different numbers of features (x-axes). The number of detectors for NSA is tripled every iteration.

Appendix D

Preprocessing Experiments with SKAB dataset

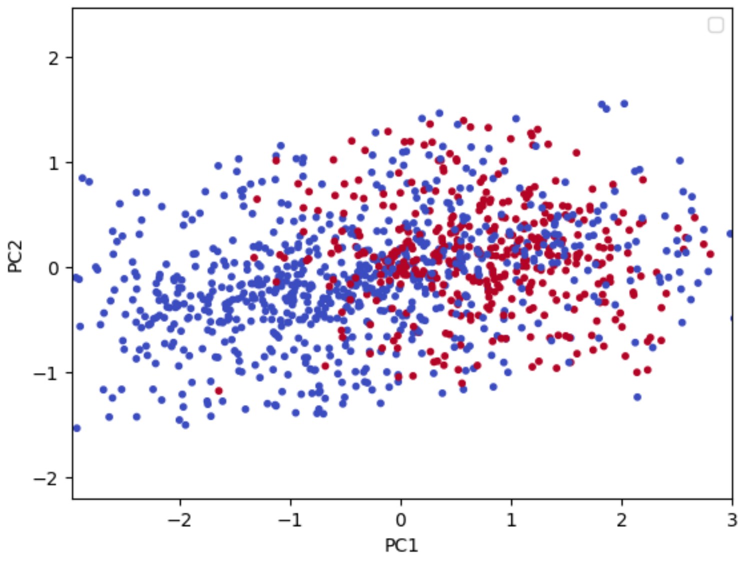


Figure D.1: 2 component PCA applied to the original SKAB dataset.

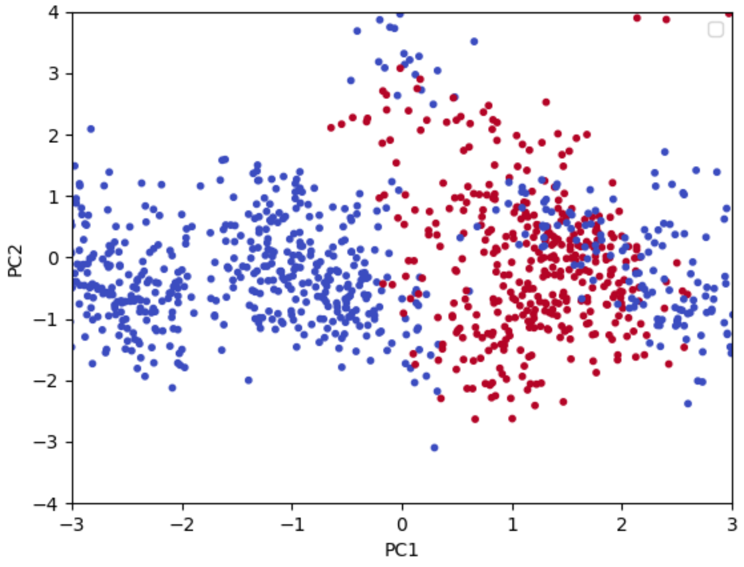


Figure D.2: 2 component PCA applied with 8 additional features computed as rolling average (window size 10).

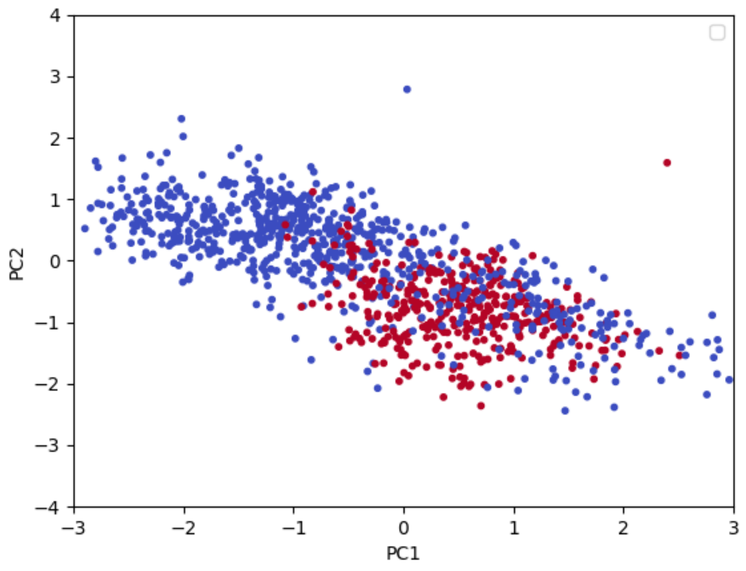


Figure D.3: 2 component PCA applied with 8 additional features computed as rolling average (window size 10) of discrete derivatives.

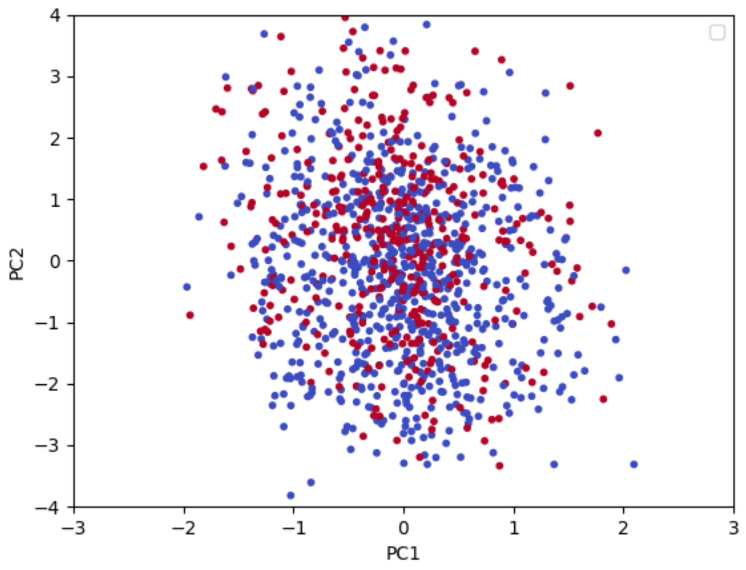


Figure D.4: 2 component PCA applied with 8 additional features computed as discrete derivatives.

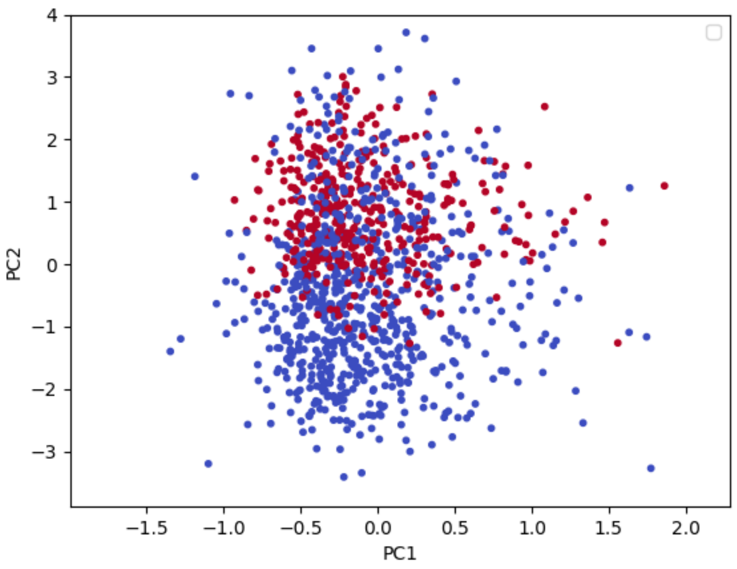


Figure D.5: 2 component PCA applied with 8 additional features computed as the absolute value of the discrete derivatives.

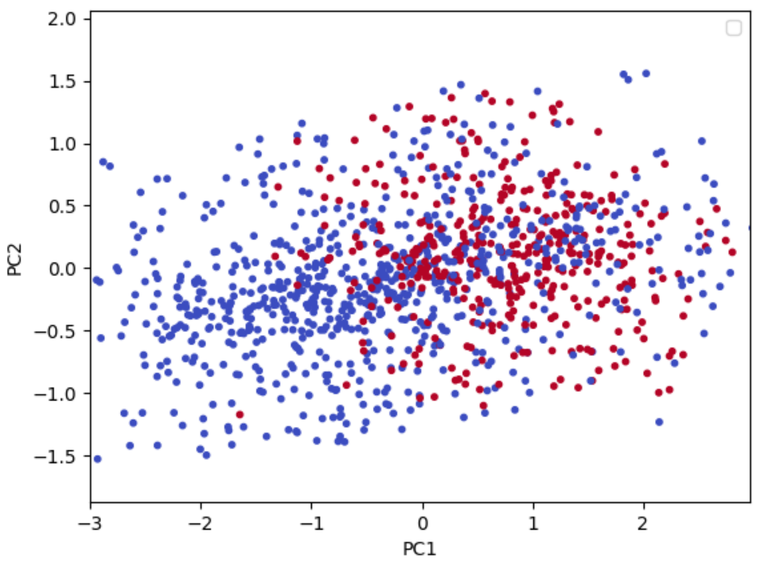


Figure D.6: 2 component PCA applied with 8 additional features computed as the distance from rolling average of window size 10.

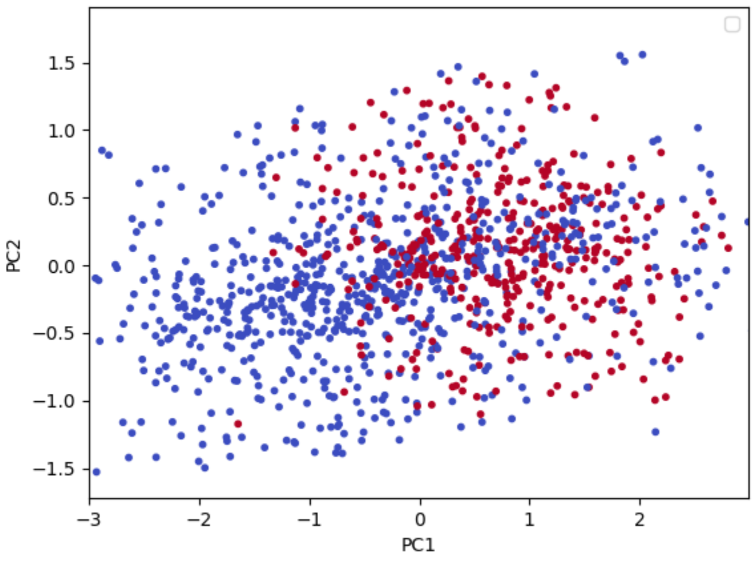


Figure D.7: 2 component PCA applied with 8 additional features computed as the squared distance from rolling average of window size 10.

Appendix E

Performance results for Normal Classification in SKAB dataset

Table E.1: Performance results for CDCA with multiple parameters. Results are sorted by ascending order of normal classification cycle delay means (first column after test instance), smaller values are better.

	Classification Cycle Delay		Classification Delay [ms]	
Test Instance	Mean	Standard Deviation	Mean2	Standard Deviation3
CDCA_100_1_20	19.370	15.148	4.201	3.449
CDCA_100_10_20_full	27.426	7.676	7.340	2.608
CDCA_100_10_5	32.084	12.023	8.694	3.662
CDCA_20_10_20	42.289	13.678	11.402	4.835
CDCA_100_10_20	42.416	13.815	11.754	4.326
CDCA_100_10_20_original	68.022	29.706	19.642	9.982

Appendix F

M2M using OPC UA Dataset Features

Table F.1: M2M using OPC UA Dataset[5]. Remaining features can be found in Table F.2

Feature	**Description**	**Data Type**
src_ip	Source IP address	String
src_port	Source port	
dst_ip	Destination IP address	Integer
dst_port	Destination port	
flags	TCP flag status	
pktTotalCount	Total packet count	
octetTotalCount	Total packet size	
avg_ps	Average packet size	Real Valued
proto	Protocol	String
service	OPC UA service call type	
service_errors	Number of service errors in OPC UA request responses	Integer
status_errors	Number of status errors in OPC UA request responses	
msg_size	OPC UA message transport size	
min_msg_size	Minimum OPC UA message size	

Table F.2: Continuation of Table F.1

Feature	**Description**	**Data Type**
flowStart	Timestamp of flow start	Real Valued
flowEnd	Timestamp of flow end	
flowDuration	Flow duration in seconds	
avg_flowDuration	Average flow duration in seconds	
flowInterval	Time interval between flows in seconds	
count	Number of connections to the same destination host as the current connection in the past two seconds	Integer
srv_count	Number of connections to the same port number as the current connection in the past two seconds	
same_srv_rate	The percentage of connections that were to the same port number, among the connections aggregated in Count	Real Valued
dst_host_same_src_port_rate	The percentage of connections that were to the same source port, among the connections having the same port number	
f_pktTotalCount	Total forward packets count	Integer
f_octetTotalCount	Total forward packets size	
f_flowStart	Timestamp of first forward packet start	Real Valued
f_rate	Rate at which forward packets are transmitted	
b_pktTotalCount	Total backwards packets count	Integer
b_octetTotalCount	Total backwards packets size	
b_flowStart	Timestamp of first backwards packet start	Real Valued
label	Binary label classification	Integer
multi_label	Multi classification labeling	String

References

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009. URL: https://doi.org/10.1145/1541880.1541882, doi:10.1145/1541880.1541882.

[2] Daniel Ramotsoela, Adnan Abu-Mahfouz, and Gerhard Hancke. A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study. *Sensors (Switzerland)*, 18(8):1–24, 2018. doi:10.3390/s18082491.

[3] P. H. Richter. A network theory of the immune system. *European Journal of Immunology*, 5(5):350–354, 1975.

[4] Iurii D. Katser and Vyacheslav O. Kozitsin. Skoltech anomaly benchmark (skab). https://www.kaggle.com/dsv/1693952, 2020. doi:10.34740/KAGGLE/DSV/1693952.

[5] Rui Pinto. M2m using opc ua, 2020. URL: https://dx.doi.org/10.21227/ychv-6c68, doi:10.21227/ychv-6c68.

[6] Kishan G Mehrotra, Chilukuri K Mohan, and HuaMing Huang. *Anomaly Detection Principles and Algorithms*. Springer International Publishing, 1 edition, 2017. doi:10.1007/978-3-319-67526-8.

[7] N. Jazdi. Cyber physical systems in the context of Industry 4.0. *Proceedings of 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2014*, pages 31–33, 2014. doi:10.1109/AQTR.2014.6857843.

[8] H. Xu, W. Yu, D. Griffith, and N. Golmie. A survey on industrial internet of things: A cyber-physical systems perspective. *IEEE Access*, 6:78238–78259, 2018. doi:10.1109/ACCESS.2018.2884906.

[9] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & information systems engineering*, 6(4):239–242, 2014.

[10] Mridula Sharma, Haytham Elmiligi, and Fayez Gebali. *Network Security and Privacy Evaluation Scheme for Cyber Physical Systems (CPS)*, pages 191–217. Springer International Publishing, Cham, 2020. URL: https://doi.org/10.1007/978-3-030-38557-6_9, doi:10.1007/978-3-030-38557-6_9.

[11] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security Privacy*, 9(3):49–51, 2011. doi:10.1109/MSP.2011.67.

[12] H. Sinanović and S. Mrdovic. Analysis of mirai malicious software. In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5, 2017. doi:10.23919/SOFTCOM.2017.8115504.

[13] N. E. Oueslati, H. Mrabet, A. Jemai, and A. Alhomoud. Comparative study of the common cyber-physical attacks in industry 4.0. In *2019 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pages 1–7, 2019. doi:10.1109/IINTEC48298.2019.9112097.

[14] Robert Mitchell and Ing Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys*, 46(4), 2014. doi:10.1145/2542049.

[15] Rozhin Yasaei, Felix Hernandez, and Mohammad Abdullah Al Faruque. IoT-CAD: Context-Aware Adaptive Anomaly Detection in IoT Systems through Sensor Association. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, 2020-Novem, 2020. doi:10.1145/3400302.3415672.

[16] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18, 2015. URL: http://dx.doi.org/10.1016/j.infsof.2015.03.007, doi:10.1016/j.infsof.2015.03.007.

[17] Hansong Xu, Wei Yu, David Griffith, and Nada Golmie. A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective. *IEEE Access*, 6:78238–78259, 2018. doi:10.1109/ACCESS.2018.2884906.

[18] R. Sanz, I. López, J. Bermejo, R. Chinchilla, and R. P. Conde. Self-X: The control within. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 38(1):179–184, 2005. doi:10.3182/20050703-6-cz-1902.01071.

[19] Devarpita Sinha and Rajarshi Roy. Reviewing Cyber-Physical System as a Part of Smart Factory in Industry 4.0. *IEEE Engineering Management Review*, 48(2):103–117, 2020. doi:10.1109/EMR.2020.2992606.

[20] Denise Ratasich, Faiq Khalid, Florian Geissler, Radu Grosu, Muhammad Shafique, and Ezio Bartocci. A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems. *IEEE Access*, 7:13260–13283, 2019. arXiv:1810.06870, doi:10.1109/ACCESS.2019.2891969.

[21] Peng Zhou, Decheng Zuo, Kun Mean Hou, Zhan Zhang, Jian Dong, Jianjin Li, and Haiying Zhou. A comprehensive technological survey on the dependable self-management CPS: From self-adaptive architecture to self-management strategies. *Sensors (Switzerland)*, 19(5), 2019. doi:10.3390/s19051033.

[22] G. Bernieri, M. Conti, and F. Turrin. Evaluation of machine learning algorithms for anomaly detection in industrial networks. In *2019 IEEE International Symposium on Measurements Networking (M N)*, pages 1–6, 2019. doi:10.1109/IWMN.2019.8805036.

[23] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. doi:10.1016/j.patrec.2005.10.010.

[24] Peng Li, Oliver Niggemann, and Barbara Hammer. A geometric approach to clustering based anomaly detection for industrial applications. *Proceedings: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, pages 5345–5352, 2018. doi:10.1109/IECON.2018.8592906.

[25] Marin E. Pamukov, Vladimir K. Poulkov, and Vasil A. Shterev. Negative Selection and Neural Network Based Algorithm for Intrusion Detection in IoT. *2018 41st International Conference on Telecommunications and Signal Processing, TSP 2018*, pages 636–640, 2018. doi:10.1109/TSP.2018.8441338.

[26] Moshe Kravchik and Asaf Shabtai. Detecting cyber attacks in industrial control systems using convolutional neural networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, CPS-SPC '18, page 72–83, New York, NY, USA, 2018. Association for Computing Machinery. URL: https://doi.org/10.1145/3264888.3264896, doi:10.1145/3264888.3264896.

[27] Beibei Li, Yuhao Wu, Jiarui Song, Rongxing Lu, Tao Li, and Liang Zhao. DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics*, 3203(c):1–1, 2020. doi:10.1109/tii.2020.3023430.

[28] Qin Lin, Sicco Verwer, Sridha Adepu, and Aditya Mathur. TABOR: A graphical model-based approach for anomaly detection in industrial control systems. *ASIACCS 2018 - Proceedings of the 2018 ACM Asia Conference on Computer and Communications Security*, pages 525–536, 2018. doi:10.1145/3196494.3196546.

[29] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv*, pages 1–50, 2019. arXiv:1901.03407.

[30] Vyacheslav Tokarev, Alexey Sychugov, and Alexander Anchishkin. Detection of Anomalies in the Information Networks of Industrial Automation Systems Based on Artificial Immune Detectors. *Proceedings - 2019 International Russian Automation Conference, RusAutoCon 2019*, pages 8–12, 2019. doi:10.1109/RUSAUTOCON.2019.8867593.

[31] Nawel Bayar, Saber Darmoul, Sonia Hajri-Gabouj, and Henri Pierreval. Fault detection, diagnosis and recovery using Artificial Immune Systems: A review. *Engineering Applications of Artificial Intelligence*, 46:43–57, 2015. doi:10.1016/j.engappai.2015.08.006.

[32] Fabrizio De Vita, Dario Bruneo, and Sajal K. Das. A novel data collection framework for telemetry and anomaly detection in industrial iot systems. *Proceedings - 5th ACM/IEEE Conference on Internet of Things Design and Implementation, IoTDI 2020*, pages 245–251, 2020. doi:10.1109/IoTDI49375.2020.00032.

[33] Giuseppe Bernieri, Mauro Conti, and Gabriele Pozzan. Amon: An automaton monitor for industrial cyber-physical security. *ACM International Conference Proceeding Series*, 2019. doi:10.1145/3339252.3340521.

[34] Giuseppe Bernieri, Mauro Conti, and Federico Turrin. KingFisher: An industrial security framework based on variational autoencoders. *SenSys-ML 2019 - Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems, Part of SenSys 2019*, pages 7–12, 2019. doi:10.1145/3362743.3362961.

[35] Hyun Jun Shin, Kyoung Woo Cho, and Chang Heon Oh. SVM-Based Dynamic Reconfiguration CPS for Manufacturing System in Industry 4.0. *Wireless Communications and Mobile Computing*, 2018, 2018. doi:10.1155/2018/5795037.

[36] Hunor Sandor, Bela Genge, Piroska Haller, Adrian Vasile Duka, and Bogdan Crainicu. Cross-layer anomaly detection in industrial cyber-physical systems. *2017 25th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2017*, 2017. doi:10.23919/SOFTCOM.2017.8115523.

[37] B. Genge, C. Siaterlis, and G. Karopoulos. Data fusion-base anomay detection in networked critical infrastructures. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–8, 2013. doi:10.1109/DSNW.2013.6615505.

[38] Peng Li, Oliver Niggemann, and Barbara Hammer. On the identification of decision boundaries for anomaly detection in CPPS. *Proceedings of the IEEE International Conference on Industrial Technology*, 2019-February:1311–1316, 2019. doi:10.1109/ICIT.2019.8754997.

[39] Peng Li and Oliver Niggemann. Non-convex hull based anomaly detection in CPPS. *Engineering Applications of Artificial Intelligence*, 87(August 2019):103301, 2020. URL: https://doi.org/10.1016/j.engappai.2019.103301, doi:10.1016/j.engappai.2019.103301.

[40] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996. doi:10.1145/235815.235821.

[41] Qianmu Li, Shunmei Meng, Sainan Zhang, Ming Wu, Jing Zhang, Milad Taleby Ahvanooey, and Muhammad Shamrooz Aslam. Safety Risk Monitoring of Cyber-Physical Power Systems Based on Ensemble Learning Algorithm. *IEEE Access*, 7:24788–24805, 2019. doi:10.1109/ACCESS.2019.2896129.

[42] Miguel Saez, Francisco Maturana, Kira Barton, and Dawn Tilbury. Anomaly detection and productivity analysis for cyber-physical systems in manufacturing. *IEEE International Conference on Automation Science and Engineering*, 2017-August:23–29, 2017. doi:10.1109/COASE.2017.8256070.

[43] Miguel A. Saez, Francisco P. Maturana, Kira Barton, and Dawn M. Tilbury. Context-Sensitive Modeling and Analysis of Cyber-Physical Manufacturing Systems for Anomaly Detection and Diagnosis. *IEEE Transactions on Automation Science and Engineering*, 17(1):29–40, 2020. doi:10.1109/TASE.2019.2918562.

[44] Rui Pinto, Gil Gonçalves, Eduardo Tovar, and Jerker Delsing. Attack detection in cyber-physical production systems using the deterministic dendritic cell algorithm. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1552–1559, 2020. doi:10.1109/ETFA46521.2020.9212021.

[45] Ada Bagozi, Devis Bianchini, Valeria De Antonellis, Alessandro Marini, and Davide Ragazzi. Big data summarisation and relevance evaluation for anomaly detection in cyber physical systems, 2017. doi:10.1007/978-3-319-69462-7_28.

[46] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A Framework for Clustering Evolving Data Streams - Proceedings of the 29th international conference.pdf. *Vldb*, pages 81–92, 2003. URL: http://www.vldb.org/conf/2003/papers/S04P02.pdf.

[47] Dau Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *In Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734. Morgan Kaufmann, 2000.

[48] Lukas Kaupp, Ulrich Beez, Jens Hülsmann, and Bernhard G. Humm. Outlier detection in temporal spatial log data using autoencoder for industry 4.0. *Communications in Computer and Information Science*, 1000:55–65, 2019. doi:10.1007/978-3-030-20257-6_5.

[49] Alexander von Birgelen, Davide Buratti, Jens Mager, and Oliver Niggemann. Self-organizing maps for anomaly localization and predictive maintenance in cyber-physical production systems. *Procedia CIRP*, 72:480–485, 2018. 51st CIRP Conference on Manufacturing Systems. URL: https://www.sciencedirect.com/science/article/pii/S221282711830307X, doi:https://doi.org/10.1016/j.procir.2018.03.150.

[50] Mauro Caporuscio, Francesco Flammini, Narges Khakpour, Prasannjeet Singh, and Johan Thornadtsson. Smart-troubleshooting connected devices: Concept, challenges and opportunities. *Future Generation Computer Systems*, 111:681–697, 2020. URL: https://doi.org/10.1016/j.future.2019.09.004, doi:10.1016/j.future.2019.09.004.

[51] Sahar Aldhaheri, Daniyal Alghazzawi, Li Cheng, Ahmed Barnawi, and Bandar A. Alzahrani. Artificial Immune Systems approaches to secure the internet of things: A systematic review of the literature and recommendations for future research. *Journal of Network and Computer Applications*, 157(July 2019):102537, 2020. URL: https://doi.org/10.1016/j.jnca.2020.102537, doi:10.1016/j.jnca.2020.102537.

[52] Stephanie Forrest, Lawrence Allen, Alan S. Perelson, and Rajesh Cherukuri. Self-nonself discrimination in a computer. *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 202–212, 1994. doi:10.1109/risp.1994.296580.

[53] Polly Matzinger. The danger model: a renewed sense of self. *Science*, 296(5566):301–305, 2002.

[54] Frank Macfarlane Burnet et al. A modification of jerne's theory of antibody production using the concept of clonal selection. *Australian Journal of Science*, 20(3):67–9, 1957.

[55] Leandro Nunes De Castro and Fernando José Von Zuben. Artificial immune systems: Part i–basic theory and applications. *Universidade Estadual de Campinas, Dezembro de, Tech. Rep*, 210(1), 1999.

[56] Berna Haktanirlar Ulutas and Sadan Kulturel-Konak. A review of clonal selection algorithm and its applications. *Artificial Intelligence Review*, 36(2):117–138, 2011. doi:10.1007/s10462-011-9206-1.

[57] Jason Brownlee. Clonal selection algorithms. *Technical Report 070209A*, (February):1–13, 2007.

[58] Leandro Nunes De Castro and Fernando José Von Zuben. An evolutionary immune network for data clustering. *Proceedings - Brazilian Symposium on Neural Networks, SBRN*, 2000-Janua:84–89, 2000. doi:10.1109/SBRN.2000.889718.

[59] Jon Timmis, Mark Neal, and John Hunt. An artificial immune system for data analysis. *BioSystems*, 55(1-3):143–150, 2000. doi:10.1016/S0303-2647(99)00092-1.

[60] N. K. Jerne. Towards a network theory of the immune system. *Annals of Immunology*, 125C(1-2):373–389, Jan 1974.

[61] Geoffrey W Hoffmann. Immune network theory. *Monograph. URL: www. physics. ubc. ca/˜ hoffmann/ni. html*, 2008.

[62] Thomas Knight and Jon Timmis. AINE: An immunological approach to data mining. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 297–304, 2001. doi:10.1109/icdm.2001.989532.

[63] Julie Greensmith, Uwe Aickelin, and Jamie Twycross. Articulation and clarification of the dendritic cell algorithm. In *International Conference on Artificial Immune Systems*, pages 404–417. Springer, 2006.

[64] J P Twycross. Integrated innate and adaptive artificial immune systems applied to process anomaly detection. (January), 2007.

[65] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

[66] Zhou Ji and Dipankar Dasgupta. Real-valued negative selection algorithm with variable-sized detectors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3102(September 2016):287–298, 2004. doi:10.1007/978-3-540-24854-5_30.

[67] Julie Greensmith. The Dendritic Cell Algorithm. *Thesis for the Degree of Doctor of Philosophy*, (October):1–316, 2007. URL: https://pdfs.semanticscholar.org/8368/b433f255b35d09b5a09d20628c19dbe8b00a.pdf%0Ahttp://ima.ac.uk/papers/greensmith_thesis.pdf.

[68] Najla Badie Ibraheem Al-Dabagh and Ismael Ali Ali. Design and implementation of artificial immune system for detecting flooding attacks. *Proceedings of the 2011 International Conference on High Performance Computing and Simulation, HPCS 2011*, pages 381–390, 2011. doi:10.1109/HPCSim.2011.5999850.

[69] Julie Greensmith and Uwe Aickelin. The deterministic dendritic cell algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5132 LNCS:291–302, 2008. doi:10.1007/978-3-540-85072-4_26.

[70] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[71] Giuseppe Settanni, Florian Skopik, Anjeza Karaj, Markus Wurzenberger, and Roman Fiedler. Protecting cyber physical production systems using anomaly detection to enable self-adaptation. In *Proceedings - 2018 IEEE Industrial Cyber-Physical Systems, ICPS 2018*, pages 173–180. Institute of Electrical and Electronics Engineers Inc., jun 2018. doi:10.1109/ICPHYS.2018.8387655.

[72] Cheng Zong, Song Huang, Erhu Liu, Yongming Yao, and Shi Qi Tang. Nowhere to Hide Methodology: Application of Clustering Fault Diagnosis in the Nuclear Power Industry. *IEEE Access*, 7:179864–179879, 2019. doi:10.1109/ACCESS.2019.2957807.

[73] Jacopo Parri, Fulvio Patara, Samuele Sampietro, and Enrico Vicario. A framework for Model-Driven Engineering of resilient software-controlled systems. *Computing*, 2020. URL: https://doi.org/10.1007/s00607-020-00841-6, doi:10.1007/s00607-020-00841-6.

[74] Adrien Bécue, Eva Maia, Linda Feeken, Philipp Borchers, and Isabel Praça. A new concept of digital twin supporting optimization and resilience of factories of the future. *Applied Sciences (Switzerland)*, 10(13), 2020. doi:10.3390/app10134482.

[75] Petar Radanliev, David De Roure, Max Van Kleek, Omar Santos, and Uchenna Ani. Artificial intelligence in cyber physical systems. *AI & society*, pages 1–14, 2020.

[76] Wissam Aoudi and Magnus Almgren. A scalable specification-agnostic multi-sensor anomaly detection system for IIoT environments. *International Journal of Critical Infrastructure Protection*, 30:100377, 2020. URL: https://doi.org/10.1016/j.ijcip.2020.100377, doi:10.1016/j.ijcip.2020.100377.

[77] Alejandro Bracho, Can Saygin, Hungda Wan, Yooneun Lee, and Alireza Zarreh. A simulation-based platform for assessing the impact of cyber-threats on smart manufacturing systems. *Procedia Manufacturing*, 26:1116–1127, 2018. URL: https://doi.org/10.1016/j.promfg.2018.07.148, doi:10.1016/j.promfg.2018.07.148.

[78] Konrad Wegener, Thomas Gittler, and Lukas Weiss. Dawn of new machining concepts: Compensated, intelligent, bioinspired. *Procedia CIRP*, 77(Hpc):1–17, 2018. URL: https://doi.org/10.1016/j.procir.2018.08.194, doi:10.1016/j.procir.2018.08.194.

[79] Ada Bagozi, Devis Bianchini, and Valeria De Antonellis. Designing Context-Based Services for Resilient Cyber Physical Production Systems, 2020. doi:10.1007/978-3-030-62005-9_34.

[80] Mingtao Wu, Zhengyi Song, and Young B. Moon. Detecting cyber-physical attacks in CyberManufacturing systems with machine learning methods. *Journal of Intelligent Manufacturing*, 30(3):1111–1123, 2019. doi:10.1007/s10845-017-1315-5.

[81] Giuseppe Bernieri and Federica Pascucci. Improving security in industrial internet of things: A distributed intrusion detection methodology. *Advanced Sciences and Technologies for Security Applications*, pages 161–179, 2019. doi:10.1007/978-3-030-12330-7_8.

[82] V. Ariharan, Subha P. Eswaran, Srinivasarao Vempati, and Naveed Anjum. Machine Learning Quorum Decider (MLQD) for large scale IoT deployments. *Procedia Computer Science*, 151(2018):959–964, 2019. URL: https://doi.org/10.1016/j.procs.2019.04.134, doi:10.1016/j.procs.2019.04.134.

[83] Mahmoud Parto, Christopher Saldana, and Thomas Kurfess. Real-time outlier detection and Bayesian classification using incremental computations for efficient and scalable stream analytics for IoT for manufacturing. *Procedia Manufacturing*, 48(2019):968–979, 2020. URL: `https://doi.org/10.1016/j.promfg.2020.05.136`, `doi:10.1016/j.promfg.2020.05.136`.