FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Enhancing ML Models for Solar Weather Forecasting using Clustering and Adversarial Anomaly Detection

Ivo Saavedra

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado em Engenharia Informática e Computação

Supervisor: André Restivo

Co-Supervisor: Filipa Barros

July 28, 2023

Enhancing ML Models for Solar Weather Forecasting using Clustering and Adversarial Anomaly Detection

Ivo Saavedra

Mestrado em Engenharia Informática e Computação

July 28, 2023

Resumo

A Ciência do Clima Espacial é um campo de pesquisa vital que visa compreender as condições na superfície do Sol, que podem afetar negativamente a vida na Terra. Apesar de ser um campo bem desenvolvido, as condições que levam a essas fenómenos nefastos ainda não são totalmente compreendidas. Esta limitação é principalmente atribuída à dificuldade em obter dados de alta qualidade da superfície do Sol.

De forma a contornar este problema, alguns modelos de simulação tentam extrapolar as condições no Sol analisando dados de outras medições. Um exemplo disso é o MULTI-VP que usa magnetogramas de várias fontes (por exemplo, Wilcox Solar Observatory) e determina a estrutura do campo magnético de fundo do vento solar. No entanto, essas simulações demoram muito tempo a convergir e requerem estimativas iniciais de especialistas, feitas manualmente. Recentemente, uma abordagem baseada em Machine Learning foi projetada para atenuar esses problemas. Esta removeu a necessidade de estimativas iniciais, prevendo automaticamente as condições iniciais da simulação. Além disso, provou que pode haver uma redução significativa no tempo de execução do simulador. Apesar disso, os modelos de previsão ainda não são robustos o suficiente para serem usados em aplicações do mundo real. Como em muitos outros problemas de Machine Learning, acreditamos que a presença de anomalias no conjunto de dados de treino tenha prejudicado a sua performance.

Uma possível teoria é que as condições iniciais estão diretamente correlacionadas com o tempo de computação da simulação, e que melhores estimativas inicias levam a tempos de execução do MULTI-VP mais rápidas. Posto isto, nesta dissertação, aplicamos vários métodos de clustering e de deteção de anomalias de forma a melhorar a qualidade das condições iniciais e verificar se isso resultava em simulações mais rápidas do MULTI-VP.

Vários métodos de clustering foram testados nos dados de magnetogramas usados no treino do modelo de previsão, para determinar qual seria o mais apropriado. Adicionalmente, um conjunto de métodos de treino baseados nas técnicas de clustering foram testados, dos quais pelo menos um gerou estimativas mais próximas às previsões da simulação. Apesar disto, não houve qualquer redução no tempo de execução da simulação.

Para melhorar os resultados dos experimentos anteriores, vários métodos de detecção de anomalia adversária foram testados. Os modelos foram retreinados sem as anomalias detectadas o que resultou em piores condições de fluxo inicial quando comparado ao resultado final do MULTI-VP; no entanto, o tempo de computação foi ligeiramente menor do que na implementação anterior.

Concluindo, os resultados das experiências com os métodos de clustering e de deteção de anomalias adversariais parecem indicar que o desempenho do simulador não está correlacionado com a proximidade das condições iniciais às soluções da simulação.

Palavras-chave: Meteorologia Espacial, Vento Solar, Aprendizagem Computacional, Clustering, Deteção Adversarial de Anomalias

Abstract

Space Weather Science is a vital field of research that aims to understand the conditions on the Sun's surface, which can negatively impact life on Earth. Despite being a well-researched field, the conditions that lead to these phenomena are still not fully understood. This limitation is mainly attributed to the difficulty in acquiring high-quality data from the Sun's surface.

To circumvent this issue, some simulation models try to extrapolate the conditions on the Sun by analyzing data from other measurements. An example of this is MULTI-VP which uses magnetograms from various sources (*e.g.*, Wilcox Solar Observatory) and determines the structure of the solar wind's background magnetic field. However, these simulations take a long time to converge and require initial expert estimations, which are handmade. Recently, a machine learning approach has been designed to attenuate these issues. It removed the need for initial estimates by automatically predicting the starting conditions of the simulation. In addition, it has shown that there can be a significant reduction in the execution time of the simulator. Despite this, given their lack of physical cohesion, the prediction models are still not robust enough for real-world applications.

We posit that initial conditions directly influence the computation time of the simulation and that better initial estimates will lead to faster executions. Thus, in this dissertation, we applied clustering and anomaly detection techniques to improve the quality of the initial conditions and determine if this would lead to faster MULTI-VP executions.

Several clustering experiments were conducted on the available magnetogram dataset to determine the best-suited clustering method. In addition, various clustering-based approaches for enhancing the prediction model were tested, with the selected method producing initial flow conditions closer to the simulation outputs. Despite this, there was no reduction in the computation time of the simulation.

To improve the previous experiments' results, various adversarial anomaly detection methods were designed and tested. The prediction models of the clustering-based experiments were retrained without the detected anomalies and resulted in worse initial flow conditions when compared to the final output of MULTI-VP; however, this time, the computation time was slightly lower than on the previous implementation.

In conclusion, the experiments conducted in this dissertation seem to indicate that the performance of the MULTI-VP simulator is not directly linked to the initial flow condition's approximation to the final solutions.

Keywords: Space Weather, Solar Wind, Machine Learning, Clustering, Adversarial Anomaly Detection

Contents

List of Figures

List of Tables

Abbreviations

AE	Autoencoder
ANN	Artificial Neural Network
AU	Astronomical Unit
AUC	Area Under the Curve
BMU	Best Matching Unit
BPTT	Backpropagation Through Time
CH	Calinski-Harabasz Index
CME	Coronal Mass Ejection
CNN	Convolutional Neural Network
D	Discriminator
DB	Davies-Bouldin Index
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DNN	Deep Neural Network
DPC	Density Peaks Clustering
DTW	Dynamic Time Warping
E	Encoder
ESA	European Space Agency
FB	Feature Bagging
FP	False Positive
FN	False Negative
G	Generator
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
HCA	Hierarchical Clustering Algorithm
LPP	Locally Preserving Projections
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MCDM	Multiple Criteria Decision-making Problem
MHD	Magnetohydrodynamic
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NN	Neural Network
PCA	Principal Component Analysis
RBM	Restricted Boltzmann Machine
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
ROC	Receiver Operator Characteristic
RUS	Random Undersampling

S	Silhouette Score
SLR	Systematic Literature Review
SMOTE	Synthetic Minority Oversampling
SSE	Sum of Squared Errors
SVM	Support Vector Machine
TP	True Positive
TN	True Negative
UB	Underbagging
VAE	Variational Autoencoder
KL	Kullback-Leibler
KNN	K-Nearest Neighbors

Chapter 1

Introduction

The Sun continuously releases a stream of particles known as *solar wind*. This stream consisting of high-velocity charged particles (e.g. protons and electrons) can reach planetary surfaces unless thwarted by an atmosphere, magnetic field, or both. In Earth's case, the magnetosphere and the atmosphere, to a smaller extent, block out most of the radiation emitted by the Sun. However, other more extreme events like solar flares and CMEs (Coronal Mass Ejections) can provoke negative effects on the Earth's surface and upper atmosphere.

These events can impact the Earth in three primary ways. Firstly, they can cause radio blackouts that predominantly affect satellites and, consequently, geolocation and communication systems. Secondly, solar radiation storms can endanger astronauts and spacecraft orbiting the Earth. Lastly, the most severe of these events are geomagnetic storms, which have historically caused significant disturbances. Examples of such disturbances include the Carrington Event in 1859 and a geomagnetic storm that affected Quebec's power grid in 1989[1].

Space Weather Science is a field that aims to prevent the consequences of such events; however, the factors that result in their formation are still not fully understood. Some simulation models have been designed to try and fill this gap [2, 3], but they require initial expert guesses. Recently, an ML (Machine Learning) model has been developed [4] to improve these initial predictions based on known data. Like other ML problems, the quality of the predictions is very dependent on the quality of the training data.

1.1 Motivation

In 1859, Carrington recorded the first and largest known solar flare in history, which is now commonly referred to as the Carrington event. This phenomenon was so extreme that it caused geomagnetic storms in unexpected latitudes and provoked fires on telegraph wires. Carrington was also able to correlate the event with a geomagnetic storm that occurred several hours later. His pioneering work is widely recognized as the inception of the scientific discipline known as Space Weather [5].

[1]List of solar storms: `https://en.wikipedia.org/wiki/List_of_solar_storms`

The field of *Space Weather Science* emerged with the aim of understanding the formation of phenomena that could affect Earth to evaluate their effects and to create early warning systems. Despite significant advancements in this scientific field, the correlations between the Sun's structures and these phenomena are not yet fully formulated and are mostly speculative. For instance, it is still unknown why the atmosphere of the Sun is considerately hotter than its surface. The main leading theory is that the magnetic field transports energy deep from the convection zone through the surface and up to the atmosphere. It is also posited that the magnetic fields on the surface sometimes collide, provoking large explosions and therefore causing the atmosphere to heat even further. Another enigma is the acceleration of the solar wind (up to millions of miles) out of the corona. Some correlation between the magnetic field and solar wind acceleration has been found; however, the effect remains a mystery.

The answer to questions like these can contribute to a greater understanding of the underlying processes of the Sun that influence solar weather. Consequently, predicting future events that could impact the Earth, satellites, and space stations orbiting it would become easier. Thus far, these answers have been delayed by the technological limitations on measuring solar events. In 2018, NASA's Parker Solar [2] probe was launched on a mission to orbit the Sun's, to understand the acceleration of solar wind at the corona. More recently, ESA launched the Solar Orbiter to measure the solar wind and record images of the uncharted polar regions, closer than every other solar probe[3]. The PUNCH[4] mission was launched to try and shed some light on the formation of the solar wind on the Sun's surface.

MHD (magnetohydrodynamic) simulators like MULTI-VP [2] and ENLIL [3], were developed in order to try and extrapolate coronal conditions from limited observations of solar events from probes and observatories. The execution of these simulations relies on initial estimations, usually performed by hand after an analysis of the data (a very time-consuming task). Additionally, it has been posited that good initial estimations have the potential to reduce the simulation's execution time significantly. The process of making the initial predictions as well as the extensive execution time of the simulations make it difficult to create early warning systems that can prevent the effects of solar events on Earth.

1.2 Problem Definition

The exponential growth of data acquisition has presented a significant challenge in promptly analyzing the vast amount of available information. The sheer amount of data is becoming increasingly hard for researchers to process, especially on data linked to near-real-time utilization.

[2]Parker Solar Probe: Humanity's First Visit to a Star https://www.nasa.gov/content/goddard/parker-solar-probe-humanity-s-first-visit-to-a-star

[3]ESA: Solar Orbiter https://www.esa.int/Science_Exploration/Space_Science/Solar_Orbiter

[4]NASA Selects Missions to Study Our Sun, Its Effects on Space Weather https://www.nasa.gov/press-release/nasa-selects-missions-to-study-our-sun-its-effects-on-space-weather

Machine learning has become one of the main methods of evaluating the data efficiently for problems associated with space weather prediction. However, most deep learning models are very susceptible to large variations in the data that can severely decrease the performance of these models. The anomalies can originate from the instrument and detector noise, statistical noise from the small flux of photons, and external noise may include instrumentation jitter, stray starlight, and cosmic ray background [6].

Recently, a NN [4] was developed to perform initial estimations for solar wind profiles that would later be fed to MULTI-VP [2]. This reduced the time needed to generate the initial estimations required by the simulation, which were previously done by hand. Additionally, it was observed that producing initial estimates closer to the final simulation reduced the computation time of the simulation, with a mean speedup of 1.06. Despite this, it was concluded that the prediction model was not producing the best possible estimates. On one hand, this might have been because the model managed to learn the most concentrated observations and failed to learn the ones in the peripheries. Another possibility is that the existence of anomalies in the dataset was hindering the performance of the model resulting in worse estimates.

The problem addressed in this thesis is to enhance the quality of the training data, which is then to be used for predicting initial conditions associated with solar wind behaviour.

1.3 Goals

This thesis aims to enhance the prediction ability of the neural network [4] responsible for generating initial predictions for solar wind formations. With this, we aim to produce closer initial condition estimations to MULTI-VP's [2] final estimates. We intend to achieve this by:

(1) applying clustering techniques in the training of the models for initial condition prediction so these can better capture the features of the data;

(2) applying adversarial anomaly detection techniques to detect and filter faulty measurements in the data used to train the prediction models.

As a consequence of better initial condition estimations, we intend to reduce the computation time that the simulation takes to reach a viable solution.

1.4 Document Structure

This first chapter has provided the context for the problem addressed in this dissertation. The remaining sections of the document are organized as follows:

- Chapter 2, explains the background needed to understand the current problems in the area of space weather science. Some concepts related to neural networks and anomalies are also introduced to provide a basis for methods discussed in the remainder of the document.
- Chapter 3, provides an analysis o fthe current state-of-the-art methods for clustering and adversarial anomaly detection.
- Chapter 4, goes into more depth on the problem this thesis aims to solve and the approach that will be taken.
- Chapter 5 explains the clustering methods used on the dataset, followed by the experiments and discussion of the results.
- Chapter 6 starts by explaining the origins of adversarial learning and the experiments undertaken with this type of approach, followed by a brief discussion of the results.
- Chapter 7 evaluates this thesis's hypothesis and provides a brief conclusion for the work carried out.

Chapter 2

Background

In this chapter, a basic introduction to solar weather and the main events associated with it will be presented. Additionally, a brief explanation of the Machine Learning (ML) terms that are needed in the context of this dissertation will be provided.

2.1 Space Weather

"Space weather refers to the dynamic, highly variable conditions in the geospace environment, including those on the Sun, in the interplanetary medium, and in the magnetosphere-ionosphere-thermosphere system. Adverse changes in the near-Earth space environment can diminish the performance and reliability of both spacecraft and ground-based systems." ([7])

2.1.1 Solar Phenomena

The increasing dependence on technologies vulnerable to solar weather conditions has made it increasingly important to detect incidents, like the Carrington event [5], that would significantly damage assets on Earth, beforehand. In this section, a brief introduction to these phenomena will be provided.

Sunspots. These structures consist of dark central regions (umbra), which are colder than the rest of the Sun's surface, and more luminous external regions (penumbra). Sunspots are known to be regions with strong magnetic fields (1000 times stronger than in the surrounding normal surface). It is theorized that these magnetic fields interfere with the convection of the Sun, effectively cooling the regions where they appear. Sunspots often originate as groups concentrated in specific areas of the Sun, and their frequency and size vary with the 11-year solar cycle [8].

Coronal Mass Ejections (CME). These events are best described as mass ejections of plasma into space after solar eruptions. It is posited that their formation occurs mainly from magnetic reconnection, which occurs when magnetic field lines collide and realign into a new configuration (releasing large amounts of energy). After their formation, CMEs can expand through space to great distances and collide with planetary atmospheres. The effects on Earth include geomagnetic storms, damage to electronics on orbiting satellites, endangerment of astronauts in extraterrestrial settings or planes' guidance systems, damage to electrical grids and disruption of radio communication. Due to their length (at least 0.25AU), CMEs can take over a day to pass Earth. While slower CMEs can take days before reaching Earth, the fastest ones arrive in approximately 15-18 hours. [9, 8].

Solar Flares. These events are often characterized as intense and temporary releases of energy that blast large amounts of charged particles into space. Flares are known to last only a few minutes and reach temperatures of 100 million K, much higher than the ones at the core of the Sun (of about 15 million K). Like CMEs, flare formations are associated with energy releases from magnetic reconnection and are primarily concentrated in the Sun's active regions. Flares are almost always associated with CMEs, but can also occur separately from them. Flares can be classified as A, B, C, M or X based on the X-ray flux measurements on Earth [8, 9].

Solar Wind. This phenomenon results from plasma's constant expulsion and expansion into interplanetary space. Specifically, the solar wind consists of mostly protons, helium nuclei and electrons that move away from the Sun at supersonic speeds and carry the Sun's magnetic field with it. It streams away from the Sun at different velocities, which allows for it to be classified as fast (700 to 750 kms^{-1}) or slow (300 to 400 kms^{-1}). The latter usually occurs on the Sun's equatorial line, and the former is concentrated in open magnetic field regions of the Sun. The exact originating factors for the slow solar wind are still unknown; however, for fast solar wind, it is known that it originates in coronal holes. The solar wind has as standard properties (at 1 AU) a velocity of 400 kms^{-1}, a temperature of 1 million K, and a density of 5 particles cm^{-3} [8, 9].

2.1.2 Magnetohydrodynamic Simulation Models

The reasons for the acceleration of solar wind are mainly attributed to thermal heating; however, this does not explain the high speeds it reaches. The additional acceleration is often attributed to the magnetic field, but no physical model can currently explain the correlation. Similarly, the origins of the solar wind are mostly unknown, especially for slow solar wind compared to fast solar wind [9].

These difficulties are mostly attributed to the absence of sensitive, high-resolution coronal magnetic field measurements that do not allow for a full explanation of coronal physics. The limitations in this field make it more challenging to comprehend solar events like CMEs and the acceleration of the solar wind [10].

To try and fill these gaps in Space Weather Science, several magnetohydrodynamic (MHD) models have been developed to try to give an answer to these questions. These models compute numerically intensive problems based on MHD equations. For this, they require the definition of appropriate boundaries and initial state definitions. Due to their complexity, MHD models often focus on single events and introduce assumptions and simplifications for the surrounding phenomena. For this reason, the research community has developed relatively simple MHD models to describe complex processes in the past decades.

2.2 Neural Networks

Neural Networks (NN) or Artificial Neural Networks (ANNs) are one of the main used ML models. The design of NNs takes inspiration from the biological neural networks found in animal brain structures. NNs consist of a set of node layers, the input layer, one or more hidden layers, and the output layer. Each node is loosely connected to other nodes, each with its associated weights and threshold values. The connections between the nodes are called edges, allowing for communication between nodes and also having their associated weights. Signals travel from the input layer through the hidden layers to the output layer. If a given node's output is higher than its associated threshold value, it is activated and sends data to the next layer.

2.2.1 Deep Neural Networks

Deep Neural Networks (DNNs) derive from the deep learning subfield of ML and are based on NNs. Their distinguishing factor from previous methods is representation learning (also known as feature learning) with multiple levels of abstraction. This technique allows models to discover the underlying data structures needed for feature detection and classification. These methods have been successfully applied in speech recognition, visual object recognition and detection, among others ([11]). The term "deep" comes from a large number of stacked layers that the model has compared to normal NNs. The most basic features are learned in the starting layers and the most complex at the bottom layers. DNNs usually have a feed-forward architecture where the data flows from the top layers to the output layer. In the end, the errors are back-propagated through the network to adjust the weights of the nodes in each layer.

2.2.2 Recurrent Neural Networks

RNNs are a type of neural network that is suitable for sequential data. The main goal of this architecture is to detect patterns in the input sequences, which makes it suitable for the tasks like natural language processing and time-series prediction. Unlike conventional feed-forward networks, RNNs have cycles that transmit data onto themselves which allows them to consider previous inputs and not only the current one. This is why some authors refer to RNNs as neural networks with "memory".

Recurrent networks take advantage of the backpropagation through time (BPTT) algorithm, which is a derivation of the backpropagation algorithm for sequence data. Like the backpropagation algorithm, BPTT is used to train the weights of the network. The main difference is that BPTT unfolds the network in time, which allows for the application of the backpropagation algorithm. This is done by unrolling the network in time and then applying the backpropagation algorithm to the unrolled network. The unrolling process is done by creating a copy of the network for each time step and then connecting them. The result is a feed-forward network that can be trained with the backpropagation algorithm. The main disadvantage of this method is that it is computationally expensive and can be unstable due to the vanishing gradient problem (more details in 6.1), as previous states may lose relevance when the sequence grows.

Some alternatives have been proposed to solve the vanishing gradient problem, such as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures. These architectures are a type of RNN that have internal mechanisms called gates that can regulate the flow of information. These gates can learn which data in the sequence is relevant to keep or discard. This allows for the learning of long-term dependencies in the data, which is not possible with traditional RNNs. The main difference between LSTM and GRU is that the former has three gates (input, output and forget) and the latter only two (update and reset). GRU is often attributed to being a lightweight version of LSTM, as it has fewer parameters and is easier to train. However, LSTM is still the most popular architecture for sequence data.[12]

2.2.3 Autoencoders

Autoencoders (AE) are a subtype of neural networks, and their main purpose is to compress existing data into meaningful representations, which are then decompressed back to the original input. Examples of applications include classification, clustering, anomaly detection (oftentimes adversarially trained), dimensionality reduction, and others. Its objective is to minimize the distance between the reconstructed and original samples. Therefore, the training goal is to minimize the loss function

$$L(\theta,\phi) = \mathbb{E}_{x\sim\mu_{ref}}[d(x, D_\theta(E_\phi(x))] \tag{2.1}$$

where θ, ϕ are the parameters of the encoder, E_θ and the decoder D_ϕ, respectively; x is the original sample, μ_{ref} is the reference probability distribution and d is the distance function that measures the reconstruction quality by comparing x with its reconstructed examples, $D_\theta(E_\phi(x)$.

Regularizations are often applied to objective functions in line with the application of the autoencoder or to avoid overfitting. Some methods purposely reduce the reconstruction ability of the autoencoder to produce more meaningful compressions and vice versa. Possible variations of AEs include *Sparse AEs*, which aims to reduce the dimensionality of the input data, *Denoising AEs*, mainly used to reduce noise in images and *Contracting AEs*, which aims to reduce the number of features that need to be learned by removing the unnecessary ones.

2.3 Anomaly Definitions

Anomalies can be classified as data points, events or observations that vary significantly from the rest of the data. The most common data faults come in the form of outliers. These can be classified as data points significantly different from the rest of the data [13]. They are often referred to as anomalies, out-of-distribution data, novelties, and deviations [13, 14].

Inliers are data points that lie within or very close to the normal distribution of the majority of the data, but they still exhibit some distinct differences from the remaining points. From a clustering perspective (refer to 5), inliers appear between typical clusters without attaching to any but still being close enough to one to be considered a part of it [15]. Because of these characteristics, inliers are significantly harder to detect than outliers.

Datasets often contain unusual characteristics that, in some cases, can be informative in determining the origins of anomalies. They can be intentional when they result from nefarious actions(e.g., credit card fraud); and unintentional when they occur naturally (e.g., sensor anomalies, input errors). Following are some examples of anomaly detection:

- **Credit-card fraud:** theft of credit card credentials can be detected by analysing the transaction history of the target.
- **Medical diagnosis:** anomalies in scans can indicate possible diseases.
- **Fault diagnosis:** detection of faults in critical components (e.g., space shuttles).
- **Intrusion detection:** detecting unauthorized access to computer networks.

Throughout this dissertation, anomalies will be used to describe faulty measurements in the data used to train machine learning models. More concretely, this term will refer to both outlier and inlier values in the data.

2.3.1 Anomaly Detection Approaches

There are three main approaches for anomaly detection: unsupervised, supervised and semi-supervised methods. The objective of the first is to detect anomalies in the dataset with no prior knowledge of the data. This approach follows the same logic as clustering: it defines one or clusters and then identifies every point outside the clusters as an anomaly. The assumption is that normal data points occur more often than anomalous ones. Data points that occur more infrequently are considered anomalies.

Supervised anomaly detection aims to model the normality and abnormality of the data, and as any supervised learning problem, requires labelled data. Like any supervised learning problem, it requires labelled data. Classification algorithms work better with balanced distributions of normal and anomalous data to improve generalization. However, achieving balanced distributions is often challenging in these problems since anomalies are typically the minority class. Supervised detection is also an efficient way of detecting inliers as they are labelled as such.

Semi-supervised detection is a compromise between the previous two approaches. The objective is to model only the normal distribution of the dataset and then use the model on the whole dataset. The model will detect the novel samples not observed during the training phase and classify them as anomalies. This approach requires a preprocessing of the dataset to create a dataset with only normal samples that can later be used in the training phase [16].

Chapter 3

State of The Art

A Systematic Literature Review (SLR) was performed to understand the current trends of clustering methods (Section 3.1) and the use of adversarial learning for anomaly detection (Section 3.2). This chapter is divided into two parts. The first part will focus on the relevant clustering methods for this work. The second part will focus on the most relevant adversarial learning architectures for anomaly detection. Each part will start with the definition of the search questions (Section 3.1.1 and 3.2.2) and the search queries (Section 3.1.2 and 3.2.2). Next, the inclusion/exclusion criteria (Section 3.1.3 and 3.2.3) for the obtained results are identified. Finally, the results are presented and discussed (Section 3.1.4 and 3.2.4).

3.1 Clustering

Several approaches have been proposed for the clustering task, including partition-based, hierarchical, density-based, grid-based and model-based methods. This section will discuss the most relevant methods for this thesis. The search for relevant clustering methods was performed on the Scopus[1] and Google Scholar[2] databases. As clustering is already a well-developed field of study, the search was limited to the last ten years. The search queries are presented in Section 3.1.2, while the inclusion/exclusion criteria are presented in Section 3.1.3. The results and their subsequent discussion are presented in Section 3.1.4.

3.1.1 Survey Research Questions

Two search questions were defined to guide the search for relevant clustering methods. These are:

C_SQ1 *What are the state-of-the-art clustering algorithms?* Throughout the years, several clustering algorithms have been proposed. This question aims to identify the most relevant and widely used ones that can be used for this study.

[1]Scopus `https://www.scopus.com/`

[2]Google Scholar `https://scholar.google.com/`

C_SQ2 *Are there any metrics that can be used to evaluate the clustering algorithms?* This question aims to identify the most relevant metrics for evaluating clustering algorithms. These metrics will be used to evaluate the quality of the developed clustering algorithm and to choose the most appropriate one for the task at hand.
C_SQ3 *Are there any clustering approaches applied to enhance the performance of neural networks?* As this study aims to improve the predictive ability of solar wind prediction models, this question aims to identify the methodologies of other studies with similar goals.

3.1.2 Search Queries

The search questions of the previous section guide the construction of the search queries. As previously said, the search was performed on two search engines. The first query was performed on Scopus, and the second on Google Scholar. The first search question is the following:

cluster* AND data* AND (method* OR analy* OR algo*) AND NOT imag*

In addition to the search query, several filters were applied to the results to limit them to the last ten years and to exclude other articles that mentioned undesired keywords, such as "gene clusters". The first part of the query, *cluster* AND data**, is used to identify articles that mention the keywords cluster and its variations and data. The second part, *(method* OR analy* OR algo*)*, is used to identify articles related to clustering algorithms and analysis methodologies. The last part, *NOT imag**, excludes articles that mention the keyword image.
A second search was done on Google Scholar with the following query:

cluster* AND (improve OR enhance)
AND (neural network OR deep OR learn*)
AND (performance OR accuracy OR classif* OR predict*) -image

The goal of this query, the goal is to retrieve every publication related to the application of clustering methods in order to enhance the performance of neural networks. The first part of the query, *cluster* AND (improve* OR enhance*)*, is used to identify articles that mention the keywords cluster and its variations and its use to improve something in general. The second part, *(neural network OR deep OR learn*)*, is used to narrow the search to the clustering approaches to enhance machine learning approaches. The last part, *(performance OR accuracy OR classif* OR predict*)* narrows the search even more to articles that specifically mention performance, accuracy and other terms related to how well a system behaves. Similar to the first query, the last part, -*image*, is used to exclude articles that mention the keyword image. In addition, the results were filtered only to include articles from the last ten years.

3.1.3 Inclusion/Exclusion Criteria

Both queries yielded many results. A set of inclusion and exclusion criteria is presented in table 3.1 which serves as a way of filtering the most relevant papers for this thesis.

Criteria	**ID**	**Description**
Inclusion	I1	The document focuses on clustering algorithms, evaluation methods.
	I2	The results are clear, and the proposed goals are achieved.
	I3	The authors provide code or an extensive explanation of the approach.
	I4	Provides a comparison between the method and other clustering approaches or analyses multiple clustering approaches for a given application.
	I5	The article focuses on the application of clustering methods to improve the performance of machine learning approaches.
Exclusion	E1	The article doesn't cover clustering. This can occur due to the ambiguity of the term "cluster", as it can also be used in unrelated contexts.
	E2	The paper does not clearly evaluate the results of the approach.
	E3	The number of recorded citations is less than 50. As clustering is a well-developed study, the criteria for the relevancy of the paper can be more restrictive than on other fields.
	E4	The paper covers clustering algorithms for unsuited data types. Articles that pertain to the use of clustering techniques on multi-view data are excluded.
	E5	Was published before 2014.

Table 3.1: Clustering Inclusion and Exclusion Criteria

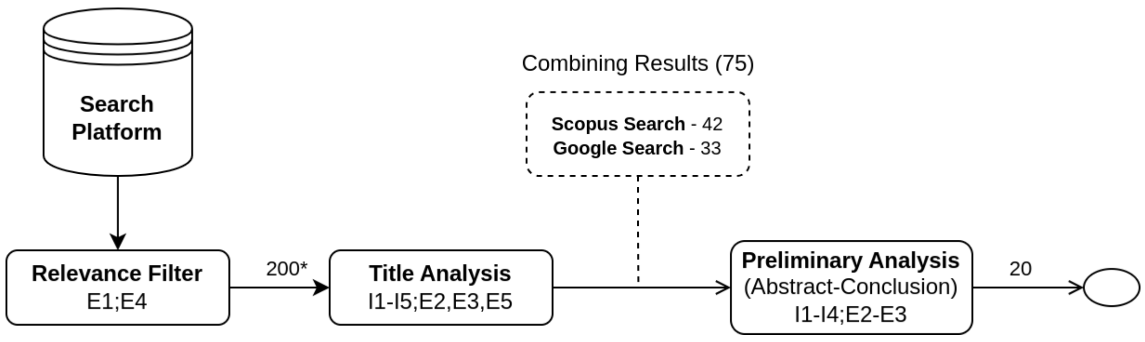


Figure 3.1: Clustering SLR Pipeline.
* Different sort order for each platform.

The result processing pipeline can be seen in Figure 3.1. The search queries were performed on both Scopus and Google Scholar. For each platform, a relevance filter (E1, E4) was applied to limit the results to the last ten years and exclude articles with low recorded citations. Next, the titles of the top 200 results of each search engine were analyzed. For this step, the results were ordered by the number of citations for Scopus Search and relevancy (best match) for Google Scholar. In addition, all inclusion criteria (I1-I5) were used to determine the relevance of the articles; however, greater importance was given to I5 in the second search to ensure more results that referenced the improvement of machine learning with clustering approaches.

After this, the abstracts and the conclusions of the selected papers were read with the same criteria as in the title analysis step. From this, 20 papers were picked for an extensive analysis, presented in Section 3.1.4. Note that one paper violates the exclusion rule E5 (published in 2010); however, it was included in the analysis as it was the only one that provided an extensive explanation of the types of clustering algorithms as well as possible metrics to evaluate them. This paper serves as a baseline by providing an overview of the most popular clustering algorithms and their applications.

3.1.4 Results

Jain [17] performs an extensive review of the K-Means algorithm. The author defines clusters as representations of n objects separated into k groups based on a similarity measure. The use cases for the application of clustering are also defined. These include the discovery of the underlying data structures, natural classification (degree of similarity between forms) and compression (a method for the organization and summarization of data).

A brief analysis of the history of clustering is provided, where the author shows the use of clustering methods in a wide range of fields and a brief explanation of hierarchical and partition-based clustering algorithms. Next, an explanation of the base K-Means algorithm is presented, along with a discussion of the parameters that need to be defined by the user. In addition, an analysis of other clustering methods, such as DBSCAN and CLIQUE, is performed.

In the next sections, the author explains the importance of the data representation and the features chosen for the clustering task. A discussion on the decision of the correct number of clusters is also performed, to which the author concludes that there is no definitive answer.

The next section explains the concept of cluster validity, which evaluates the results of cluster analysis in a quantitive and objective manner. The validity criteria assess the internal cluster structure, the degree of separation between structures and the degree of correspondence between the clustering and the external information (class labels). Possible cluster admissibility criteria are also presented. These are defined by rules that aim to ensure that clusters do not intersect each other and that the chosen algorithm provides the same results on data with different transformations, such as scaling.

Finally, the authors enumerate a set of guidelines that should be taken into account when choosing a clustering algorithm based on the criteria discussed in the previous sections.

Rodriguez et al. [18] propose that cluster centres have a higher density than their neighbours and are more distant from other cluster centres. In addition, the proposed algorithm is resilient to outliers as it identifies them and excludes them from the analysis.

The proposed method can detect non-spherical clusters and identify the correct number of clusters, like DBSCAN. Cluster centres are determined by selecting the points with the local maxima density values. The algorithm works by calculating the density of each point in the dataset and the distance from that point to other high-density points. After this step, the other points that are not considered centres are assigned to the nearest high-density point. This is done in a single step, making this algorithm more efficient than previous ones.

Tests were conducted on synthetic point distributions to evaluate the implementation and compared with other clustering algorithms. The results show that the algorithm is capable of reliantly detecting cluster structures in the different datasets. The authors also conclude that the algorithm is robust to changes in the scale of the dataset (as long as it doesn't affect the distances of the points) and that the approach is more practical than other methods, as it does not require the definition of parameters such as the probability distribution.

Fahad et al. [19] provide a detailed analysis of the application of different classes of clustering methods on big data. The authors' goal is to overcome the shortcomings of other surveys by systematically categorizing clustering algorithms; presenting the advantages and disadvantages of each class of algorithms; providing several evaluation measures; and finally analyzing the most representative algorithm of each category.

The authors identify five different clustering algorithm categories. These are partitioning, hierarchical, density-based, grid-based and model-based. A set of clustering algorithms is identified for each of the categories.

In the next section, the authors categorize the clustering methods in accordance with three properties of big data. These are volume, velocity and variety. The first refers to the ability of the algorithm to deal with large datasets with high dimensionality and also includes the existence of outliers. Velocity refers to the time complexity of the algorithms. And finally, variety refers to the ability of the algorithm to deal with different types of data and the cluster shapes produced by it.

The algorithm for each category is chosen by picking the ones that satisfy most of the above-defined criteria. An explanation of the algorithm is provided for each of the candidates. Eight simulated datasets, ranging from denial of service (DOS) attacks to water treatment operation logs, were used in the testing phase. Each of the candidates is evaluated based on the validity of the generated clusters, the stability[3] of the algorithms and the total execution

[3]Some clustering algorithms are based on a random component, which can lead to varying results in different executions.

time. The results show that no algorithm can satisfy all the criteria and that the choice of the algorithm depends on the type of data and the desired results.

Kou et al. [20] intend to evaluate several clustering algorithms on financial risk datasets as a multiple criteria decision-making problem (MCDM). The reason for this methodology is the lack of objective measures to determine the quality of clustering algorithms. The authors propose validating clusters regarding external and internal assessment and a relative test. Internal criteria evaluate the similarity of observations inside a cluster, while external measure the inter-cluster distances. Relative tests take advantage of previous knowledge of the dataset, such as class labels.

The process of clustering algorithm evaluation is divided into four steps. The first step is to select three financial risk datasets. Next, six popular clustering algorithms are picked for each dataset. In the third step, the authors aggregate eleven performance metrics into a single matrix for each dataset. Finally, the resulting matrices are passed through three MCDM methods that rank the clustering methods for each dataset.

The results showed that the K-Means repeated bisection algorithm was the best choice for the financial risk datasets, despite disagreements between the MCDM methods. The authors conclude that more research should be undertaken to find compromised solutions when MCDM methods disagree.

CAN [21] (Clustering with Adaptive Neighbors) is a clustering algorithm that simultaneously learns the data similarity matrix and clustering structure. Its goal is to assign each point's adaptive and optimal neighbours in the dataset. For this, the authors assume that data points with smaller distances should have larger probabilities of being neighbours. In addition, the algorithm imposes a constraint on the Laplacian of the similarity matrix so that the number of connected components in the data is the same as the number of clusters.

Projected Clustering with Adaptive Neighbors (PCAN) for high-dimensionality data is also implemented and is used to attenuate the difficulties of clustering on datasets with many features. The main goal was to reduce the dimension of the data without hindering the goals of the developed adaptive neighbours' algorithm.

In the experimentation phase, the authors start by testing the CAN algorithm in a toy dataset consisting of two clusters, in which the algorithm can reliably detect both connected components. Next, a comparison is performed with K-Means on a synthetic clustering dataset, in which CAN outperform the other method in accuracy. PCAN is tested alongside two dimensionality reduction methods, PCA and Locality Preserving Projections (LPP). The projection, as well as the clustering abilities, are evaluated for each of the reduction methods. PCAN can find the correct subspaces for the projection task compared to the other methods. In addition, the clustering results after the projection are also more reliable than the other methods.

Finally, CAN and PCAN, alongside other clustering methods, are tested with real-world datasets. The results showed that both methods outperformed the other baselines in every dataset (but one) regarding accuracy. CAN and PCAN alternated in the different datasets, with one outperforming the other in each of them.

Granato et al. [22] provide a critical analysis on the use of principal Component Analysis (PCA) and Hierarchical Clustering Algorithms (HCA) in the field of bioactive compounds. In this field, the discipline of chemometrics[4] is often used to assess the differences/similarities between observations or to project them into lower dimensions.

First, the authors provide a brief explanation of PCA is presented followed by an example of its application on fruit juices' chemical composition and antioxidant activity. The number of components that explain the most variance is decided by analyzing the cumulative explained variance of the dataset of PCAs with different numbers of components. The results showed that 81% of the data variation was explained by two components, with the first explaining 50% and the other 31%. After this example, an state-of-the-art analysis is provided on using PCA in food science studies.

In the next section, a brief explanation of HCA and the approaches to resolving the grouping problem is provided. The agglomerative approach considers every data point as a cluster at the beginning of the algorithm and then merges the cluster in pairs. The second method (divisive) starts with a single cluster and then divides it into smaller ones. In addition, the most used metrics of sample distance and linkage methods are enumerated. As in the previous section, an analysis of the state-of-the-art of application of HCA in food science is done.

In the end, the authors provide an example to explain the most common problem faced with the use of PCA and HCA in this field. They start by applying PCA to project the data samples into two dimensions. An analysis of the results showed that in one of the classes, the existence of outliers made it so some of the samples were significantly further away than the others. Then by applying HCA, it was demonstrated that different linkage distances produced varying numbers of clusters and would ultimately need to be decided by the user. With this simple example, they conclude that HCA and PCA should be avoided in this field and that calculating correlation coefficients would, in most cases, provide a better analysis of the data at hand.

Lin et al. [23] propose a clustering-based undersampling method for class-imbalanced datasets. The authors propose overcoming the shortcomings of undersampling methods, which come with the risk of excluding important features from the majority class. To achieve this, they intend to replace the random undersampling strategy with clustering methods. By using undersampling to cluster, the majority class will yield clusters with a similar size to the minority class.

[4]Science of extracting information from chemical processes using mathematics and statistics information.

An analysis of the traditional methods for resampling and classifier ensembles is first introduced. These include methods such as synthetic minority oversampling (SMOTE), random undersampling (RUS) and Underbagging (UB). Next, the main hypothesis of the work is presented in the form of clustering-based undersampling. The process consists of first dividing an imbalanced dataset into training and test sets. In the second step, the training data is divided into majority and minority class sets. In the following phase, the clustering-based undersampling method is used to reduce the number of samples in the majority class. Finally, the balanced training set is used to train a classifier, which is then used to classify the test set.

The authors present two strategies for clustering-based undersampling. In the first, the number of clusters equals the number of observations in the minority class. Then KMeans is applied to the majority class producing k cluster centres (centroids), which are then used to replace the entire majority class data. Ultimately, the number of observations will be the same for the majority and minority classes. The second strategy uses the same method to cluster the majority class. However, instead of using cluster centroids to reduce a single cluster into one observation, the authors fetch the sample in the cluster which is closest to the centroid (in terms of Euclidean distance). Both samples produce the same number of clusters, but the results from the latter are slightly different from the former.

Two studies were conducted to evaluate the methods discussed in the previous section. The first evaluates the performance of the methods in several small datasets and the other in two large datasets. Five classifiers and five state-of-the-art resampling methods were used to examine the classification performance. For the first study, results showed that the proposed methods significantly outperformed the baseline in terms of the receiver operator characteristic (ROC) curve. In addition, the nearest neighbour clustering-based undersampling method scored higher than the centroid-based one. The best classifier, in terms of accuracy (from the initial five), was the multilayer perceptron (MLP). In the second study, the results were mostly the same, with the best classifier (C4.5) being better than the MLP from the previous test.

The authors conclude that the design ensembles are well suited as a substitute for the traditional resampling methods and discuss the possibility of employing feature and instance selection to filter out unrepresentative features and data samples. In addition, other state-of-the-art classification algorithms could be combined with the clustering-based undersampling method to try and yield better results.

Douzas et al. [24] developed a novel heuristic oversampling method based on KMeans and SMOTE. The main goal is to overcome the common issues of SMOTE, such as the overfitting of the training data and the generation of noisy samples. Clustering allows for oversampling to target areas of input where artificial data generation is safer by ignoring noisy regions.

The algorithm is divided into three parts: clustering, filtering and oversampling. In the first step, the input is clustered into k clusters. Then, the clusters with higher proportions of minority-class samples are retained during the filtering phase. In the final step, SMOTE is applied to each selected cluster to achieve the desired target ratio of minority and majority instances.

The algorithm's effectiveness is compared with three classifiers trained on several imbalanced datasets for binary classification problems. Each dataset was subjected to five other oversampling techniques and then used to train the classification models. A model with unaltered data was also trained for each of the datasets. The results showed that models trained with the K-Means SMOTE method outperformed the ones trained on the original data and the models trained with data after applying the baseline oversampling methods.

K-Shape [25] is a highly accurate and efficient clustering algorithm for time-series data. In this paper, the authors propose a scale/translate/shift-invariant distance measure derived from a cross-correlation measure, present a new way of calculating cluster centroids with the new distance measure, and develop a new algorithm for time-series data based on the previous two.

An enumeration and explanation of five possible time-series invariances are provided. These include scaling and translation invariances, shift, uniform scaling, occlusion and complexity invariances. The authors conclude that some problems, like scale and translation invariances, can be attenuated by normalizing the input data beforehand. However, choosing a proper distance metric can resolve the less straightforward ones. Following this, the Euclidean Distance and Dynamic Time Warping (DTW) metrics are the most prevalent distance measures for time-series data. Further details are provided for the state-of-the-art clustering algorithms and time-series averaging techniques, after which the hypothesis is formulated. The authors set out to solve the scaling and shifting invariance problems in it.

The method is based on calculating time-series centroids with the cross-correlation metric. This measure captures the shape of similar signals by ignoring the shifts in phase and amplitude, making it possible to determine the similarity of two sequences even if they are not aligned. In addition, it is concluded that normalising the input data is necessary to achieve the best possible comparisons. After this, a new shape-based distance measure (SBD) is formulated around the cross-correlation statistic and data normalization techniques.

In the following sections, a detailed explanation of the algorithm is provided based on the application of the newly defined distance measure and possible optimizations. The algorithm starts by randomly assigning time-series sequences to clusters and then computes the centroids of each cluster with the *ShapeExtraction* algorithm based on the SBD distance measure. Finally, the algorithm refines the memberships of the clusters with the help of the same distance algorithm. This process is repeated 100 times or until the centroids do not change (convergence).

In the testing phase, the authors compare the SBD measure with the Euclidean Distance and two variations of the DTW measure. Six clustering baselines are selected to compare with the developed algorithm. At the end of the experiments, it was concluded that cross-correlation measures are as competitive as other distance metrics, such as DTW, but are significantly faster. The authors note that the choice of clustering method is as important as the distance measure and that the K-Means with Euclidean distance is the clustering method for time-series data. However, K-Shape outperformed every other state-of-the-art and was significantly faster than K-Means with Euclidean distance.

DPC-KNN [26] is a density peak-based clustering algorithm developed to overcome the shortcomings of the original density peaks clustering (DPC) algorithm [18], which is not capable of detecting clusters with different densities. The authors propose a new method for determining the density of each point based on the KNN algorithm.

In addition to the main problem regarding the density of the clusters, the authors also identify other issues with the original algorithm. These include the difficulty in handling high dimensional data, which tends to confuse DPC by hiding clusters in noisy data, and the algorithm's inability to consider the local geometries of clusters.

To solve this last issue, the authors propose changing the way the density of each point is calculated. In the original algorithm, the density of a point is calculated as the distance between that point and every other point in the data. In the new method, the density of a point is calculated as the mean distance between that point and its k nearest neighbours. This change allows the algorithm to consider the local geometry of the clusters. The dimensionality problem is solved by applying PCA on the previous DPC-KNN, resulting in the DPC-KNN-PCA method.

In the testing phase, the authors perform experiments on several 2D synthetic datasets, in which DPC-KNN achieved perfect scores. In the next experiments, real-life datasets with high dimensionality test the effectiveness of both DPC-KNN methods with DPC and the other baselines. In the end, the authors conclude that DPC-KNN outperforms every other method in terms of accuracy in low-dimensionality data, with data with more than seven features resulting in better results for DPC-KNN-PCA.

Malav et al. [27] employ a K-Means clustering on UCI Heart Disease Data to train an ANN for classifying cardiovascular diseases. The goal was to overcome the limitations of previous implementations in this area and provide a more accurate disease prediction. The main issues in the literature for this application are the low accuracy scores obtained by the classifiers and the use of resource and time-intensive algorithms.

The authors propose a pipeline where they first identify the main attributes commonly associated with heart disease in the data, followed by a preprocessing step, where categorical values are encoded into numeric ones. Attributes used to diagnose the disease are converted to binary values. Measurements with a value higher than the reference are encoded to one, while normal values are encoded to zero. In the next step, the data is clustered with K-Means and the centroids are used to train an ANN. Finally, the trained ANN is used to predict the presence of heart disease in the test set. The results show an increased accuracy when compared to other state-of-the-art approaches.

ClusterNet [28] is a novel point cloud representation which provides 3D rotation invariance. In addition, a deep hierarchical clustering network is developed to better adapt to the new representation.

A novel Rigorous Rotation Invariant (RRI) representation is proposed, which maintains all the necessary information of point clouds, except volatile information associated with the rotation of objects. This representation is achieved by projecting the points onto a 2D plane and then applying a series of rotation-invariant transformations. The architecture consists of three modules; the RRI module generates the desired representation; the cluster abstraction module extracts the features from the RRI representation with the help of the aggregate subcluster features from the agglomerative clustering algorithm applied to the same representation; and finally, the classification module that generates classification scores for each object in a cluster.

Testing was done on several 3D object classification datasets. The authors test the architecture along with other state-of-the-art methods to evaluate the performance of ClusterNet. They also demonstrate the effectiveness of the rotation-invariant representation by comparing the results with and without it. The results showed that ClusterNet, coupled with the RRI representation, performed better in accuracy than the other methods.

DEC [29] (Deep Embedded Clustering) is a method that aims to simultaneously learn feature representations and cluster assignments with the help of autoencoders.

The algorithm is divided into the parameter initialization and the clustering phases. In the first, the data is passed through and DNN encoder to obtain a preliminary latent representation. This is then passed through the K-Means algorithm to determine the initial cluster centroids. In the clustering phase, the centroids are used to calculate the soft assignments of the data points to the clusters. The centroids are then updated with the new assignments and the process is repeated until convergence. In the end, the authors produce a clear separation of the clusters in the latent space.

To evaluate the implemented method, the authors experiment on several image and one text datasets with other clustering methods. The performance of each algorithm is evaluated based on the accuracy calculated through the ground truth labels of each category in the dataset. DEC proved to be a better choice than the other methods for the clustering task.

Fahiman et al. [30] employ the K-Shape clustering technique to improve load forecasting accuracy in electrical infrastructures. The clustering algorithm groups clients with similar consumption patterns, improving the models' accuracy.

The authors intend to work only on a single, smart meter dataset with household consumption records. A preprocessing step is done to scale large numbers and to fill in missing values with interpolations from previous records. They also identify three approaches for load forecasting methods. In the literature. These include a completely aggregated method where the data is used to train a single forecasting model F; a completely disaggregated method which predicts consumption patterns of single consumers by assigning a model for each consumer; and finally, a clustering-based forecasting approaches where clients are grouped into clusters and a model is trained for each cluster. The authors propose a new approach based on the last one that normalizes the weights of the clusters based on the number of clients in each cluster. This then allows for the weighted summation of the predictions of each client, making it possible to group consumption periods in different clusters for each client.

After delineating the method, the authors define the mean absolute percentage error (MAPE) to evaluate the results based on the sum of predictions and the total consumption of each cluster. Next, a feature selection step is done to identify the most meaningful attributes for the task.

In their implementation, the authors employ the time-series clustering K-Shape algorithm with a Multilayer Perceptron (MLP) and a Restricted Boltzmann Machine (RBM). A K-Means clustering algorithm is also used for comparative reasons. After comparing the results, it was concluded that the K-Shape algorithm paired with MLP produced the best results in terms of MAPE. The authors propose developing dynamic clustering algorithms for real-time clustering and forecasting in future work.

Behera et al. [31] devise a new approach for detecting credit card fraud by combining fuzzy clustering with a neural network.

The fraud detection pipeline is divided into three main steps. First, an initial authentication and verification of the credit card attributes are performed. The expiry date, credit card pin and the amount of credit before each transaction are verified.

The Behaviour and Analysis phase evaluates the consumption patterns of the cardholder and checks if the new transaction is in line with the client's history. This is done with the help of the Fuzzy c-means (FCM) clustering method. Each client has its cluster based on previous consumption patterns. When a new transaction occurs, the distance between the new transaction point and the cluster's head is calculated. The transaction is flagged as suspicious if this distance is above a precalculated upper threshold or below a lower threshold for the current client.

After the previous steps, the suspicious transactions are passed through a feed-forward neural network in the Learning Phase to determine whether they are fraudulent.

The method is evaluated on a widely used synthetic transactions dataset. The approach yielded results similar to other implementations for fraud detection in terms of true positive rate but resulted in fewer false positives.

Tang et al. [32] develop an evolved fuzzy neural network (EFNN) that is used for predicting traffic speeds by periodically evaluating traffic flow conditions. The authors' main goal is

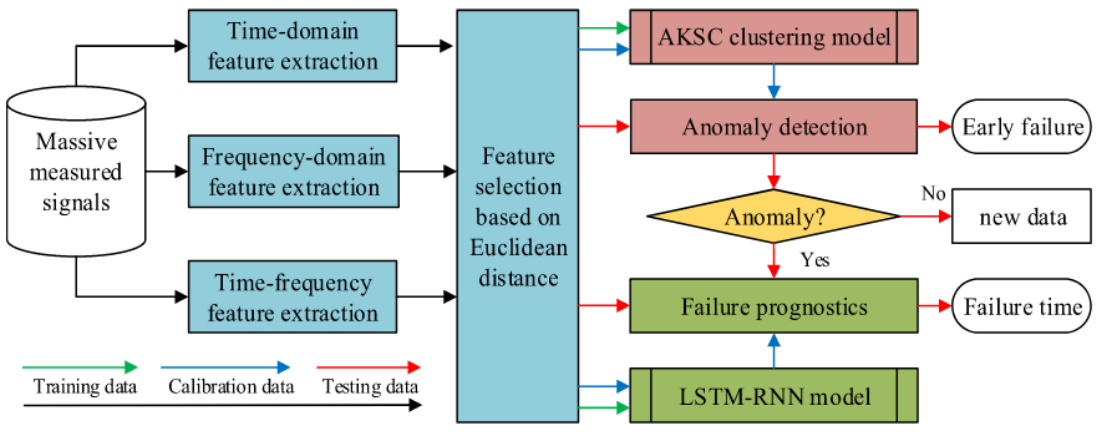


Figure 3.2: Methodology employing AKSC. Taken from [1]

to improve the accuracy of the predictions by employing a fuzzy neural network with a K-Means clustering algorithm and a Gaussian fuzzy membership function. The data for this study consists of historical data collected on a busy Beijing road section.

EFNNs are an improvement over the previous fuzzy neural networks, as they can continuously evolve structure and functionality, making them more suited for real-life continuous problems. EFFNs have a distinct learning process divided into unsupervised and supervised phases. First, the K-Means clustering method is applied to the input samples to determine the cluster centroids. In the supervised step, a set of k fuzzy rules (one for each cluster) is generated to define the membership criteria of each cluster. In this study, the authors choose Gaussian-type fuzzy functions to determine memberships.

In the proposed method, the speed data collected in the last 21 days is passed through a regression function to determine a daily periodic pattern for traffic speed. This is based on the assumption that traffic speed periodicity is cyclical. After this step, a speed function based on the current time t is defined with the help of the periodic component from the previous step. The residual part of the regression is used as a training dataset to optimize the EFNN parameters and predict residual errors in the next step. The method can accurately predict values for real-speed data with the combination of the regression component and the residual error.

The resulting system is evaluated on three performance metrics and multiple traditional prediction statistical models. The authors conclude that EFNN+CP (EFNN with cyclical periodicity) is better than the other methods, especially when forecasting multiple steps.

AKSC [1] (Adaptive Kernal Spectral Clustering) is a novel algorithm to identify machine anomaly behaviours in machine health monitoring. The developers aim to apply the new algorithm and an LSMT-RNN network to improve its accuracy and efficiency in data with high dimensionality.

The authors describe the method in three steps (Figure 3.2). The first consists of feature extraction and selection to identify the features that indicate degradation phenomena from measured signals. In the anomaly detection step, the AKSC method is applied to the selected features to adaptively identify anomalies in real time. The final step is to run a failure prognostic to continuously update and predict the failure time of the machine with the help of an LSTM-RNN network. The failure prognostics and anomaly detection results are used in conjunction to refine the predictions of the LSTM-RNN network and to produce more accurate failure times.

The proposed clustering algorithm uses an iterative and adaptive spectral clustering method. It maps original features into a new feature space to find complex non-linear cluster boundaries and identify slight changes in the data. AKSC is divided into three stages. These are initialization, where kernel spectral clustering is applied to the data; the calibration stage, where the clustering parameters are updated to improve future identification accuracy; and the detection stage, where an outlier indicator is defined to identify anomalies in the machine.

The authors create a real-life bearing test to evaluate the method, simultaneously measuring the vibration and temperature of four bearings with different conditions. At the end of the tests, it was concluded that the developed approach accurately predicted the mean time to failure of the rolling bearings. In addition, the authors also point out that the method could be applied to various other industrial contexts due to its generalizability.

Jahangir et al. [33] employ a micro-clustering technique based on K-Means and Gaussian SVM along with a Bi-LSMT to develop a reliable forecasting system for electrical grid parameters.

Micro-clustering is applied to data sequences collected in the last hour, t, and creates clusters that will then be used to train Bi-LSMT networks for forecasting three network profiles (variables). In the unsupervised clustering phase, the authors execute several K-Means algorithm computations over the data they are trying to cluster by varying the number of clusters. The most appropriate K-Means model is based on the Davies-Bouldin (DB) index score. In the final stage, a Bi-LSTM network is trained for each cluster produced for the given hour. This process is repeated until all 24 hours of the day are covered.

After having models for each of the variables for each hour, t of the day, the pre-trained models from the previous stage are used to forecast the network profiles for the next days. For each hour t new measures are assigned to the cluster centres from the previous stage with the help of a Gaussian-SVM. Then the Bi-LSMT for that cluster at the time t is fetched to forecast the desired parameter.

The resulting method is evaluated on data sequences collected in Ontario due to the prevalence of wind power in the network, which tends to cause fluctuations in the grid. The robustness of the approach is compared with other statistical and machine learning techniques, each of which is evaluated based on mean absolute error (MAE), mean absolute percentage error (MAPE) and root mean squared error (RMSE). After an extensive evaluation of the results, the authors conclude that the devised method was more robust than the other baselines and could be used in real-life situations to forecast conditions in power systems.

Zhang et al. [34] propose a novel patient-specific electrocardiogram (ECG) classification algorithm based on RNNs and density-based clustering techniques.

The algorithm framework consists of fetching normal ECG data based on density clustering results that assign types of heartbeats to different classes and then training a common RNN model to classify these classes. A model is created for each patient based on his history by clustering to previous ECGs and then training an RNN classification model with feedback from the common model. In the end, new patient records can be fed to the patient-specific model to classify the heartbeat type.

The authors evaluate the method by comparing it with the results of previous state-of-the-art ECG classification methods. The approach consistently outperformed the others regarding accuracy, specificity, sensitivity and positive predictivity.

Yan et al. [35] develop a cluster-based pooling method to reduce the overfitting and increase data diversity in a Bayesian deep learning-based probabilistic load forecasting (PLF) model.

The framework is divided into the initialization, multitask Bayesian deep learning and probabilistic forecasting stages. In the first step, household smart meter data is clustered into k clusters based on their load consumption patterns. In the next stage, a multitask Bayesian neural network (MT-BNN) is trained using load profile pools from the previous step until model performance stabilizes. If equilibrium is not reached, then the algorithm goes back to the first step with an increased number of clusters as long as a maximum number of iterations is not reached. In the forecasting phase, probabilistic forecasting on the selected customers is performed to assess the predictive performance.

The resulting system is evaluated in terms of MAE and RMSE on a large household smart-grid dataset. The MAE and RMSE scores were identified as the main evaluation criteria. MT-BNN is compared with Global and Separate BNN models according to the data pooling strategy. The results showed that the MT-BNN model outperformed the other two in terms of the defined error measures. Similarly, a benchmark is performed on several other statistical models with the proposed method outperforming every other on the same metrics.

3.1.5 Analysis

A brief analysis of the clustering SLR results is presented in Table 3.2. The following categories were created for a summarization of the papers that were analysed:

- **Type:** This category indicates the type of article. This can be either a survey, a review or a primary study.
- **Application:** This class indicates the application area of the article. This can be either a general application, as is the case for most clustering algorithms, or a specific one.
- **Objective:** This category indicates the main objective of the article. This can be either the development of a clustering algorithm, a cluster decision method, or the improvement of a machine learning task via clustering.
- **Dimensionality Reduction (DR):** This category indicates if the article uses DR techniques (can be either "yes" or "no"). Dimensionality reduction techniques may be important for this study, as the data being used is high-dimensional.

The results show that most papers are primary studies (17 out of 20). The remaining three are two surveys and a review. The surveys include the introductoImprovery article for the clustering theme ([17]) and another one that focuses on clustering algorithms for high-dimensional datasets ([19]). The review article ([22]) focuses on the use of principal component analysis (PCA) for the identification of bioactive compounds.

As was expected, most articles that present new clustering methods/algorithms are mostly applied to general cases. The only exceptions are the two surveys, which were to be expected as they tend to have a broader scope by discussing several possible approaches to multiple applications. The remaining articles are mostly prediction tasks, with two applied to the banking sector [20, 31] and several others for forecasting system/network conditions [30, 32, 33, 35].

Only four novel clustering algorithms surfaced for **C_SQ1** [25, 21, 18, 26]. Of these, three are density-based, and one is suited for time-series data. From this, it can be concluded that most papers use preexisting clustering algorithms for the tasks.

Jain [17] and Fahad et al. [19] provide an overview of the types of evaluation methods that we were trying to obtain from **C_SQ2**. In most approaches, external criteria are used to evaluate the clustering; however, in cases where there are no labelled datasets, these cannot be used. For this problem, Kou et al. [20] devise an MCDM method to choose the most appropriate based on several validity measures (which might include external metrics).

Regarding **C_SQ3**, about half of the papers (11/20) aim to develop a new methodology for improving machine-learning approaches, either by creating cluster-based sampling methods or training separate models for each cluster.

Only six papers directly apply dimensionality techniques to the data. This is mostly seen in approaches where the data is high-dimensional, and the authors want to reduce the number of features to improve the performance of the clustering algorithm. The remaining papers do not apply dimensionality reduction to the data; however, most mention the possibility of such methods in improving the results.

Figure 3.3 shows the distribution of the results through the years. Note that the introductory paper [17] is not included in the plot as it was added to the results after the SLR. The plot shows

Paper	Year	Type	Application	Objective	DR
Jain [17]	2010	Survey	N/A	Cluster Algo. + Analysis	No
Rodriguez et al. [18]	2014	Primary	General	Cluster Algo.	No
Fahad et al. [19]	2014	Survey	High Dimension Datasets	Cluster Algo. + Large Data	Yes
Kou et al. [20]	2014	Primary	Financial Risk Assessment	Cluster Decision Method	No
CAN [21]	2014	Primary	General	Cluster Algo + Large Data	Yes
Granato et al. [22]	2018	Review	Bioactive Compound Identification	Method Assessment	Yes
Lin et al. [23]	2017	Primary	General	Improve ML (Sampling)	No
Douzas et al. [24]	2018	Primary	General	Improve ML (Sampling)	Yes
K-Shape [25]	2015	Primary	Time Series Clustering	Cluster Algo.	No
DPC-KNN [26]	2016	Primary	General	Cluster Algo. (Improve)	Yes
Malav et al. [27]	2017	Primary	Heart Disease Monitoring	Improve ML	No
ClusterNet [28]	2019	Primary	Object Classification (3D point cloud)	Improve ML	No
DEC [29]	2016	Primary	General	Cluster Algo.	Yes
Fahiman et al. [30]	2017	Primary	Electrical Grid Load Forecasting	Improve ML	No
Behera et al. [31]	2015	Primary	Bank Fraud Detection	Improve ML	No
Tang et al. [32]	2017	Primary	Road Network Load Forecasting	Improve ML	No
AKSC [1]	2019	Primary	Machine Health Monitoring	Improve ML	No
Jahangir et al. [33]	2021	Primary	Electrical Grid Forecasting	Improve ML	No
Zhang et al. [34]	2017	Primary	Eletrocardiogram Monitoring	Improve ML	No
Yan et al. [35]	2020	Primary	Electrical Grid Load Forecasting	Improve ML (Sampling)	No

Table 3.2: List of reviewed papers for the clustering SLR.

that most selected papers are from 2014 or 2017, with only two papers from 2020 and upwards and no early publications. A possible explanation for this will be provided in the next section.

3.1.6 Threats to SLR Validity

The lacklustre results from the first query (see Section 3.1.2) were unexpected. The search query was designed to retrieve the most relevant articles for the topic. However, the results were not satisfactory. Most relevant articles were older than the rest as the sorting criteria were set to

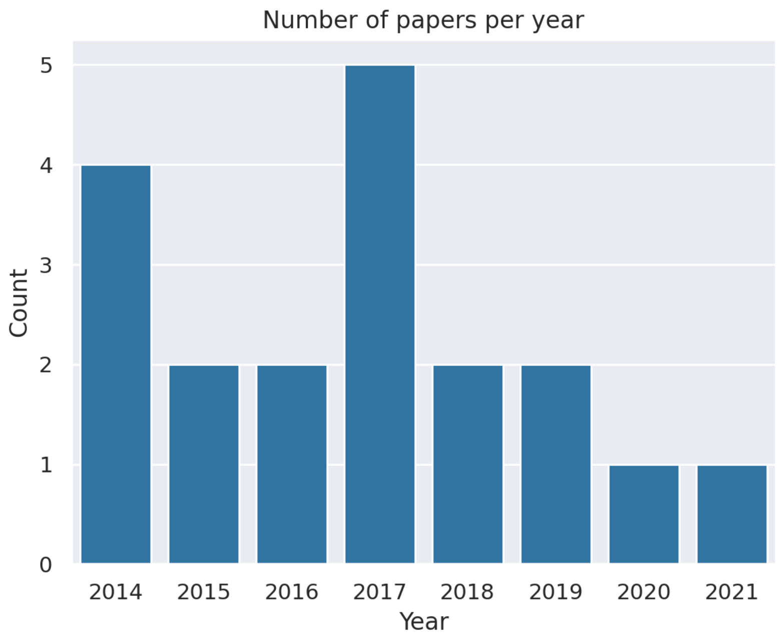


Figure 3.3: Number of papers per year for the clustering SLR. Does not include the above-mentioned introductory paper from 2010.

citation count, which inherently gave more relevance to these papers. Consequently, most papers retrieved from Scopus are older than the ones from Google Scholar. In addition, most of the papers didn't relate to applying clustering techniques to machine learning tasks, as these terms were not directly specified in the query. To rectify this, another search similar to the one in Google Scholar was done in Scopus; however, the results were still very similar to the ones from the first query.

The second search yielded slightly better results than the previous one, with more recent articles focusing on applying clustering techniques to machine learning tasks. The results were sorted based on their "relevancy," which, according to the chosen platform, indicates the best match between each document and the search terms. Consequently, the first results weren't as skewed to older documents as in the first query.

3.2 Adversarial Anomaly Detection

This section explains the SLR for adversarial anomaly detection and is organized in the following manner. The survey research questions used to guide the search process are in Section 3.2.1, followed by the search query in Section 3.2.2. Section 3.2.3 explains the criteria used to filter the results, which are presented in Section 3.2.4. Finally, a brief analysis of the results is done in Section 3.2.5, followed by the threats to the whole research process (Section 3.2.6).

3.2.1 Survey Research Questions

Two search questions were defined to identify the most relevant articles that closely align with the requirements of this work:

G_SQ1 *What are the current adversarial learning architectures for anomaly detection in tabular data?* Most adversarial learning approaches (like GANs) are designed for problems with image datasets. However, the dataset used in this study consists only of tabular data. The intention is to include only architectures designed or adapted for anomaly detection in tabular datasets.

G_SQ2 *What components of the architectures can be used for the anomaly detection phase?* We know that in GANs, the objective is for both the generator and discriminator to learn the normal distribution of the data. In most applications, the generator is picked to create new samples from random latent representations and the discriminator is often discarded. With this question, we are trying to understand what state-of-the-art application use only the discriminator as a detection mechanism.

3.2.2 Search Query

The search questions defined in the previous sections were aggregated into a single search query. The query construction was done incrementally to refine the search results to address the specific questions of interest. In the end, the search query was the following:

(gan* OR adversarial learning OR generative adversarial net*)
AND ((anomal* OR outlier? OR abnormal OR novel*) AND detect*)
AND NOT (imag* OR video* OR segment* OR photo*)

The first part consists of a mixture of terms associated with GANs and intendeds to only retrieve articles with one of those terms in the title, abstract and keywords. The second restricts the results to adversarial architectures for anomaly detection in the same three fields as the previous one. Several synonyms for anomaly were used to increase the number of relevant papers. The final term was only applied to the documents' keywords and was intended to exclude GANs and other architectures applied to image datasets. All articles were retrieved from Scopus [5].

[5]Scopus: https://www.scopus.com/

Criteria	ID	Description
Inclusion	I1	The document focuses on GANs for anomaly detection.
	I2	The results are clear, and the proposed goals are achieved.
	I3	The authors provide code or an extensive explanation of the architecture.
	I4	Provides a comparison of the developed GAN with other baseline models (not necessarily GANs).
Exclusion	E1	The article was cited less than 6 times. For earlier publications, the number of citations was reduced to half.
	E2	Surveys and reports on works carried out by other authors.
	E3	Does not use tabular data. Either it has one or more unwanted terms in the title (e.g. image "photo") or only performs tests on image datasets.
	E4	Was published before 2014.

Table 3.3: Inclusion and Exclusion criteria.

3.2.3 Inclusion and exclusion criteria

The query defined in the previous section yielded 1489 results, making it impractical to analyze each one manually. A set of criteria was determined to reduce the number of documents that needed to be studied (see Table 3.3). Note that E3 only exists because both search queries failed when the documents did not indicate the use of images in the keywords. These were later used with several steps to exclude non-relevant papers and narrow the state-of-the-art analysis iteratively.

Figure 3.4 illustrates the processing pipeline. 1489 results were retrieved from Scopus with the defined query. The first processing setup applies exclusion criteria *E1* to remove papers with little to no citations, which resulted in 168 documents. 61 were left after a preliminary title analysis with the criteria (I1; E2-E3). In the final step, a preliminary analysis of the remaining documents' abstracts and conclusions was undertaken to only select the most relevant to the defined search questions. The inclusion criteria for this step were I1 to I4. In addition, the documents that were surveys or reviews of multiple implementations and articles that did not deal with tabular data were excluded. In the next section, the resulting papers from this last step will be explained.

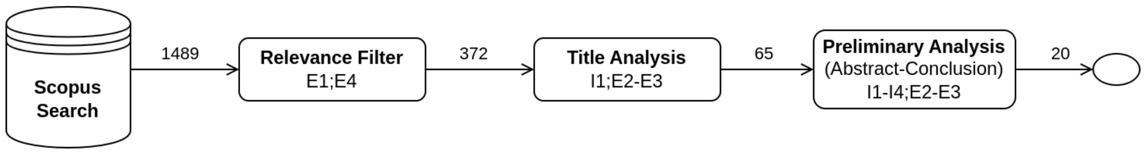


Figure 3.4: GAN Systematic Literature Review Pipeline

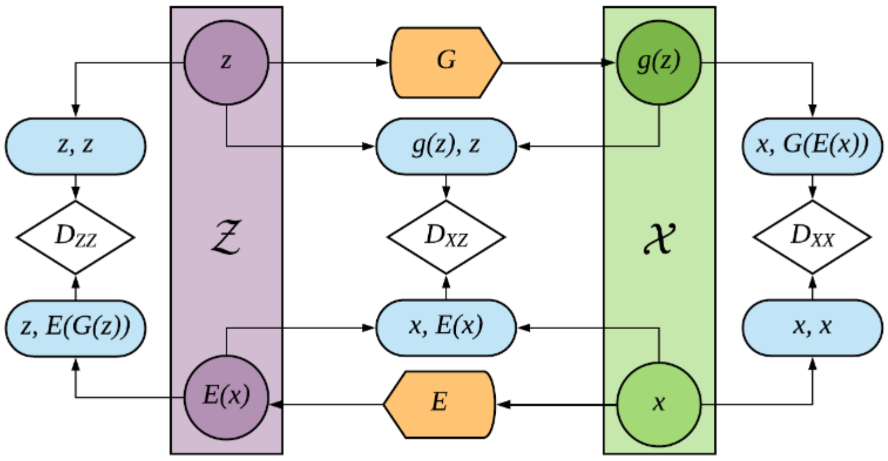


Figure 3.5: ALAD Architecture. D_{xx}, D_{xz} and D_{xx} are the discriminators (white), Z (purple) and X (green) represent the latent and data spaces, respectively; G and E (orange) are the generator and the encoder, respectively. Reprinted from [38].

3.2.4 Results

MAD-GAN [36] is an architecture for anomaly detection in multivariate data with spatiotemporal correlations. The generator and the discriminator are composed of Long-Short-Term Recurrent Neural Networks (LSTM-RNN). As is usual in other GANs, the generator creates fake samples from a vector of latent points. It feeds them to the generator, which aims to distinguish generated from the original samples. However, instead of just using the discriminator to detect abnormal samples in the testing phase, the authors propose employing the generator for the same task. The theory for this is that by generating correct samples, the generator can learn the normal distribution of the dataset.

During the test phase, the discriminator receives a sample from the test dataset and performs the same classification as in the previous stage. However, the generator will receive a version of the sample mapped to the latent space and be tasked with reconstructing the sample. Next, the reconstruction error is calculated by comparing the reconstructed sample with the original one. This error and the discriminator outputs are used to compute the *Discriminator and Reconstruction Anomaly Score* (DR-Score). A sample is considered abnormal if it has a DR-Score higher than a predefined value τ.

The developed architecture was compared with five baselines. These include PCA, K-Nearest Neighbours (KNN), Feature Bagging (FB), an Autoeconder (AE), and the *Efficient GAN* (EGAN) [37]. The tests were performed on three intrusion detection datasets. MAD-GAN was able to outperform the other baselines on almost all datasets consistently.

ALAD [38] is a reconstruction-based anomaly detection architecture that employs multiple bi-directional GANs. The proposed method, *Adversarily Learned Anomaly Detection* (ALAD), intends to use both the discriminator and the generator for the task. The ALAD architecture is shown in Figure 3.5.

An encoder network E maps data samples x into the latent space z during training. Several additional discriminators are used to achieve cycle consistency (to ensure that the reconstructed samples resemble the original ones) and to stabilise the model. D_{xz} is an improvement from other similar solutions that solve the saddle-point problem by ensuring cycle consistency, which is not always the case when using encoders. Further, entropy regularisation is imposed on both G and E by the discriminators D_{xx} and D_{zz}. The latter receives two pairs of latent points and must distinguish the real (z, z) from the synthesized one $(z, E(x))$; the former follows a similar logic but with examples extracted from the dataset.

The authors propose a new score function for anomaly detection that captures the confidence of D_{xx} when distinguishing real from synthesized pairs. This is because a poor-quality reconstruction would indicate that the generator did not learn how to reconstruct that sample and, consequently, should be considered an anomaly. Finally, the designed model was compared with five standard anomaly detection methods and another GAN for anomaly detection on two anomaly detection datasets. The developed model outperformed the baselines on one of the datasets but could not do so on the other. This was because this dataset had a small number of samples, and ALAD, like other GANs, requires large amounts of training data.

USAD [39] is an architecture based on adversarially trained autoencoders for anomaly detection in multivariate time series data, more concretely, logs from IT systems. The authors proposed solving the convergence and mode collapse problems experienced in other GANs. USAD is composed of one encoder E and two decoders $D1$ and $D2$, which in conjunction with the encoder, result in two autoencoders ($AE1$ and $AE2$). E takes data samples as windows W and encodes them to latent space vectors z. The function of the decoders is to reconstruct the samples in those windows.

The autoencoders are trained with normal samples to learn the data distribution during this phase. In the detection stage, both autoencoders are trained adversarially. $AE1$ reconstructs samples from the real dataset and $AE2$ must distinguish examples reconstructed by $AE1$ from the real data. The anomaly score is calculated based on the reconstruction errors obtained on both autoencoders. The proposed model was evaluated along with other unsupervised methods for anomaly detection on intrusion detection datasets. USAD outperformed the other baselines in terms of F1-Score.

MO-GAAL [40] as the goal of generating informative outliers to overcome the significant class imbalance and lack of correct labels in outlier detection datasets. The authors developed two proximity-based outlier detection methods that do not rely on previous knowledge of the dataset. The first was named *Single-objective Generative Adversarial Active Learning*

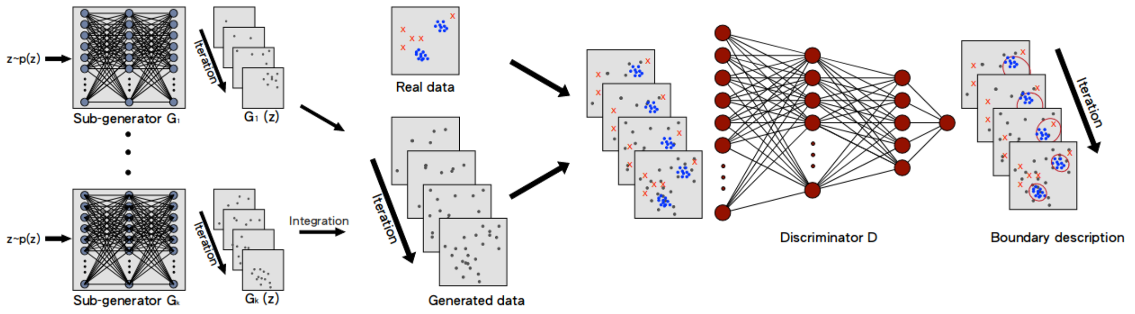


Figure 3.6: MO-GAAL Architecture. Each generator G_i to G_k (left) generates outliers for the respective cluster; The discriminator D (right) aims to draw increasingly smaller boundaries around the real data distribution. Taken from [40].

(SO-GAAL). Like other GANs, it plays the same mini-max game between the generator G and the discriminator D. However, the objective for G is to produce outliers that occur inside or close to the real data. Similarly, the goal of D is to create a division boundary that separates the real data from potential outliers. G gradually learns the generating mechanism and synthesizes an increasing number of potential outliers, and D gets better at creating the divisions that enclose the real data. The point of generating outliers is to create a reasonable reference distribution for the real data.

Despite providing good results, the first model proved to be very unstable due to the problem of mode collapse. At some point, after a good amount of iterations, the precision of the model would greatly diminish. Another technique called *Multiple-objective Generative Active Learning* (MO-GAAL) was designed to solve this issue. A general workflow of the architecture can be seen in Figure 3.6. The authors proposed generating outliers for specific real data subsets using a generator for each cluster in the dataset (which requires cluster identification). This solved the mode collapse problem on the first model and stabilized the performance.

Both architectures were tested on fourteen datasets and ten other baseline outlier detection methods. Despite other methods performing better in specific datasets, MO-GAAL proved more reliable even on non-cluster datasets.

IGAN-IDS [41] or *Imbalanced Generative Network* was designed to cope with class imbalance problems that other outlier methods for intrusion detection suffer from. IGAN, which can be seen in Figure 3.7 (middle module), comprises an imbalanced data filter, Generator G, and a Discriminator D. Each sample, $s = (x, y)$, is a vector containing the values and the class labels. The imbalanced filter takes only samples from the minority classes, denoted as $s' = (x', y')$. It calculates the generating factor k for each class (ratio of samples that should be generated for each minority class). G receives a set of latent points z and the class label y' and outputs a vector with a generated value $G(z, y')$ from the class label that it received. This vector is then passed to the discriminator, which aims to distinguish synthesized feature

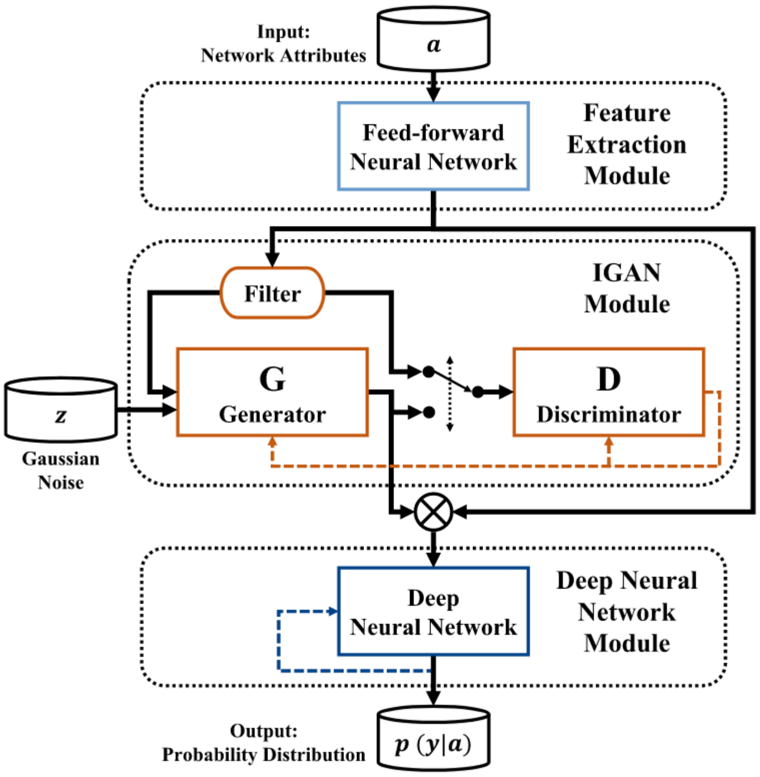


Figure 3.7: Full IGAN-IDS architecture. Taken from [41].

vectors from the ones extracted from the dataset. In the training process, *G* and *D* are trained alternatively. First, *D* is fed only real samples while *G* is fixed, and in the next iteration, *G* is optimized, and *D* is fixed.

With the problem of class imbalance dealt with, the authors set out to perform intrusion detection with IGAN-IDS (Figure 3.7). The feature extraction module (top) embeds discrete data into one-hot encoded vectors and discretizes continuous variables, which are encoded. All values are concatenated and fed to IGAN, which generates samples for the imbalanced classes. Finally, a DNN (bottom module) is used for outlier detection. In the training stage, it receives both synthesized and real data and, during the testing phase, calculates the distributed probabilities for each inclusion class of each sample. The proposed solution was tested with several other class-balancing techniques on three standard intrusion detection datasets. It outperformed every method in precision, accuracy, recall and AUC score.

Jiang et al. (2019) [42] propose a conditional GAN architecture to detect anomalies in univariate Industrial Time Series Data. The model is trained with only normal samples. The dimensionality of the original data was reduced by employing a feature extractor. The generator consists of two encoders G_e and $G_{e'}$ and an intermediate decoder G_d. The first encoder maps real samples into the latent space z while the decoder G_d is tasked with reconstructing the encoded sample back to the real data space.

During training, the GAN is only tasked with reconstructing normal data samples so that both components learn the normal distribution of the dataset. Two loss functions are defined for the generator, the *Apparent loss* and the *Latent loss*. The first measures the distance between the original and synthesized samples, and the other measures the distance between the latent vectors z and z encoded by G_e and $G_{e'}$, respectively. The loss function of the discriminator compares the feature vector from the actual sample $f(x)$ with the synthesized one $f(G(x))$. The anomaly score is defined as the sum of the two losses.

In the testing phase, two rolling bearing datasets were used. The authors fine-tunned the models by adjusting the hyper-parameters of the network. A significant difference in anomaly score $A(x)$ was observed for faulty parts in the dataset, which proved the efficacy of the designed model. A comparison was also performed with the state-of-the-art BiGAN [43], which showed that the developed GAN was more reliable on datasets with differing sizes.

TadGAN [44] aims to solve the problem of scalability and portability in state-of-the-art unsupervised methods for anomaly detection. Its goal is to detect anomalies in time series datasets. The authors used LSMT Recurrent Neural networks for both the generator and the critic. Two types of anomalies are identified single point (abnormal data point) and collective (sequence of abnormal data points) anomalies.

The proposed architecture is a reconstruction-based anomaly detector with a generator G which receives encoded samples in the form of latent points z and reconstructs them back to the original sample distribution; an encoder that takes samples in the normal distribution and encodes them into latent point vectors; and two critics, one to distinguish real data points from synthesized ones (C_x) and the other (C_z) to evaluate between the distribution of real z and the ones that were encoded by E ($E(x)$). To cope with the mode collapse problem, common in most GANs, the authors adopt the Wasserstein loss function (for critics) and a cycle consistency loss function (for the generator and encoder). For the reconstruction errors, the point-wise difference (distance between the real and synthesized point) and area difference (distance between "windows" of the same area in the real and synthesized data) were defined. Furthermore, the authors also chose a *Dynamic Time Warp* (DTW) measure, which, similarly to area difference, can identify minor differences over long periods but can also handle time shift issues.

Two methods of combining the critic outputs with the reconstruction errors were devised. For the first, a weighted sum of the reconstruction error and the critic output is done; in the other method, both values are directly multiplied. Several baselines are chosen to compare with the developed model in the testing phase. TadGAN and the baselines were tested on eleven datasets for anomaly detection (two of which were from NASA). The developed network outperformed every other baseline on six of the eleven datasets (based on the F1-score). Despite this, the mean, standard deviation and average of the F1-score in all datasets showed that TadGAN was more reliable than the other methods. Finally, the authors defined ten iterations of TadGAN, each with a different anomaly score with either one reconstruction function, one critic output or a combination of the two. The worst result was with the anomaly function consisting only of the critic output, and the best was the one in which the critic score and DTW were multiplied.

adVAE [45] employs a Variational Autoencoder (VAE) for anomaly detection. The authors propose an encoder E that encodes real samples into random point vectors z, fed to a generator G that is then tasked with synthesizing examples close to the real distribution of the dataset. Competition is introduced in the form of a Gaussian transformer T that receives the encoded vectors from E and is tasked to generate latent vectors z_T with a similar distribution to z (outliers). G is tasked with generating as different as possible examples from both similar latent vectors. Finally, the examples from G are encoded again by E, and the resulting distributions are compared with the original ones. To make the equilibrium of these three models feasible, the authors freeze the gradients of E in the first training phase to only train the G and T. This way, T will generate abnormal latent variables close to the real distribution, and G will be able to distinguish them using reconstruction errors.

In the second phase, both G and T have their gradients frozen, and the encoder is trained to encode the inputs as close as possible to the prior distribution (only if the inputs are from the dataset and do not result from Gaussian variables generated by T). In the testing phase, the trained encoder and generator are used to detect anomalies by calculating the reconstruction error of the input samples. The solution was evaluated on tabular anomaly detection datasets and several other state-of-the-art outlier detection methods. Furthermore, several ablation models from adVAE were derived by removing the discriminative factors of either the generator or the encoder. The results showed that adVAE and its variations outperformed most baselines on the chosen datasets regarding precision and AUC score.

Blance et al. (2019) [46] propose using adversarially trained autoencoders as a way of improving the separation between background from the signal in synthesized high-energy collision events. The authors train an NN that can distinguish signal events from the background and intentionally smear the background data in distinct directions. With this, they prove that the classifier's performance is highly dependent on the smearing of the background samples.

An adversary is used to try and remove the dependence of the classifier on the smearing of samples. Both are forced into a zero-sum game in which the classifier must learn how to make predictions without using any information from smearing and tries to make it as hard as possible for the adversary to discriminate the background samples. The classifier receives samples from the dataset and sends its outputs to the adversary, which tries to determine the background class based on the outputs. The results showed that this method significantly reduced the dependence of the classifier on the smearing of background samples.

In addition, the authors propose another method in which they use an autoencoder alongside an adversary. The function of the autoencoder is to reconstruct only background samples, and the adversary is tasked with identifying them. As the autoencoder only learns the distribution of background events, it will not be able to reconstruct signal events as well (i.e.

signal events will be considered outliers). The adversary takes as input the loss of the AE and tries to determine the background smear class. The results show that the method could remove the dependence between the autoencoder classification and the smear direction of the background samples. Despite this, this architecture proved less effective than the previous one.

FGPAA [47] is an adversarially trained autoencoder that aims to monitor the conditions of roll-bearings by analyzing the vibration signals. The model consists of four components, a discriminator D, a generative discriminator GD, an encoder E and a Low-dimension discriminator LD. E takes one signal at a time and encodes it into the low-dimension manifold (latent space) z. LD discriminates if the output of the encoder follows the same distribution as the latent space z. The latent points vector z is passed to the encoder that works as a generator by synthesizing high-dimension signals from the low manifold distribution. The generative discriminator, GD, tries to distinguish reconstructed signals from those originating from the dataset.

The anomaly score is calculated for each sample by comparing the distributions of the generated low-dimension manifold and the reconstructed sample with the distributions from the actual dataset. This score is then used during detection to identify signal data faults. The proposed solution was tested on three roll-bearing datasets alongside well-known ML anomaly detection methods. FGPAA outperformed all in terms of F1-score, but they displayed a higher execution time than the rest.

FGAN [48] is an architecture close to original GANs but with a modified loss function more suited for anomaly detection. The authors propose adapting the model's objective so that the generated samples lie close to the boundaries of the real data distribution instead of overlapping it. The objective is to generate data around low data density regions δX around the real dataset.

The authors use the discriminator score to identify the domain of δX and then estimate it with the generator. At the end of the training phase, the synthesized points must enclose the entirety of the data. This goal is achieved by replacing the typical loss of the generator with the *Generator Encirclement Loss*, which penalizes points generated inside or far away from the real distribution. *Generator Dispersion Loss* is also introduced to guarantee that the generated points enclose the whole distribution and not just a single area (similar to mode collapse in other methods). It maximizes the distance between points by penalizing the generator if the synthesized points are too close to each other. The resulting loss function is a weighted sum of the two proposed ones. Similar to the generator, the loss function of the discriminator is also modified to prioritize classifying real data correctly by reducing the second term of the original discriminator function (refer to equation 6.1).

To evaluate the developed mode, the authors tested its performance on a synthesized 2D dataset. Next, the proposed solution was tested on image and tabular datasets and other state-of-the-art models. FGAN outperformed all baselines for anomaly detection on both types of datasets.

MENSA [49] is an autoencoder-based GAN architecture used to detect intrusions on next-generation Electrical Grids (also known as Smart Grids). Furthermore, the proposed model can also detect and classify different classes of cyberattacks that often occur on the TCP (Transport Communication Protocol) and the DNP3 (Distributed Network Protocol 3) protocols.

The architecture consists of a Generator-Encoder and a Discriminator-Encoder. The first receives input noise samples and inflates them to produce samples that resemble the desired data to learn the normal distribution. The Discriminator-Encoder then compresses the synthesized samples into a single point that indicates if the sample is from the real dataset or is a fake. For the detection phase, the Latent Model is derived from the initial layers of the discriminator. The generator, having learned the normal distribution of the data, tries to reconstruct samples from the real dataset. These are then passed to the Latent Model that calculates the Adversarial Loss score by comparing the real sample with the reconstructed one. Note that the generator will fail to reconstruct abnormal samples, as they were not a part of the training process.

The classification model is derived from the previous architecture, in which the Discriminator-Encoder also learns to classify the attack class of a given sample. The generator learns to generate samples based on the class labels conditionally. Similar to the previous implementation, the discriminator receives the synthesized samples and identifies them as real or fake, but this time, it also tries to determine the class label.

The proposed solution was tested on real Smart Grid datasets and several other intrusion detection methods. The Accuracy, True-Positive and False-Positive rates and F1-Score were calculated for each solution. MENSA outperformed other methods on all datasets except for one.

Liu et al. (2022) [50] propose a deep feature enhanced generative adversarial network to improve fault detection performance in roll bearing imbalanced datasets. New and preexisting methods are introduced to solve mode-collapse during training and enhance the feature learning of the network, which aim to increase the overall detection performance of the architecture. The adopted methodology is shown in Figure 3.8.

A new loss function is designed for the generator with a *pull-away* term. It measures the distance between the generated samples inside a given batch and penalizes the generator if the batch's samples are too similar. As a consequence, this solves the mode-collapse problem during the training phase. A self-attention model is introduced to the discriminator and the generator so they can learn the features of the original vibration signals more deeply. The self-attention feature maps capture local details and global information in every layer of the network.

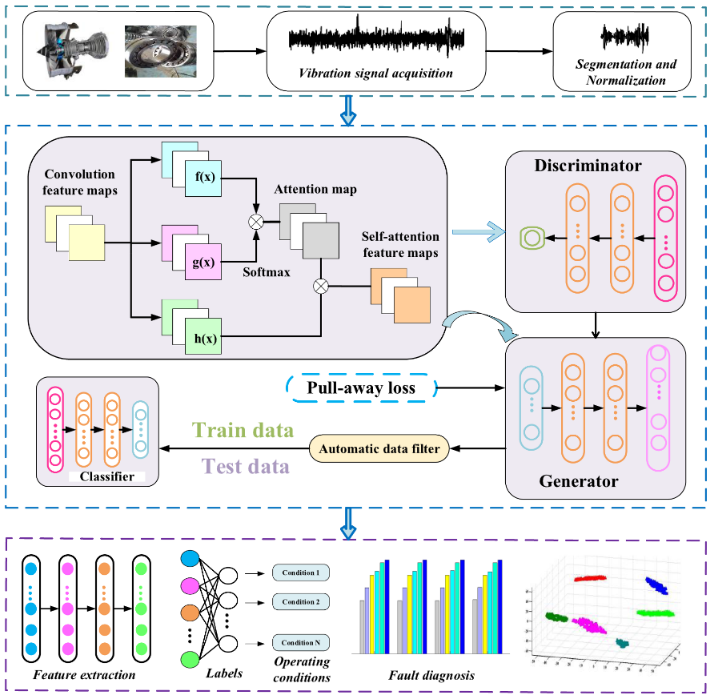


Figure 3.8: Methodology for anomaly detection in roll bearing datasets. Taken from [50].

During training, the generator synthesizes signals from random noise fed to the discriminator along with an actual signal. The discriminator must then distinguish which one is real and which one is fake. With the proposed mechanisms, the generator must synthesize samples as far as possible to each other to confuse the discriminator. The training phase stops when three criteria are met. These are defined by the *automatic data filter*, which evaluates the accuracy and diversity of the generated samples based on discriminator probability, Kullback-Leibler (KL) divergence and maximum mean discrepancy. When these criteria surpass a predefined threshold, the training phase is concluded. The fault detection phase uses a classifier trained on the generated data.

The quality of synthesised signals was compared with other standard generative models using KL divergence and maximum mean discrepancy of the *automatic data filter*. The proposed method surpassed the others in these metrics. Next, three experiments were set up to evaluate the detection quality of the proposed method. In each experiment, three classifier models were used. In the first, the classifiers learned to detect anomalies from the original signals dataset. In the next, a WGAN-GP architecture was used to generate a more balanced signals dataset. Finally, the three classifiers were trained on data synthesized by the proposed method in the last experiment. This methodology was performed on laboratory and locomotive roll-bearing datasets. Every model trained on the generated data (for both datasets) from the designed solution outperformed the models from the other experiments.

DOPING [51] is an adversarial autoencoder that aims to improve the performance of unsupervised anomaly detection by oversampling *infrequent normal samples*. With this, the authors intend to reduce the number of false positives that usually occur on datasets with normal samples close to the classification boundary (close to but not anomalies).

In the training phase, the autoencoder receives samples from the entire dataset distribution. The latent vectors generated by the encoder, E, from the data samples, X, are saved for the next stage, in which only the latent variables Z at the tail-end distribution of the normal data are sampled into a pool of Z_{edge}. From this pool, z_{edge} variables are sampled randomly and interpolated to generate new latent vectors z_{new}. In the next stage, de decoder D synthesizes minority samples from the z_{new} latent vectors. Later the synthesized examples can be used in conjunction with the original ones for anomaly detection.

An isolation forest anomaly detector was used to evaluate the effects of the oversampling method on the detection performance. The authors used three synthesized cluster datasets with outliers, the popular MNIST image dataset, and four real medical record datasets. The results showed that the outlier detection method in conjunction with DOPING achieved better results than the baseline (detection without DOPING).

Bot-GAN [52] aims to improve the detection performance of botnets in network-flow data. The architecture is similar to the one from vanilla GAN. However, the authors chose a botnet detection model as a discriminator. Like vanilla GAN, it receives examples from the training dataset and synthesizes ones from the generator. The main difference is that the discriminator classifies each sample as normal (from the training dataset), an anomaly or fake (either synthesized or from the dataset). The generator's objective is to synthesize more samples similar to the ones in the dataset to assist with the training of the detector.

To test the proposed approach, choose a botnet dataset and perform some preprocessing to normalize the formats of the entries and map the features into vectors. The resulting feature maps are then scaled so that each value falls between [-1,1]. The model was evaluated on precision, accuracy, false positive rate, recall and F1-Score. The results show that it is possible to improve the performance of the classification model by enhancing its training with GANs.

Yuan et al. (2020) [53] propose employing GANs to learn the normal conditions of smart meter operations to detect power outages in electrical grid zones. In addition, the authors propose circumventing the problems of other models applied to this problem, which are the assumption that every network node is directly observable. The distribution network is subdivided into zones determined by two neighbouring observable nodes (nodes in which the voltage and demand). Each zone has its designated GAN, which learns the time-series data collected by the two nodes. Any deviation from each node's normal measured data distribution will be considered an anomaly.

The architecture is very similar to the one on vanilla GAN [54]. The generator's objective is to synthesize Time-Windows (frames with recorded sequential events) with events similar to the ones from the assigned zone. Similarly, the discriminator must distinguish between real and fake Time-Windows. At the end of the training phase, both components captured the normality of the data for the given zone.

In the detection phase, both generator and the discriminator are used. Time-Windows with actual recorded events are fed to the latter, which calculates the Discrimination Loss. Additionally, an inverted mapping of the Time-Window to the latent space is given to the generator, which is tasked with reconstructing it. The weighted sum of the reconstruction error and discriminator loss is used to calculate the Anomaly Score.

The solution is evaluated on data collected over three years by smart meters on a complex power distribution network. The results proved that the proposed approach could reliably detect power outages in real distribution networks. Finally, numerical comparisons are performed to compare the developed method with a preexisting SMV model to detect power outages. The authors conclude that the developed GAN can achieve better results (in terms of accuracy) with a significantly reduced amount of data.

AMBi-GAN [55] is bidirectional LSTM GAN for anomaly detection in industrial multidimensional time-series data. Unlike univariate time series, multivariate time series consists of multiple measurements in a given time step. The authors propose solving the difficulties of other methods in extracting temporal information, feature extraction and lack of large amounts of labelled data.

The architecture consists of a discriminator and a generator using the same bidirectional LSTM network (AMBi-LSTM). An attention mechanism is also proposed to learn the importance of each time-series element. It calculates a given sample's weight to determine how much it should affect the parameters in the network.

In the training phase, each sample is extracted using a sliding window that subdivides the whole dataset into equal-length subsequences (each with multiple values for a given time). The generator's goal is to generate windows with the same distribution as the original ones. The discriminator receives real and false samples and must distinguish between them.

When both components have reached an equilibrium, anomaly detection can be performed. Random samples are extracted from the testing dataset and fed directly to the discriminator. At the same time, the samples suffer from inverted mapping into the latent space, so they can be given to the generator, whose goal is to reconstruct them as well as possible. Next, the *Discriminator Score* is calculated by combining the loss from the discriminator and the reconstruction error from the generator.

AMBi-GAN and the other two baselines were evaluated on precision, recall and F1-score on three time-series datasets. The proposed solution outperformed the other baselines on every metric. In addition, several variations of AMBi-GAN were developed, from changing the number of hidden layers of AMBi-LSTM to assessing which one performed better on the chosen datasets.

TAnoGAN [56] was designed to detect anomalies in industrial time-series data with a small number of samples. The model consists of a generator G and a discriminator D, with both architectures based on LSTM networks. In the training phase, G learns to generate realistic time-series sequences from a latent space distribution, and D distinguishes fake samples from real ones. The samples consist of small time series sequences extracted from the dataset with a sliding window method.

In the detection phase, real-time-series samples are mapped to the latent space and then reconstructed by G. Anomaly detection is done by evaluating the reconstruction error of the synthesized sample with the original one. Mapping from the original sample to the latent space is done iteratively. First, a random sample z^i from the latent space z is chosen and fed to G, which generates a fake sample. The resulting fake data $G(z^i)$ is compared to the original sample x with a point-wise dissimilarity measure L_R. Next, the parameters of z^i are updated and thus, in the next iteration, the reconstructed sample will more closely resemble the authentic one. This process is repeated Λ times (a predefined parameter). At the end of the inverse mapping, the final z^i is compared to the original sample, and the anomaly score is obtained with the weighted sum of L_R and the discriminator loss L_D.

To deal with the small number of samples in the datasets, the authors varied the number of hidden layers in each architecture. It was observed that discriminators with many hidden layers easily overfitted the data. In contrast, generators with small numbers of hidden layers failed to synthesize realist time series sequences.

The solution is evaluated on a large number of time-series datasets along with other unsupervised state-of-the-art anomaly detection methods. The performance (measure in accuracy, recall, F1-score, AUC, and Cohen Kappa Score) demonstrates that TAnoGAN is more suited than other models to detect anomalies in small time-series datasets.

MinLGAN [57] aims to detect outliers by generating both normal and abnormal samples during the training phase. The authors employ minimum likelihood regularisation to the generator, G, to produce more abnormal samples and prevent them from converging to the normal distribution of the data. This solution ensures that the performance of the discriminator, D, does not deteriorate in the final phases of training as it receives samples from the generator that are increasingly closer to the real distribution of data. The regularization is done by adapting the loss function of the generator with a KL divergence measure. It penalizes the generator when the produced samples are distributed close to the dataset and, thus, prevents the convergence of G with the original data distribution.

To deal with the instability of the discriminator in the early phases of training resulting from the data's randomness, the authors propose *Ensemble learning*. Two ensemble methods are presented, which are bagging and boosting. The latter consists of a set of models trained from random subsamples of the training dataset. In contrast, the models are obtained in the former by emphasizing training samples that other models misclassified. The authors independently trained a set of D models in line with this. The outputs of each discriminator (before sigmoid activation) are combined into a single value in two different score functions for anomaly detection (one is scaled to account for the minimum and maximum ranges of the outputs).

In the experimentation phase, three GANs were created. One baseline MinLGAN and two other models with ensemble learning and the score functions that were defined. All of them were trained along with five other unsupervised anomaly detection approaches on an image and several tabular datasets. Despite providing good results in all datasets, the authors pointed to the difficulty felt in the training phase due to the complexity of the proposed approach.

ATTAIN [58] is an architecture that makes use of GANs to detect anomalies in cyber-physical systems (CPS). Unlike other methods, ATTAIN can learn data distribution at runtime without needing static data. This allows it to adapt to previously unseen novel attacks continuously.

The solution consists of a Digital Twin Model, a digital replica of a real system, and a Digital Twin Capability; the GAN used for anomaly detection. The first model is built with historical and real-time measurements from sensors and actuators, while the detector only learns from real-time data. The generator's objective is to produce samples from the latent distribution into the original data distribution. The function of the discriminator is to distinguish between normal, and attack samples that come either from real-time or are synthesized by the generator. Therefore, the output of the discriminator consists of four categories.

The generator, *G*, is composed of an input layer which encodes discrete values into one-hot vectors; a Graph Convolutional Layer (GCN) that is tasked with learning the independent relationships between sensors and actuators; a pooling layer which collapses the outputs of the previous layers; and an LSTM layer with the function of retaining the temporal features of the data. The discriminator receives both real and generated samples, which are concatenated and suffer a linear transformation. Next, the resulting vector is passed through a *tahn* activation function before being passed to the next layer. The following step calculates a ground truth label for the received fake sample. It is first given to the Digital Twin Model that predicts its state, and later, the hamming distance d between the predicted and real state is obtained to help in the labelling process. The discriminator determines if the sample is real or fake. In the case of the latter, the d measure from the previous step is compared to determine if the example is a regular or attack adversarial sample. The loss between the ground truth and the likelihood (obtained by softmax of the output of D) is calculated in the final layer.

The designed model was tested with two other baselines on three intrusion detection datasets. The performance metrics were outlier precision, recall and F1-score. A comparison of all the results shows that using the digital twin model to guide the training of the GAN improved the overall outlier detection capability of the model compared to the other solutions.

However, the authors recognize the possible threats to validity as they could not test the solution on a real CPS system.

3.2.5 Analysis

In Table 3.4, a brief analysis of the SLR results will be carried out to study the effectiveness of the chosen methodology. Aside from the paper name and year, four other categories were identified to compare the evaluated solutions:

- **Training Objective:** This category aims to describe the training approach for a given model. With this, the goal is to compare models that fit the normal data distribution during the training phase and other alternatives. This allows for a straightforward summary of the training approaches that can be adopted to solve the proposed problem.
- **Anomaly Detection:** In this group, the goal is to identify several of the approaches that can be used to detect anomalies. These methods can be relative to calculating an anomaly score or, in some cases, using classifiers to achieve this goal. The possible values for this category include Reconstruction errors, Discriminator Loss (D loss), and Classifiers.
- **Architecture:** This class aims to analyze the different types of architectures defined in each of the papers. These can be "Normal" in the case they use the vanilla architecture of GANs; "Mixed" when other components like encoders, E, classifiers, C, feature extractors (FE) and Self-attention devices (SA)[6] are used; and autoencoders (AE).
- **Application:** This last category describes the specific problem that each of the designed models tries to solve (i.e. the scenario to which they are applied). Some models have no direct application and can be used in many types of problems, and because of this, they have no assigned application (N/A)

Figure 3.9 shows the number of analysed papers per year, and Figure 3.10 shows the year distribution of the papers analysed by hand after the application of the citation filter (Figure 3.4). The majority of the papers that were chosen were from 2020. Even after the restrictions applied to the search query, a significant portion of the results is still comprised of GANs for image synthesis.

Further analysis can be done with regard to the search questions defined in Section 3.2.1. For **G_SQ1**, it was shown that several solutions apply GANs or variations in anomaly detection in tabular datasets. Surprisingly, some models worked on image and tabular datasets with the proper adjustments. Additionally, 40% use either an autoencoder or parts of it to help with anomaly detection in complex data types and 9/20 use some variation of reconstruction errors to determine if a sample is anomalous.

As for **G_SQ2**, it was shown that most methods (75%) learn the normal distribution during the training phase and then detect anomalies. 20%, generate anomalies (usually outliers) that are

[6]Module used during the training phase to make the discriminator and generator learn the data features more thoroughly.

Paper	**Year**	**Training Objective**	**Anomaly Detection**	**Architecture**	**Application**
MAD-GAN [36]	2019	Normal	Reconstruction + D loss	Normal	Time Series
ALAD [38]	2018	Normal	Reconstruction	Mixed (E)	N/A
USAD [39]	2020	Normal	Reconstruction	AE	N/A
MO-GAAL [40]	2020	Outlier Generation Division Boundary	Classifier	Mixed (C)	N/A
IGAN-IDS [41]	2020	Outlier Oversampling	Classifier	Mixed (FE + C)	Intrusion Detection
Jiang et al. [42]	2019	Normal	Reconstruction	AE	Time Series
TadGAN [44]	2020	Normal	Reconstruction + D loss	Mixed (E)	Time Series
adVAE [45]	2020	Normal	Reconstruction	AE	N/A
Blance et al. [46]	2019	Outlier Oversampling	Classifier	AE	High Energy Physics
FGPAA [47]	2020	Normal	Reconstruction	Mixed (AE)	Roll Bearing Fault
FGAN [48]	2019	Normal Division Boundary	D loss (adapted)	Normal	N/A
MENSA [49]	2021	Normal	D (initial layers)	AE	Intrusion Detection
Liu et al. [50]	2022	Normal	Classifier	Mixed (C + SA)	Roll Bearing Fault
DOPING [51]	2018	Minority Oversampling	Classifier/Model	AE	Performance Improvement
Bot-GAN [52]	2018	Normal	Classifier (D)	Normal	Bot Detection
Yuan et al. [53]	2020	Normal	Reconstruction	Normal	Power Outage Detection
AMBi-GAN [55]	2021	Normal	Reconstruction + D loss	Normal	Industrial Time Series
TAnoGAN [56]	2020	Normal	Reconstruction	Normal	Time Series
MinLGAN [57]	2018	All Data	D loss	Normal	N/A
ATTAIN [58]	2021	Normal	D Loss	Normal	Cyber-physical Systems

Table 3.4: List of reviewed papers for Adversarial Anomaly Detection.

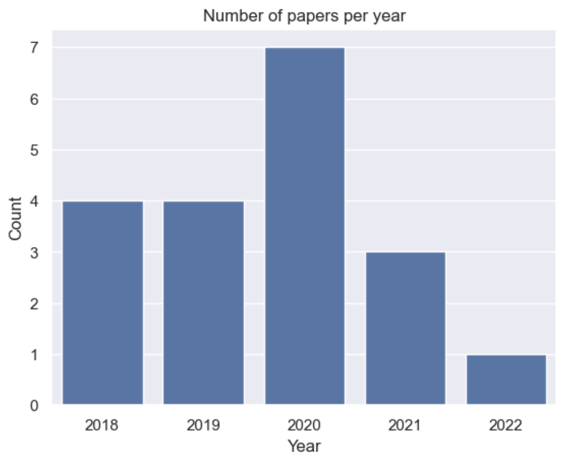


Figure 3.9: Number of reviewed papers per year.

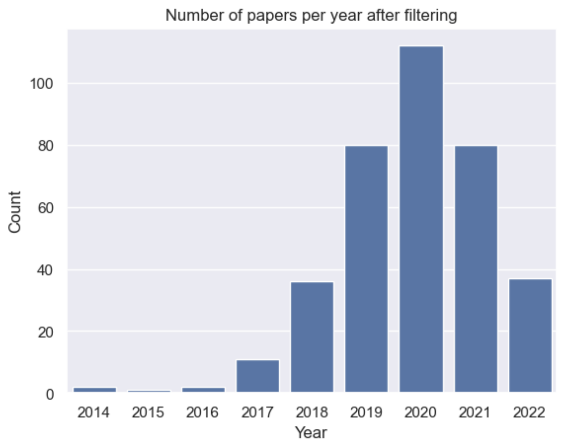


Figure 3.10: Number of reviewed papers after citation filter step (Figure 3.4)

used to train classifiers for detecting anomalies. Only [57] makes use of both abnormal and normal data distribution in the training phase. This indicates that most adversarial solutions for anomaly detection can generate realistic data with the same distribution as the original data. However, not all use the discriminator to distinguish between normal and abnormal samples, as they employ classifiers. In addition, some architectures (50%) use a reconstruction-based anomaly detection mechanism that employs the generator (often with the discriminator, but sometimes alone).

3.2.6 Threats to SLR Validity

The chosen limit for the minimum citations might have been too restrictive, especially for the papers recently published. To address this concern, an adaptation was made to the inclusion criteria I4 (Table 3.3) to allow for fewer citations in earlier papers, aiming to mitigate this risk. Despite this adaptation, the number of analyzed papers for 2022 was notably smaller than in previous years. This raises the concern that some of the newer and relevant papers published during that period might have been inadvertently overlooked.

Furthermore, as only one search engine was used for the SLR, there is also the risk that some relevant papers from other platforms were not encountered during this process.

Another threat may arise due to the terms used to search for relevant papers. After analyzing multiple articles, various synonyms for both GANs and anomalies were collected to reach the most number of papers possible. Despite this, there exists the possibility that some authors didn't use any of these terms to characterize their approach in the title, abstract or keywords of the document. This way, there is a small risk that some relevant papers might have eluded the search query.

Similarly, by prohibiting documents that referenced images, or anything other than tabular data, some relevant papers might have been excluded. This can occur, for instance, in architectures that were firstly designed for image anomaly detection but were also suited for tabular datasets.

3.3 Summary

This chapter explained the methodology used to aggregate the papers relevant to the problem in question. For each part, a set of search questions were defined (Section 3.1.1 and 3.2.1) followed by the respective queries (Sections 3.2.2 and 3.1.2). The selection criteria and the processing pipeline were defined in Sections 3.1.3 and 3.2.3. Summaries of each of the selected papers for both parts are provided in Sections 3.2.4 and 3.1.4. Next, a categorization and summary of the processes are performed in Sections 3.1.5 and 3.2.5, followed by the threats to the SLR process (Section 3.1.6 and 3.2.6).

In the clustering part, 20 papers were selected for final evaluation after the aggregation of results from both Scopus and Google Scholar. Several novel clustering algorithms and methods were identified. Furthermore, most articles talked about applying clustering methods to improve machine-learning tasks in some way, which was the main goal of this part.

For the second part, the same number of papers were analyzed; however, these were retrieved from only one platform. As the search was more restrictive, the number of papers was significantly smaller than in the previous part. Despite this, several novel GAN-based architectures were identified, and the majority of the papers were published in 2020. This is also due to the novelty of this area, as many solutions were published in the last few years.

Chapter 4

Research Statement

This chapter will provide an overview of the problem this thesis is trying to solve. First, a brief description of MULTI-VP's workflow is provided (Section 4.1), followed by existing methods for initial flow estimation (Section 4.2). An extensive analysis of the data used in the previous and subsequent approaches is done in Section 4.3. This thesis's hypothesis and research questions are proposed in Section 4.4.

4.1 MULTI-VP

MULTI-VP [2] is a global MHD model that simulates the three-dimensional structures of the solar wind. In addition, it also estimates the conditions at the Sun's chromosphere, transition region, corona, and low heliosphere. The model computes many one-dimension solar wind solutions from full flux-tube geometries and heating functions. Background magnetic field geometries are extrapolated from publicly available magnetogram data. The method can estimate solar wind profiles across the Sun's atmosphere up to 30 solar radii. The results directly link the geometry of magnetic flux tubes in the lower corona with the distributions of fast and slow solar wind flows. MULTI-VP proved faster than other MHD models and did not suffer from cross-field diffusion effects. For a more in-depth data analysis, refer to the next chapter.

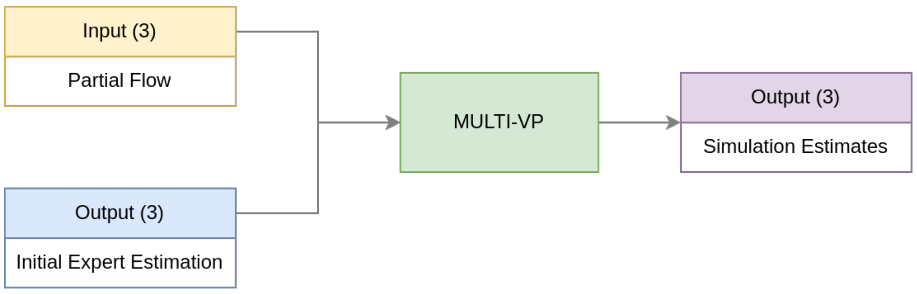


Figure 4.1: MULTI-VP methodology dataflow. The model takes the partial flow and its associated expert initial guess as input and then derives a better solution.

A MULTI-VP simulation workflow overview can be seen in Figure 4.1. It takes as input flux-tube partial flows and initial expert estimations for the solar wind conditions. After a long simulation time, these last estimations are approximated to final estimates more congruent with the actual conditions on the Sun's surface.

4.2 ML for Initial Flow Estimation

Due to its complexity and many calculations, MULTI-VP, like other MHD simulations, still takes a long time to reach viable solar wind predictions. Furthermore, the need for initial expert guesses also significantly delays the process. These factors directly affect the prediction capability and preparation for extreme solar events. Recently in "Initial Condition Estimation in Flux Tube Simulations using Machine Learning" [4], it has been proved that machine learning techniques can accurately produce good initial flow conditions that MULTI-VP can later use. The authors also proved that the quality of the flow estimations is directly linked to the total execution time of the simulation. These allow for faster convergence of the MHD simulation as the initial estimates are closer to the final solution.

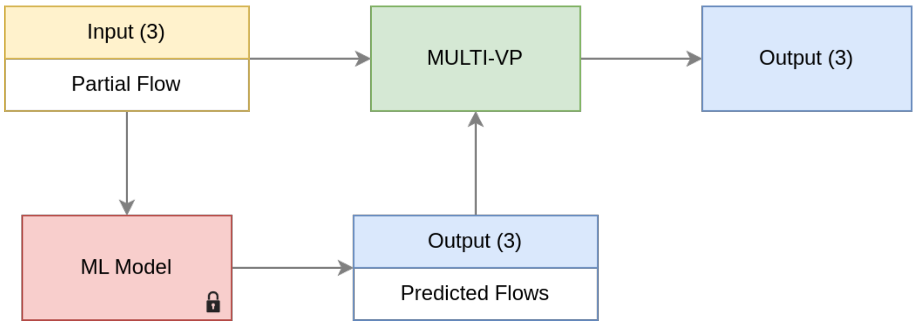


Figure 4.2: ML methodology dataflow. An ML model estimates the initial conditions of a given partial flow. These are then passed to MULTI-VP, approximating them into a final solution.

The approach, illustrated in Figure 4.2, uses an ML model to predict the initial expert estimates from the initial partial flow input. Analogous to the method presented in Section 4.1, MULTI-VP takes as input the partial flows along with their initial conditions predicted by the ML model.

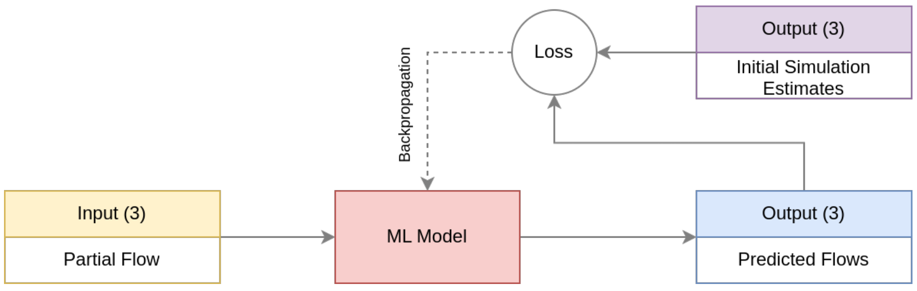


Figure 4.3: Model training method. Takes as input initial partial flows and tries to predict flow estimates close to the ones from previous MULTI-VP simulations.

An illustration of the training method can be seen in Figure 4.3. In this phase, the model takes as inputs initial partial flows and tries to predict the initial conditions. Next, the predictions for a given flow are compared to those from previous MULTI-VP runs to calculate the prediction's loss and update the model's parameters with backpropagation. The logic behind this was that the model would learn to predict initial estimations closer to the final solution, and thus, the simulation would converge faster.

Due to the high simulation time of MULTI-VP, a small number of files were randomly selected from the whole dataset to be used as a validation set. These were excluded from the training and testing phases. The performance of the ML method was evaluated by feeding MULTI-VP with the predictions of the validation dataset and then assessing the error of the simulation estimates and the overall computation time.

However, the reduction in execution time was minimal. The authors posit that the presence of anomalies during the training phase might have hampered the overall prediction quality of the model. Another potential issue arises from the nature of the data, which exhibits significant dispersion, with high concentrations in certain areas and low concentrations in others. This characteristic can pose challenges as the neural network model tends to converge towards the densely populated points, potentially failing to learn the patterns in the peripheral areas.

4.3 Exploratory Data Analysis

The data used by MULTI-VP and the prediction model consists of magnetogram data from the Wilcox Solar Observatory. Each file contains 12 columns representing measurements of the magnetic field in the solar atmosphere at different heights. Every variable comprises 640 points (abscissas) measured at different radial distances from the Sun (up to 30 Solar radii). In addition, the data is distributed evenly throughout five batches, each consisting of solar wind measurements at different surface locations. For this work, only six columns will be used, as the others are derivations of these and, thus, are redundant.

Input (Partial Flows)			Output (Estimations)		
$R[R_{sun}]$	$B[G]$	$\alpha[deg]$	$n[cm^{-3}]$	$v[km/s]$	$T[MK]$

Table 4.1: Data columns of magnetogram used by MULTI-VP.

The data columns can be seen in Table 4.1. These are divided into two parts: the input and the output (predictions). The former comprises the set of variables the simulation uses to approximate solar wind conditions, and the latter the initial expert guesses needed to kickstart the multiple flux simulation. Note that, like in Barros [4] (Figure 4.3), the output variables are the predictions of previous MULTI-VP simulations and not actual expert predictions.

The input data includes the magnetic field amplitude, B, the flux tube inclination, α, and the radial coordinate, R. The output data consists of the number of charged particles per unit volume, n, the velocity, v, and the temperature, T.

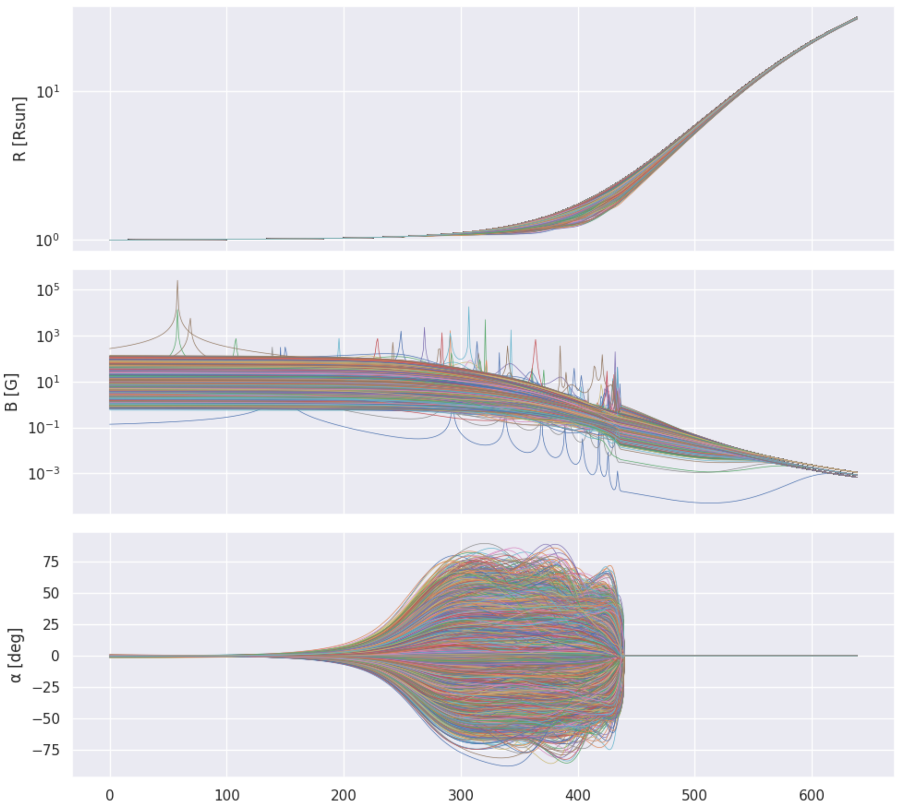


Figure 4.4: Joint plot of the input variables of each file used in this work. The first row is the plot of the radial coordinate radius, R, the second the magnetic field, B, and the last the flux tube inclination α. All are plotted in function of position in the magnetogram file.

Figure 4.4 shows a joint plot of the input variables. Based on these plots, it can be inferred that several input files contain anomalous data. This is evident from the graph of variable B, where specific files exhibit significant deviations from the overall distribution. Furthermore, there are instances of faulty lines within the normal distribution, which further complicates the detection of these anomalies.

As previously explained (in Section 4.1), MULTI-VP requires initial expert guesses to kick-start the simulation. These consist of the output variables presented in Table 4.1 and, during the simulation, are approximated to better solutions. In line with the work carried out in Barros [4], we will be using the outputs of previous simulations as initial guesses (refer to Figure 4.3) for more details). Figure 4.5 shows the joint plot of the output variables. Similar to the input plots, several faulty predictions can be seen in all variable plots, resulting from simulations carried out on anomalous inputs.

A preliminary statistical analysis of the data can be seen in tables 4.2. The mean, standard deviation, minimum, maximum, and quartiles of each variable are presented in it. Out of the three input columns used, the magnetic field (B) has the highest standard deviation, which might indicate that this physical quantity is more prone to anomalies than the others. The radial coordinate (R) has the lowest standard deviation, which is expected, as it is almost a constant value for each file.

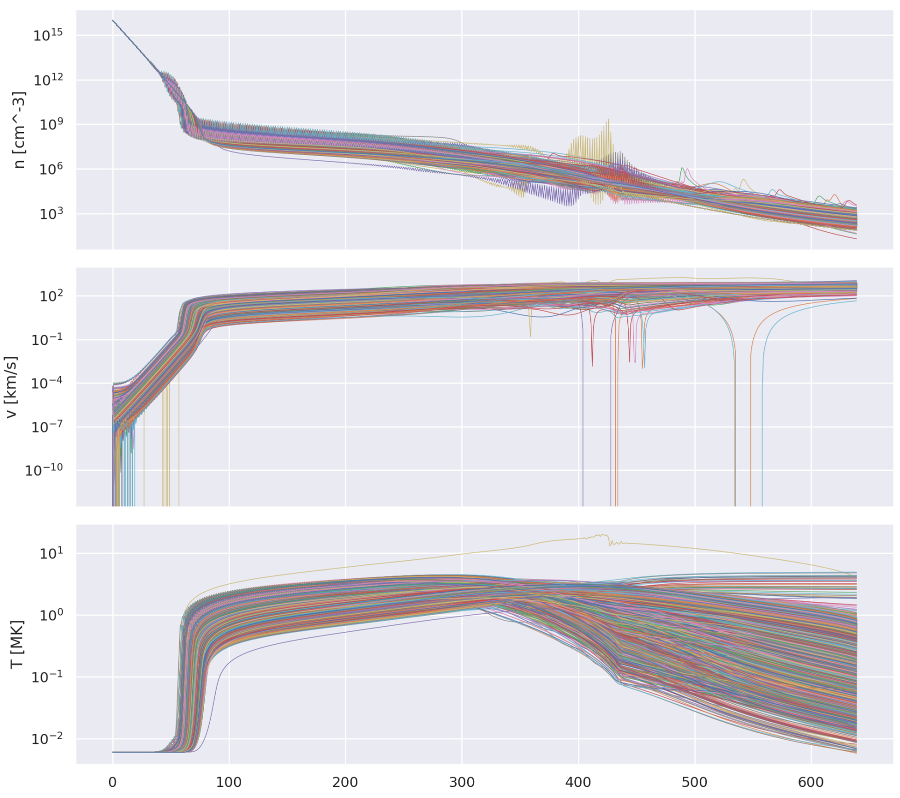


Figure 4.5: Joint plot of the output variables of each file used in this work. The first row plots the number of charged particles per unit volume, n, the second the velocity, v, and the last the temperature, T. All are plotted in function of position in the magnetogram file.

	R [Rsun]	B [G]	α [deg]	$n[cm^{-3}]$	$v[km/s]$	$T[MK]$
mean	4.755	5.471	1.885	8.630e+13	2.553e+02	1.384
std	7.165	9.178e+01	1.472e+01	6.839e+14	2.148e+02	8.968e-01
min	1.000	5.122e-05	-8.763e+01	1.973e+01	-6.757e-03	5.765e-03
25%	1.021	4.051e-02	-1.079e-01	1.622e+04	4.926e+01	7.179e-01
50%	1.151	2.095	0.000	2.351e+06	2.110e+02	1.337
75%	4.250	5.582	9.997e-01	2.132e+07	4.508e+02	2.098
max	3.150e+01	2.470e+05	8.925e+01	1.010e+16	1.889e+03	1.990e+01

Table 4.2: Statistical Analysis of the dataset.

The output variables have a similar standard deviation, with the number of charged particles per unit volume (n) having the highest and the temperature (T) the lowest. The velocity (v) has a standard deviation similar to T.

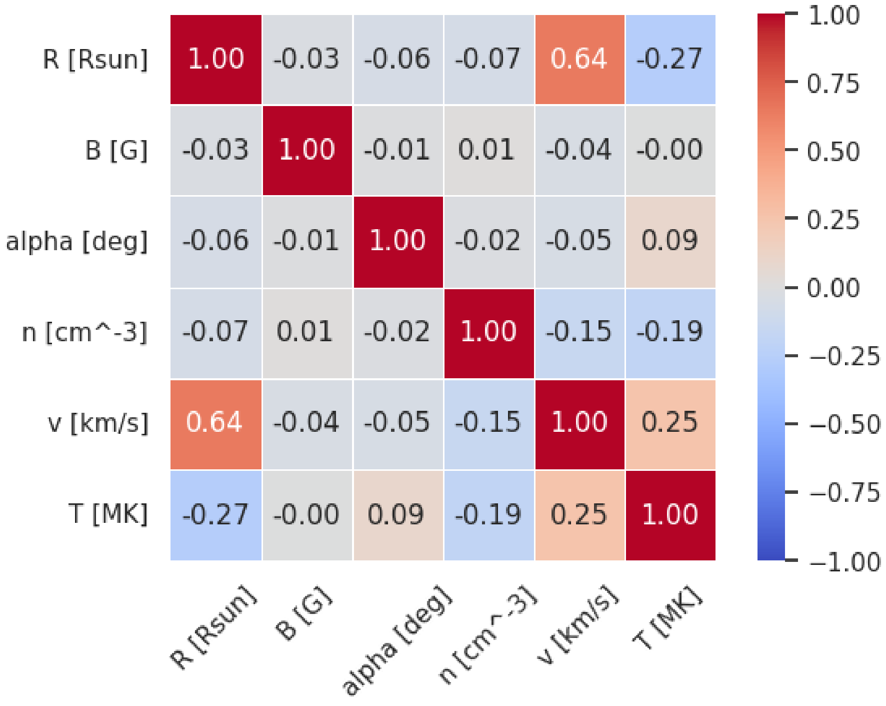


Figure 4.6: Correlation plot of all variables used in this work.

In addition to these statistics, the correlation between the variables can be seen in Figure 4.6. There is virtually no correlation between the input variables (R, B, and α), which is expected, as they are independent observations.

The output variables (n, v, and T) are somewhat correlated. The temperature, T, is positively

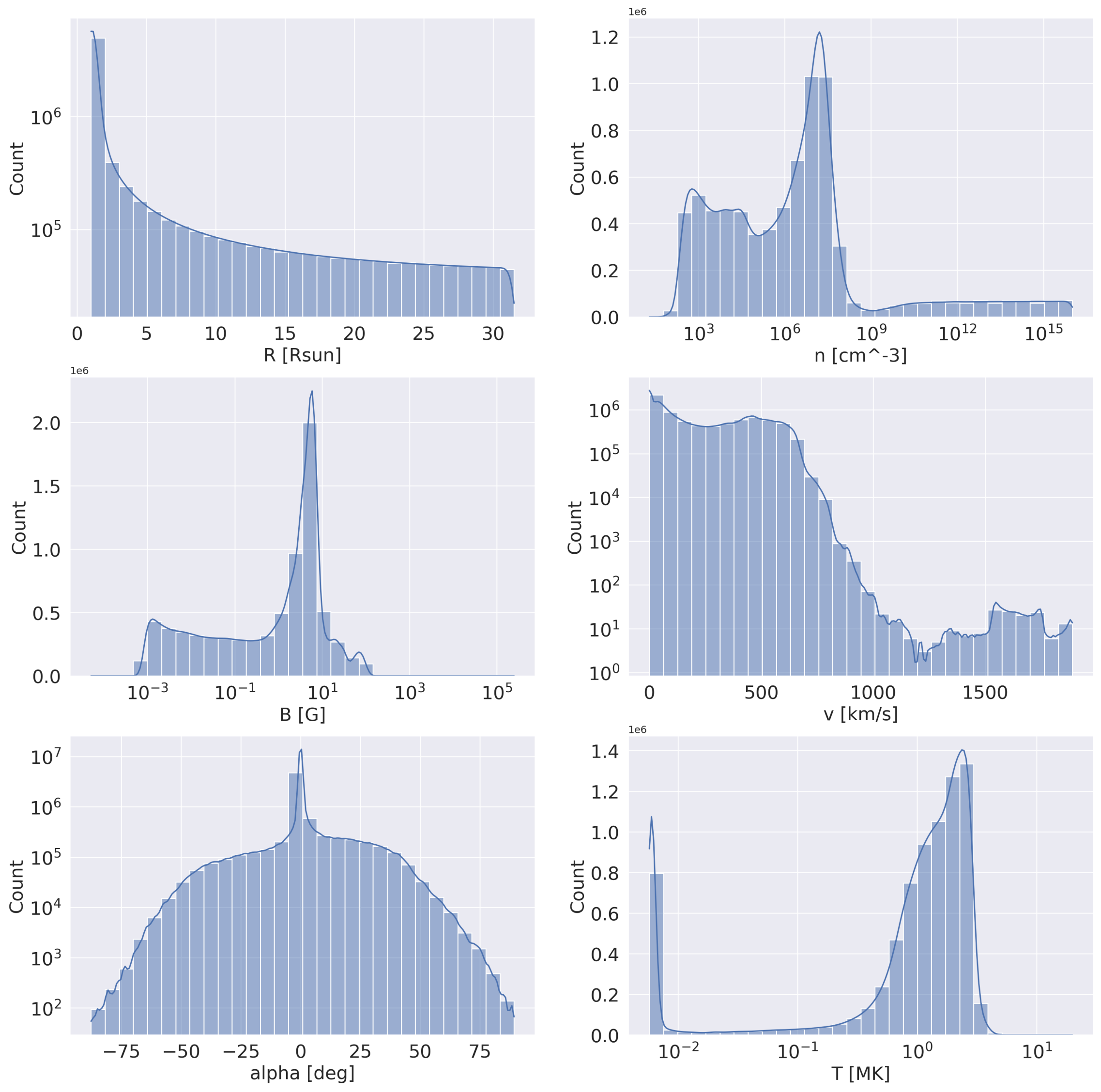


Figure 4.7: Plot of the value distributions of the input (left) and output (right) variables.

correlated with the velocity of the solar wind, v, which is expected as with higher wind speeds, the particles there is a tendency for higher temperatures. The opposite is true for the volume density of the solar wind, n, which is negatively correlated with the temperature. This is expected as with higher densities, the particles will have less kinetic energy and, therefore, a lower temperature. The velocity of the solar wind is also slightly negatively correlated with the volume density.

A high correlation between the radial coordinate R and the velocity v can be observed, as v is expected to increase with distance from the Sun. Additionally, the temperature of the solar wind drops as the distance to the Sun increases, which is reflected in the negative correlation between R and T.

A distribution plot of each input variable can be seen in the left column of Figure 4.7. As expected, the values of R range from 1 to about 31.5. This is corroborated by Table 4.2, with most values concentrated around 1. $B[G]$ displays a skewed distribution. At the same time, $\alpha[deg]$ is more evenly distributed (almost symmetric in relation to $x = 0$).

Contrary to the findings in the input variables, the values of the output variables are much less evenly distributed. The number of charged particles per unit volume, n, has a very skewed distribution, with most values being concentrated in the 10^3 to 10^7 range. The velocity, v, has a larger distribution of values, with most indicating slower velocities. The temperature, T, has a strange distribution, with most values concentrated near 0 and the rest ranging from 0.5 to 5.

4.4 Hypothesis

In an attempt to solve the issues of the previous approaches, the following hypothesis can be formulated:

> By integrating clustering and adversarial anomaly detection techniques, the initial conditions predicted by RNNs for the MULTI-VP simulator will be closer to the final simulation results and contribute to faster executions.

The following questions offer a disambiguation of the proposed hypothesis:

Q1 *What do estimates closer to the final simulation mean?* As explained in Section 4.1, MULTI-VP takes initial flow conditions provided by experts as input and slowly converges to a final and viable solution. With these terms, we intend to verify if the estimates predicted by the RNN models will be nearer to the simulation outputs than the original expert guesses.

Q2 *What does "faster execution" mean in this context?* These terms are mainly used to describe the computation time that MULTI-VP takes to reach a viable solution. A faster simulation would mean a decrease in the time the simulation is busy trying to reach a solution.

Taking these factors into consideration, the research questions of this thesis are:

RQ1 *Are clustering methods capable of detecting characteristics in the dataset that were overlooked by the original RNN and would help with the prediction task?* This question aims to assess the potential improvement in estimating solar wind by incorporating clustering techniques during the training of the prediction model. Specifically, it investigates whether the clustering methods can identify and capture dataset characteristics previously overlooked by earlier iterations of the RNN model. The objective is to determine if the integration of clustering techniques can ultimately result in better estimates of solar wind.

RQ2 *Do the estimates obtained with clustering-based training significantly improve the simulation's performance?* This question aims to validate whether using clustering in the method significantly reduces the computation time of MULTI-VP for solar wind estimates. In addition, we are trying to assert if the final estimates from MULTI-VP are closer to the desired outcome.

RQ3 *Can adversarial learning methods detect anomalies in solar wind profiles?* Considering anomaly samples as the positive class, we mostly try to reduce the False Negative Rate (FN) since it has the most detrimental impact on the training of the predictive model. As a secondary priority, we will focus on reducing the amount of False Positives (FP) to ensure that almost no relevant samples are excluded from the training process.

RQ4 *Does the resulting dataset significantly improve the predictive ability of the RNN?* If the resulting dataset following the removal of anomalies leads to an improvement in the model used to predict initial conditions from input flows. In other words, does the mean square distance between the actual estimations and those predicted by the model decrease compared to the previous method?

RQ5 *Does the improved predictive ability of the RNN result in a further reduction of execution time for MULTI-VP?* This question aims to clarify if the developed model for initial flow estimation can produce improved approximations of solar wind flows and thus reduce the time it takes for MULTI-VP to reach a solution.

4.5 Methodology

To tackle the abovementioned questions, we will employ clustering techniques on the magnetogram data from various sources (e.g., Wilcox Solar Observatory) to improve the performance of the initial flow estimation model. With initial conditions closer to the final ones, we expect that MULTI-VP will reach better solutions in a smaller time frame. The research approach for this phase will be performed quantitatively, with the estimates from this new method being compared to the ones obtained in Barros [4].

In addition to this, we will also employ adversarial learning techniques for anomaly detection in solar wind profiles. This approach will be used to assert if extreme data values hinder the RNN model's overall performance and, consequently, the MULTI-VP simulation. The research for this phase will take on a qualitative and quantitative approach. Due to the lack of labelled data and consensus on normal and anomalous profiles, the results from anomaly detection will need to be evaluated visually. In other words, the chosen anomaly detection approach will need to be picked based on the visual inspection of the dataset after anomaly detection. Like in the clustering phase, the comparison of this approach and the previous ones will be made in a quantitive way.

Chapter 5

Clustering

In this part of the thesis, we used clustering methods to improve the performance of the previous prediction models. The goal is to find groups of data points that are similar to each other but different from the rest of the data. This is done by finding the distance between each data point and the rest of the data. Most algorithms calculate the distance between two data points using a distance metric.

In this chapter, the clustering methods that were used will be briefly explained in section 5.1. Section 5.2 provides an overview of the clustering validity metrics used in this work. Section 5.4 presents the experiments undertaken to determine the most appropriate clustering method for the dataset. Section 5.5 explains the experiments undertaken with the solar wind prediction model with the most appropriate clustering from the previous task. Finally, Section 5.6 provides an overview of the work in this chapter.

5.1 Clustering Methods

Clustering is defined as finding and then grouping the data values that are close to each other in some way. Due to these reasons, clustering works as a valuable technique not only for data analysis but also for enhancing machine learning performance. Additionally, clustering can be used for anomaly detection, as abnormal samples tend to be far from the rest of the data. In this section, the clustering methods that were used in this work will be presented.

There are several approaches to the clustering task; however, these can be divided into two main categories: Hierarchical and Partitional. In the hierarchical approach, the data is iteratively divided in accordance with its patterns in a bottom or top-down approach. This category is further subdivided into agglomerative and divisive techniques. For the first, clusters start at a single point and are iteratively merged with other points forming increasingly larger clusters. On the other hand, the divisive approach starts with all the data points in a single cluster and then divides them into smaller clusters. These methods tend to originate dendrograms representing the nested grouping of patterns and similarity levels. [59, 60]

The Partitional approach divides the data into a set of non-overlapping clusters. These methods are usually based on optimising a criterion function, such as minimising the sum of squared errors. The most common methods in this category are K-Means and K-Medoids. This category can also be subdivided into distance, density and model-based approaches. The distance approach divides the data into clusters based on the distance between the data points, such as in the KMeans method. Density-based approaches are based on the idea that clusters are dense regions of data points that are separated by regions of lower density. Model-based approaches often apply decision trees or neural networks to learn the patterns in the data and then use these models to cluster the data. [60]

5.1.1 K-Means

K-Means aims to find a partition of the data into K clusters, where each data point belongs to the cluster with the nearest mean. The K-means algorithm is an iterative algorithm randomly assigning each data point to a cluster. Then, the mean of each cluster is calculated, and the data points are reassigned to the cluster with the nearest mean. This process is repeated until the data points stop changing clusters.

The number of divisions is defined by the k parameter, which can be determined with the help of the Elbow Test, a heuristic method used to find the optimal number of clusters. This method plots the sum of squared errors (SSE) for each value of k and then finds the value of k that has the "elbow" in the plot. This value of k is considered to be the optimal number of clusters. This method presents some disadvantages, as it is not always clear where the "elbow" is, making it a subjective method. Also, the SSE is not a good measure of the clustering quality, as it is biased towards clusters of similar sizes.

K-means is advantageous compared to other clustering methods because of its simplicity and the fact that it doesn't require any prior knowledge of the data. However, the disadvantages are that it is biased towards spherical clusters with similar sizes, is sensitive to the initial position of cluster centroids and requires the determination of the number of clusters beforehand.

5.1.2 SOM

This method is based on the idea of self-organizing maps, which are artificial neural networks used to find a low-dimensional representation of a high-dimensional space. It is usually employed to reduce the dimensions of the data to a map but can also be used as a clustering method, as it groups similar data.

Unlike other neural network algorithms, SOM is trained with competitive learning. In this method, neurons compete to be the most similar to the data point. This point can only cause the activation of a single neuron in the network, called the *Best Matching Unit* (BMU). At the end of the training process, each neuron is specialized in a specific region of the input space, presenting a cluster.

The algorithm starts by randomly assigning each data point to a neuron in the map. Next, a random input vector is chosen from the dataset, and the BMU is determined. This is done by measuring the distance between each neuron's input vector and weight vectors and then choosing the neuron with the minimum distance to the input. The BMU and its neighbours are then updated to be more similar to the input vector. This process is repeated until the map converges.

The advantage of this algorithm is that it can map high-dimensional input vectors to a low-dimensional space while preserving the original topology of the data. Unlike other clustering methods, which cannot provide good results for data with high dimensionality.[1]

5.1.3 Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering method that assigns each data point to its cluster. Then, the two closest to each other are merged into a single cluster. This process is repeated until all the data points are in the same cluster. These are combined by comparing intra-cluster and inter-cluster distances. Distance between the clusters is calculated using a linkage function, and the distance between the data points in the clusters is calculated using a distance metric (normally the Euclidean distance). These two parameters directly influence the size and shape of the clusters.

The number of clusters can be determined with the help of a dendrogram plot. This plot shows the distance between the clusters as they are merged. Each leaf in the dendrogram represents a data point fused with a similar point into the same cluster as the height of the tree increases. The height of the fusion (on the vertical axis) represents the dissimilarity score between the two clusters. Higher fusions indicate that the clusters are more distinct. As a rule of thumb, the number of clusters can be determined by looking at the height of the fusion(s) with the most distinct clusters (higher dissimilarity scores).[2]

One of the problems with this method is that there is no clear way to determine the number of clusters, as the dendrogram doesn't always produce clear divisions, making choosing the number of clusters subjective.

5.1.4 DBSCAN

This unsupervised clustering method was first introduced in Ester et al. [61]. It is a density-based clustering algorithm used to find clusters of data points close to each other. The algorithm assigns "core" labels to the points with at least *minPts* points within a distance *eps* from them. Points that are reachable from a core point, but do not have at least *minPts* points within a distance *eps* from them, are assigned "border" labels. These are considered part of the cluster formed by the core point. Points not reachable from any core point are assigned "noise" labels and are considered outliers. Because of this, the algorithm provides a basic method for outlier detection.

The algorithm is advantageous because it doesn't require determining the number of clusters beforehand, as it can find clusters of any size and is also robust to outliers. Despite this, it is

[1]Self-Organizing Maps https://sites.pitt.edu/~is2470pb/Spring05/FinalProjects/Group1a/tutorial/som.html

[2]Agglomerative Clustering https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/

sensitive to the parameters *minPts* and *eps*, with slight changes in these parameters significantly impacting the results.

5.2 Validity Measures

As previously stated, one of the main difficulties in clustering is deciding the right number of clusters to ensure a clear data division. This is a subjective process in which the use of validity measures can aid. These measures, often called indices, are used to evaluate the quality of the clustering and can be used to determine the optimal number of clusters. In this section, the validity measures that were used in this work will be presented.

Cluster validity indices can be divided into three categories: external, internal and relative. External validity indices compare the clustering results with the ground truth (labels). On the other hand, internal validity indices serve to evaluate the quality of the clustering without the use of external information. Relative validity evaluates the clustering structure by varying different parameters of the algorithm, such as the number of clusters.

External validity criteria will not be considered for this study as no ground truth information is available. Instead, internal and relative validity approaches will be used to evaluate the clustering quality. The internal validity indices that were used are the following:

Silhouette Score (S). This score is among the most widely used to evaluate clustering goodness. It is also called the mean Silhouette Coefficient for all clusters and is calculated with the mean intra-cluster distance and the mean nearest-cluster distance. Observations with a score close to 1 are well clustered, while observations close to -1 are likely to be assigned to the wrong cluster. Scores around 0 indicate overlapping clusters.

Calinski-Harabasz Index (CH). This index is also known as the Variance Ratio Criterion. It is calculated by dividing the between-cluster or inter-cluster dispersion by the within-cluster or intra-cluster dispersion. The higher the value of the index, the better the clustering. Like the silhouette score, it evaluates the goodness of the clustering structure, with higher values indicating better clustering; contrary to the previous score, it has no reasonable bound and can take any value. Because of this, it is difficult to use this metric to compare clusterings generated with different methods.

Davies-Bouldin Index (DB). This index is calculated by taking the average similarity measure of each cluster with its most similar cluster. This is done by calculating intra-cluster dispersion for each cluster, i, followed by the separation measure of each cluster with every other cluster, j. Then the similarity of a cluster is obtained by dividing the sum of the intra-cluster dispersions of i and j by the distance between the centroids of the two. Next, the maximum similarity measure is selected for each cluster, and the average of these values is calculated. Unlike the previous measures, the lower the value of the index, the better the clustering, as this indicates less similarity between clusters.

5.3 Dimensionality Reduction

In Data Science and Machine Learning, dimensionality refers to the number of features of the dataset. The dimensionality of a dataset can be reduced by removing features irrelevant to the problem at hand. This can be done using feature selection methods, which select the most relevant features for the task. However, this can also be done by using dimensionality reduction methods, which transform the data into a lower-dimensional space while preserving the most important information.

These processes are useful in these contexts because they can reduce the computational cost of the algorithms, as well as the time needed to train the models. They can also improve the performance of the algorithms, as they can remove noise and irrelevant features from the data.

From the data analysis in Section 4.3, it can be seen that each line, representing a distinct variable, has 640 abscissas which translate to the same number of features. This extremely high number of features can be problematic for some algorithms, as the features are not all equally relevant to the problem at hand. Therefore, it is important to reduce the dimensionality of the dataset so that the algorithms can be trained more efficiently and efficiently.

In this case, variables with the same repeating values will be considered less important and discarded by the dimensionality reduction algorithm, emphasising other variables with more variation. This will be the case for the *R* variable, which is almost constant throughout every line measurement and thus provides almost no meaningful information.

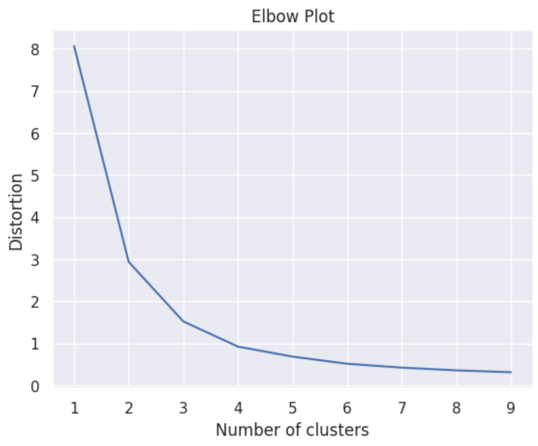
For this work, we will be using the following dimensionality reduction methods:

PCA Principal Component Analysis is a linear dimensionality reduction method that uses Singular Value Decomposition to project the data into a lower-dimensional space. It is advantageous because it is fast and efficient but also sensitive to outliers.

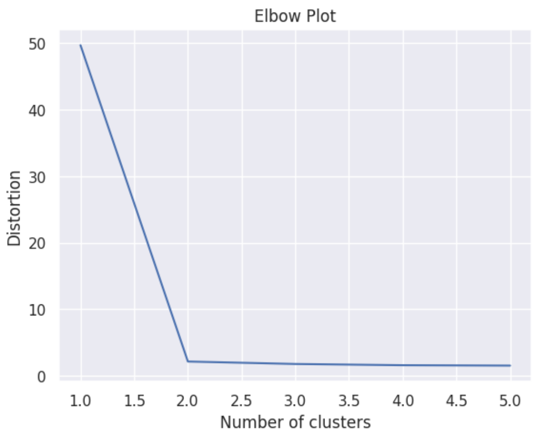
t-SNE t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction method based on the idea that similar data points should be close to each other in the lower-dimensional space. It is advantageous because it can preserve the topology of the data, but it is also computationally expensive.

5.4 Experiments

After identifying the most common clustering and dimensionality reduction methods, the next step is to apply them to the dataset and evaluate their performance. This section will describe the experiments that were carried out to try to better understand the dataset.



(a) Magnetic Field (B).



(b) Flux-tube inclination (α).

Figure 5.1: TimeSeriesKMeans Elbow Tests

The data was scaled in every experiment with the QuantileTransformer module from the *sklearn* [62] library. This module transforms the data to follow a uniform or a normal distribution. This is done to avoid the influence of outliers in the results of the clustering algorithms. This scaling method was also chosen out of consistency, as it will be later used to scale the data for the machine learning algorithms. Additionally, every algorithm and method that depends on random initialization was set to the same seed to ensure reproducibility.

5.4.1 Time Series KMeans

The first method to be tested was the *TimeSeriesKmeans*[3] clustering algorithm from *tslearn* [63]. As the name indicates, this algorithm is a variation of the K-means algorithm used for time series data. It is advantageous as it can cluster time series data without transforming it into a lower dimension, which can be problematic because it can lead to the loss of information. This approach was tested due to these reasons, as the data used in this thesis has some similarities with time series data, which are a large number of features and somewhat correlated consecutive observations. Despite this, the method is still based on the K-Means algorithm presented in Section 5.1.1 and presents the same disadvantages as the original algorithm.

Clustering was conducted first on the magnetic field (B) and then on the flux tube inclination (α) variable. For each one, an elbow test was conducted to try and determine the most appropriate number of clusters.

Figure 5.1a shows the elbow test for the magnetic field variable. From this plot, it can be seen that the elbow is at $k = 4$, which indicates that the optimal number of clusters is 4. However, this is not a clear elbow, as the curve is not smooth, and the elbow is not very pronounced. This indicates that the optimal number of clusters is unclear, and the results might not be very good.

[3]TimeSeriesKMeans `https://tslearn.readthedocs.io/en/stable/gen_modules/clustering/tslearn.clustering.TimeSeriesKMeans.html`

K	S score	DB	CH
2	0.524	0.669	20376
3	0.511	0.628	25585
4	0.498	0.606	30677
5	0.458	0.648	31995
6	0.449	0.671	34394
7	0.431	0.695	35462
8	0.422	0.712	36070
9	0.403	0.738	36368

(a) Magnetic Field (B)

K	S score	DB	CH
2	0.865	0.190	263222
3	0.499	1.078	160481
4	0.493	1.089	120114
5	0.463	1.123	93537
6	0.480	1.104	82431
7	0.259	1.312	82038
8	0.257	1.306	76633
9	0.252	1.374	70313

(b) Flux-tube inclination (α)

Table 5.1: Validity metrics for different TimeSeriesKMeans models obtained by varying the number of clusters.

The validity metrics discussed in Section 5.2 were calculated for different KMeans models obtained by varying the number of clusters to get a more precise number of clusters. The results of these tests can be seen in Table 5.1a. From the first three entries, it can be seen that the highest silhouette score is obtained with $K = 2$, but the lowest Davies-Bouldin index is obtained with $k = 4$, which also has the highest Calinski-Harabasz index of the three. This can indicate that the most appropriate number of clusters for this variable is 4.

Following the same procedure as for the magnetic field, an elbow test was conducted on the flux tube inclination variable, α. The results of this test can be seen in Figure 5.1b. In contrast with the results of the previous test, this elbow is much clearer, with the elbow being at $k = 2$. This indicates that the optimal number of clusters is 2.

The validity metrics (Table 5.1b) also provide a clear indication that the correct number of clusters for this variable is 2, with the highest Silhouette and Calinski-Harabasz scores being obtained with $k = 2$. The Davies-Bouldin index is also the lowest for this value.

5.4.2 SOM

Following the experiments with TimeSeriesKmeans, the SOM algorithm was tested, which is also seen as a useful method for clustering high-dimension data. A Python implementation of the algorithm [64] was tested on the same variables as the previous algorithm, B and α. Unlike KMeans, this algorithm has no tests to determine the number of clusters visually. Because of this, the number of clusters was determined by trial and error by varying the number of clusters and evaluating the results of the validity metrics (Table 5.2). The first two columns of each subtable indicate the x and y dimensions of the SOM map used for the clustering task. The number of clusters is obtained by multiplying these two values.

x	y	S score	DB	CH
2	2	0.500	0.602	30962
2	3	0.281	5.228	3429
3	2	0.454	0.660	35001
3	3	0.407	0.727	37133

(a) Magnetic Field (B)

x	y	S score	DB	CH
2	2	0.269	1.479	122195
2	3	0.004	1.325	5254
3	2	0.269	1.326	89759
3	3	0.231	1.426	66878

(b) Flux-tube inclination (α)

Table 5.2: Validity metrics for different SOM models obtained by varying sizes of the maps (x and y variables).

From looking at the results of the validity metrics for the magnetic field variable (Table 5.2a), it can be concluded that the highest Silhouette score and DB index are obtained with $x = 2$ and $y = 2$, which translate to 4 clusters. The algorithm failed to generate a proper separation for a map size of $x = 2$ and $y = 3$. The Calinski-Harabasz index is the highest for $x = 3$ and $y = 3$, which translates to 9 clusters. However, this does not indicate the correct number of clusters, as the DB index is higher than the first clustering.

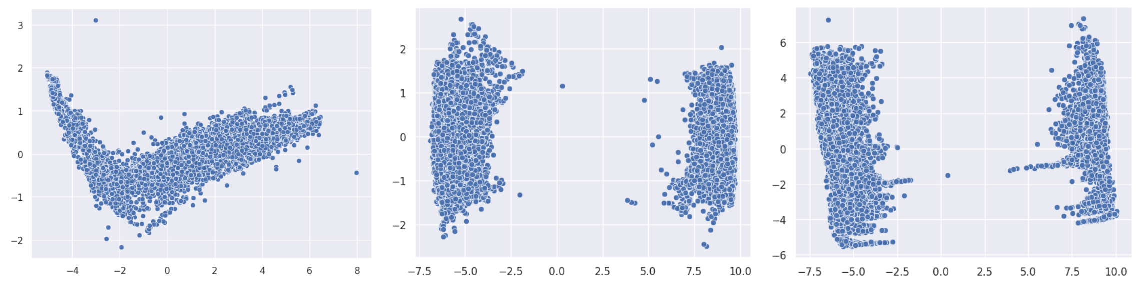
The results of the validity metrics for the flux tube inclination variable (Table 5.2b) are even less clear than the previous ones. The highest Silhouette score is obtained with the first and third models. The best CH index was by far the first one. The DB index is very high for every generated model, which indicates that clustering in this variable may not be a good idea.

Overall the results from this experiment indicate that the SOM algorithm is not a good choice for clustering this dataset, as it cannot generate a clear separation between the clusters. This is especially true for the flux tube inclination variable, where the algorithm failed to generate a proper separation. The only variable where the algorithm could generate a somewhat clear separation was the magnetic field with a map of 2x2 neurons.

5.4.3 PCA Clustering Approach

The next approach that was tested was to apply PCA to the dataset and then apply the clustering algorithms to the reduced dataset. This approach was tested on the magnetic field variable, $B[G]$, the flux tube inclination, $\alpha[deg]$, and a combination of all the input variables.

Tests were conducted to try and find the optimal number of components that would explain the dataset. This was done by analyzing the cumulative explained variance of PCA models with different *n_components*. For B and α, about 99 % of the variance was explained by just two components. As for the combined dataset, 98% was explained by also two components. With this, it can be concluded that it is valid to reduce the dimensionality of this dataset to just two components. This is useful because it provides a simplified data representation while preserving most information.



(a) Magnetic Field (B). (b) Flux-tube inclination (α). (c) Joint Inputs (R, B and α).

Figure 5.2: PCA applied to the different variables. (a) and (b) represent the PCAs of the magnetic field variable ($B[G]$) and the flux-tube inclination variable ($\alpha[deg]$), respectively; (c) is the PCA of all input variables combined.

The representations for each approach can be seen in Figure 5.2a. Note that the representation generated for the flux-tube indication and the joint inputs are very similar. This indicates that the flux-tube inclination variable is the one that contributes the most to the PCA.

For the clustering part, the KMeans and the AgglomerativeClustering methods of the *sklearn* library were applied to each representation to determine the correct number of clusters with the same methodology as in the previous sections.

a) PCA of the Magnetic Field

The KMeans and the AgglomerativeClustering methods were applied to the PCA of the magnetic field variable. The results of the elbow tests for KMeans can be seen in Figure 5.3. This plot shows that the elbow is at $k = 4$, indicating that the optimal number of clusters might be 4.

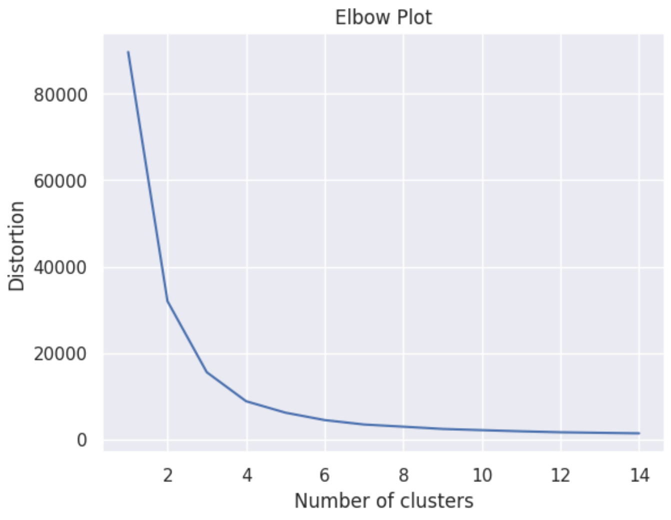


Figure 5.3: KMeans Elbow test for the PCA of the magnetic field variable.

The validity measures from the previous sections were calculated for each method to get a more concrete outlook. The results of these tests can be seen in Table 5.3. For the KMeans

	KMeans			**Agglomerative**		
K	**S score**	**DB**	**CH**	**S score**	**DB**	**CH**
2	0.538	0.645	21224	0.507	0.669	17899
3	0.534	0.586	28086	0.496	0.622	23868
4	0.531	0.549	35942	0.502	0.550	30612
5	0.502	0.579	39762	0.484	0.593	37737
6	0.494	0.592	44986	0.477	0.585	40121
7	0.486	0.600	49233	0.460	0.598	44158
8	0.454	0.637	49757	0.436	0.638	45487
9	0.457	0.636	53265	0.427	0.630	47528

Table 5.3: Validity metrics obtained by different clustering methods on the PCA of the magnetic field variable. Various models were created for each method by varying the number of clusters, K.

algorithm, the highest Silhouette score is obtained with $K = 2$. Still, the lowest Davies-Bouldin index is obtained with $K = 4$, which also has the highest Calinski-Harabasz index of the first three results. This further confirms the results of the elbow test.

The results of the Agglomerative method are not as clear as the previous ones. The highest silhouette score is also $K = 2$, but as in KMeans, the lowest DB index was $K = 4$. In addition, the CH score is much higher for the model with 4 clusters than for the $K = 2$ model.

With the consensus of both clustering methods, it can be concluded that the optimal number of clusters for the PCA of the magnetic field variable is 4. Interestingly, this is the same number of clusters that were obtained with the TimeSeriesKMeans algorithm in Section 5.4.1 and the SOM algorithm in Section 5.4.2 for the magnetic field.

b) PCA of the Flux-tube Inclination

Following the same procedure as in the previous approach, an elbow test was conducted for the KMeans model of the PCA of the flux-tube inclination variable. The results of this test can be seen in Figure 5.4. The plot indicates that the most appropriate number of clusters is 2.

These results are further corroborated by the validity metrics in Table 5.4. The highest Silhouette score and DB index are obtained with $K = 2$, with the highest Calinski-Harabasz index of all the results. This indicates that the optimal number of clusters is 2 for both tested methods. This division occurs because the α variable may take negative or positive values.

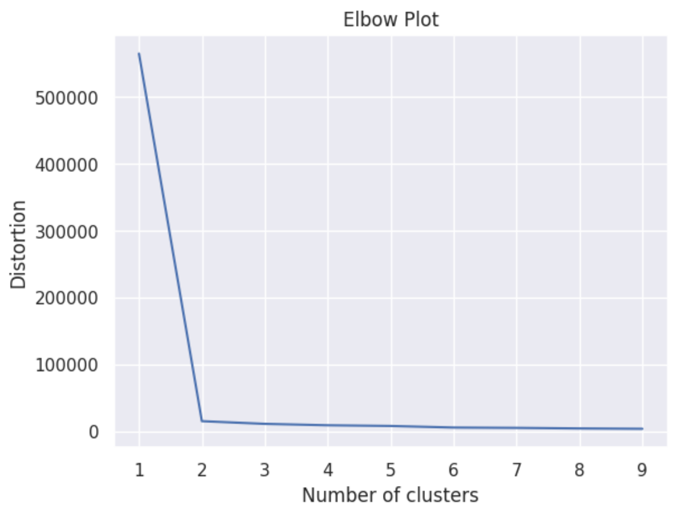


Figure 5.4: KMeans Elbow test for the PCA of the flux-tube inclination variable.

	KMeans			**Agglomerative**		
K	**S score**	**DB**	**CH**	**S score**	**DB**	**CH**
2	0.898	0.145	428761	0.898	0.145	428761
3	0.586	0.757	294424	0.574	0.782	287595
4	0.593	0.662	243422	0.378	1.022	239082
5	0.583	0.713	207238	0.354	0.955	219099
6	0.400	0.887	237197	0.359	0.924	215351
7	0.392	0.917	216689	0.360	0.900	211272
8	0.391	0.875	225758	0.338	0.906	199083
9	0.381	0.901	223637	0.336	0.890	189224

Table 5.4: Validity metrics obtained by different clustering methods on the PCA of the flux-tube inclination variable. Various models were created for each method by varying the number of clusters, K.

c) PCA of the Joint Inputs

The PCA was applied to the joint inputs, R, B and α in this next approach to capture the most relevant features of the input variables, as they would later be used in the prediction task. The results of the elbow test for the KMeans algorithm can be seen in Figure 5.5. It is difficult to determine the optimal number of clusters from this plot. Possible "elbows" include the ones at $K = 3$ and $K = 6$.

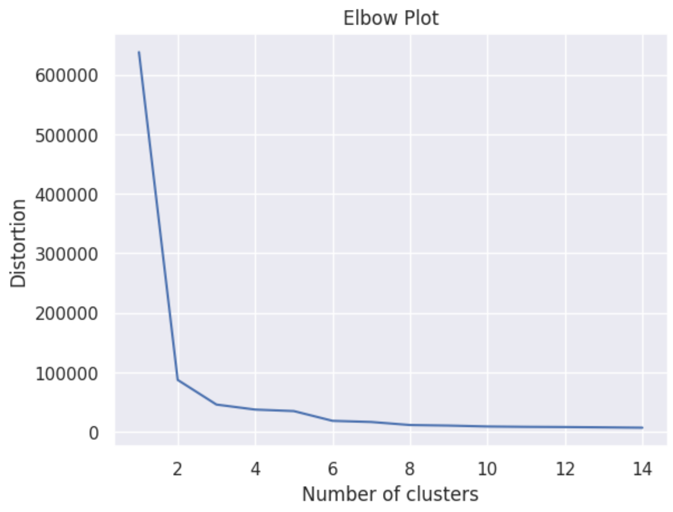


Figure 5.5: KMeans Elbow test for the PCA of the joint inputs.

	KMeans			**Agglomerative**		
K	**S score**	**DB**	**CH**	**S score**	**DB**	**CH**
2	0.777	0.326	74261	0.777	0.326	74261
3	0.647	0.494	75974	0.625	0.503	70716
4	0.605	0.567	62966	0.544	0.564	72768
5	0.553	0.698	50851	0.510	0.620	84236
6	0.471	0.714	78768	0.464	0.704	86339
7	0.465	0.747	73566	0.427	0.789	83135
8	0.438	0.777	91850	0.396	0.819	82333
9	0.435	0.786	87596	0.382	0.858	82094

Table 5.5: Validity metrics obtained by different clustering methods on the PCA of the joint inputs. Various models were created for each method by varying the number of clusters, K.

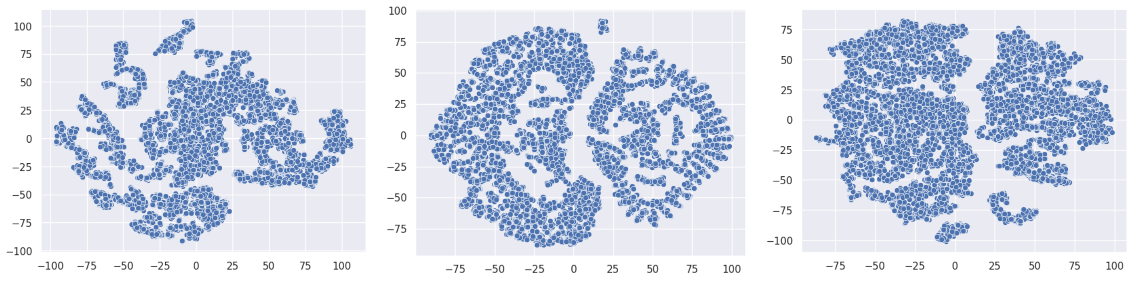
To draw better conclusions, the validity metrics were calculated for each method. The results of these tests can be seen in Table 5.5. For the KMeans algorithm, the highest Silhouette score and

DB index are from $K = 2$. This conflicts with the results of the elbow test, which indicated that the optimal number of clusters was 3 or 6. The highest CH index of the three is with $K = 6$, but this model has the worst Silhouette and DB scores. At first glance, the optimal number of clusters would be 2 for the KMeans algorithm.

Next, for the Agglomerative method, the best silhouette score is from $K = 2$ with the same results as the previous method. This is because both algorithms generate the same clear division of the dataset into two clusters that can easily be construed from Figure 5.2c. The remaining validity metrics are also very similar to the ones obtained with the KMeans algorithm. However, the clusters generated by the Agglomerative method for $K = 3$ had a smaller CH score than the ones for $K = 2$. This might indicate that this method's best number of clusters is 2.

5.4.4 t-SNE Clustering Approach

The last tested approach was to apply t-SNE to the dataset and then apply the clustering algorithms to the reduced dataset. As with the previous approach, this one was tested on the magnetic field variable, $B[G]$, the flux tube inclination, $\alpha[deg]$, and lastly, on a combination of all the input variables. The embedded 2D representations obtained from the t-SNE algorithm can be seen in Figure 5.6. Looking at the first two, it can be concluded that the clustering task will be more complicated than the PCA representations.



(a) Magnetic Field (B). (b) Flux-tube inclination (α). (c) Joint Inputs (R, B and α).

Figure 5.6: t-SNE applied to the different variables. (a) and (b) represent the t-SNE of the magnetic field variable ($B[G]$) and the flux-tube inclination variable ($\alpha[deg]$), respectively; (c) is the t-SNE of all input variables combined.

As in the previous section, each representation was subjected to the KMeans and Agglomerative clustering algorithms from the *sklearn* library. The same validity metrics were calculated to determine what number of clusters would generate the best clustering.

a) t-SNE of the Magnetic Field

The elbow test for the magnetic field variable was inconclusive, as the elbow was unclear. Possible values could be with 4 or 6 clusters. The highest Silhouette score is obtained with $K = 2$, but the lowest Davies-Bouldin index is obtained with $K = 4$, with the highest Calinski-Harabasz index of the first three results.

	KMeans			**Agglomerative**		
K	**S score**	**DB**	**CH**	**S score**	**DB**	**CH**
2	0.323	1.265	6275.825	0.361	1.141	7475.524
3	0.404	0.863	10259.411	0.349	0.883	7335.993
4	0.409	0.838	11029.298	0.386	0.848	9553.107
5	0.400	0.867	11319.886	0.387	0.853	10085.435
6	0.424	0.756	12010.235	0.390	0.777	10395.708
7	0.406	0.831	11557.172	0.358	0.842	10453.778
8	0.402	0.798	11639.116	0.346	0.823	10364.565
9	0.397	0.792	11998.942	0.357	0.827	10653.976
10	0.403	0.785	12363.613	0.373	0.820	11237.525

Table 5.6: Validity metrics obtained by different clustering methods on the t-SNE of the magnetic field variable. Various models were created for each method by varying the number of clusters, K.

The validity metrics from Table 5.6 show a clear candidate for the KMeans method at $K = 6$, which has the best scores of every other model. This disambiguates the results of the elbow test conducted previously. For the Agglomerative clustering method, the results aren't as pronounced. The best silhouette and DB scores were obtained for 6 clusters. Despite the CH index not being as higher as the rest, there is not much variation of this metric above 5 clusters. The results of this test indicate that the optimal number of clusters is 6 for both clustering methods.

b) t-SNE of the Flux-tube Inclination

In line with the results from the experiments on the magnetic field, the elbow test for the α variable is also difficult to interpret. The elbow is not clear enough to be able to reach an estimation.

The validity metrics in Table 5.7 also fail to indicate the optimal number of clusters for the KMeans algorithm. This can be because the algorithm is not suited to handle this representation, as it is very complex. The same can be said of the Agglomerative method, with all entries having very similar silhouette scores. A comparison of both DB and CH scores indicates that the number of clusters for both methods can be 8 or 9, but there is no clear indication of which is best.

c) t-SNE of the Joint Inputs

Contrary to the other two variables, the elbow test for the joint inputs was very clear with an elbow at $K = 3$, indicating that the optimal number of clusters is 3. This is also corroborated by the validity metrics in Table 5.8. For the KMeans algorithm, the highest Silhouette score and

	KMeans			**Agglomerative**		
K	**S score**	**DB**	**CH**	**S score**	**DB**	**CH**
2	0.367	1.142	7613	0.343	1.112	6787
3	0.410	0.803	10814	0.385	0.864	9815
4	0.389	0.817	11185	0.350	0.887	8899
5	0.382	0.877	11611	0.336	0.945	9269
6	0.365	0.866	11396	0.366	0.867	9925
7	0.396	0.779	12086	0.351	0.861	10223
8	0.394	0.754	12678	0.338	0.799	10276
9	0.395	0.776	12365	0.325	0.785	10429
10	0.398	0.789	13223	0.341	0.853	10870

Table 5.7: Validity metrics obtained by different clustering methods on the t-SNE of the flux-tube inclination variable. Various models were created for each method by varying the number of clusters, K.

DB index are $K = 2$, which also has the highest Calinski-Harabasz index of the first three results. Another possible division would be 7 clusters because of the similar silhouette and DB scores and a higher CH score.

For the Agglomerative method, the most appropriate number of clusters is 3. Despite having a worse DB index than $K = 4$, the silhouette and CH scores are better. This indicates that the optimal number of clusters is 3 for both clustering methods.

5.4.5 DBSCAN Experiments

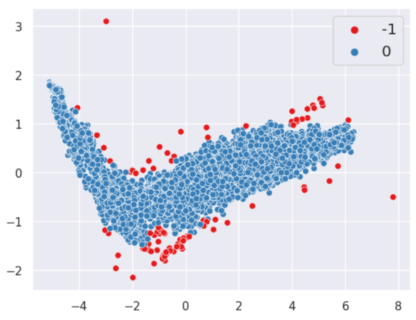
Additional experiments were carried out with the DBSCAN clustering method on both PCA and t-SNE representations. This algorithm was chosen because it can perform the clustering task and detect anomalies simultaneously.

The tests were carried out in each of the representations from Sections 5.4.3 and 5.4.4. The only experiment that yielded somewhat good results was when DBSCAN was applied to the PCA of the magnetic field variable.

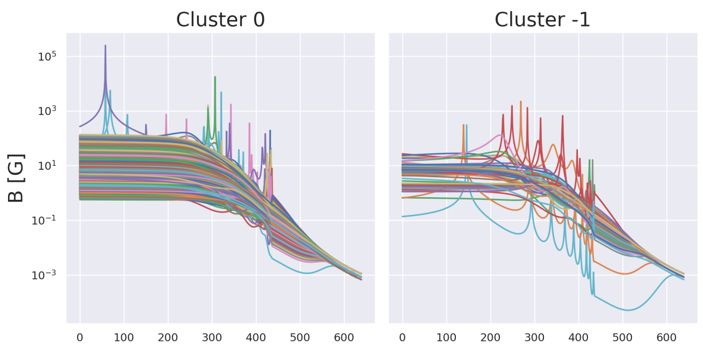
Figure 5.7a shows the clustering obtained for this method, and Figure 5.7b the separation of the magnetic field lines per cluster. From there, it can be concluded that this method can serve as a basic anomaly detection approach. The method can detect some anomalies in the magnetic field variable (cluster -1); however, many anomalous lines remain in the final dataset (presented in cluster 0).

	KMeans			**Agglomerative**		
K	**S score**	**DB**	**CH**	**S score**	**DB**	**CH**
2	0.383	1.067	7701	0.328	1.068	5684
3	0.430	0.791	11317	0.400	0.851	10083
4	0.384	0.829	10493	0.370	0.790	9549
5	0.370	0.918	10792	0.341	0.902	9797
6	0.391	0.856	11249	0.330	0.972	9609
7	0.399	0.798	11645	0.319	1.015	9572
8	0.385	0.806	11612	0.328	0.941	9924
9	0.377	0.850	11576	0.341	0.894	10287
10	0.379	0.820	11369	0.337	0.888	10364

Table 5.8: Validity metrics obtained by different clustering methods on the t-SNE of the joint inputs. Various models were created for each method by varying the number of clusters, K.



(a) DBSCAN clustering



(b)

Figure 5.7: DBSCAN results on PCA of $B[G]$. (a) DBSCAN clustering of the PCA of the magnetic field, $B[G]$; (b) magnetic field lines separated by clusters (-1 is the outliers).

The results from these experiments might have been hindered by using the *QuantileTransformer*, which tends to mitigate the effects of outliers. More appropriate scalers for outlier detection tasks were tested to try and improve the test outcome. Due to several extreme values, these failed to produce valid representations that DBSCAN could use.

5.5 ML Experiments

Following the mostly exploratory clustering experiments, the next step was to apply the clustering algorithms to the dataset and then use the resulting clusters to train different machine learning models. This experiment aimed to determine if models trained on clusters generated by different algorithms could produce better predictions than models trained on the entire dataset.

The reasoning is that by grouping similar data points, the models could learn more specific patterns for each cluster. This is especially useful in this context as the data is very dispersed, and the resulting models would converge to the mean of the dataset.

Due to the difficulty in selecting the most appropriate clustering, a grid-search-based approach was followed for this task. Several clusterings, C, were generated for each method described in the previous sections by varying the number of clusters, K. The same clustering parameters from the previous experiments were used for consistency because they were chosen to be the most appropriate for each method and the dataset at hand.

Then, an ML model, M_i, was trained for each of the generated clusterings, C_i, was trained for each cluster, C_{ij}, where i is the ith clustering and j is the jth cluster of C. Each model, M_i, is trained with the same methodology as in [4]. This time, instead of training a single model for the entire dataset, a model was trained for each cluster, C_{ij}.

Following the methodology in Barros [4], the same validation files were extracted from the training dataset for later use in the MULTI-VP tests. For each cluster, C_{ij}, a train test split of 85/15 % was done to obtain training and testing datasets. The training dataset was then used to train the models M_{ij}, and the testing dataset was later used to evaluate the models in terms of performance. In the next step, a hyper-tuning random search model from the *keras* [65] library was employed to find the best hyper-parameters for M_{ij}, based on the best test MSE loss. Finally, the best model was saved for later use in the MULTI-VP experiments.

The generated models were saved along with their statistics and the hyper-parameters used. The results of the MULTI-VP experiments can be seen in Section 5.5.1.

5.5.1 Clustering ML Results

The results from the above-mentioned experiments are presented in Table 5.9. Only the top 10 results based on MSE loss measured on the testing dataset are shown. The models are sorted based on the average MSE loss in each experiment. This is obtained by averaging the MSE loss of each model, M_{ij}, for each cluster, C_{ij}, of each clustering, C_i. The models are sorted from lowest to highest average loss. The first column shows the model ID, which briefly characterizes the model,

Model ID	Dim. Reduct.	Variable	Method	K	Avg. Loss	std Loss
tsne_agg_2	TSNE	Joint	Agglom.	2	0.0108	0.0058
pca_kmeans_2	PCA	Joint	KMeans	2	0.0113	0.0016
pca_kmeans_3	PCA	Joint	KMeans	3	0.0119	0.0037
pca_agg_2	PCA	Joint	Agglom.	2	0.0125	0.0012
tsne_kmeans_mag_3	TSNE	Mag. Field	KMeans	3	0.0125	0.0085
tsne_agg_8	TSNE	Joint	Agglom.	8	0.0127	0.0077
tsne_agg_alpha_2	TSNE	Alpha	Agglom.	2	0.0129	0.0003
pca_kmeans_alpha_2	PCA	Alpha	KMeans	2	0.0130	0.0009
tsne_agg_3	TSNE	Joint	Agglom.	3	0.0132	0.0093
tsne_agg_mag_3	TSNE	Joint	Agglom.	3	0.0132	0.0090

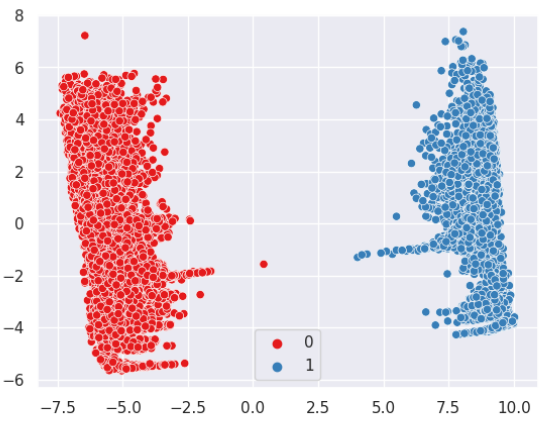
Table 5.9: Results of the Clustering ML Experiments. The table shows the ten best models based on the average MSE loss. The models are sorted by the average loss, from lowest to highest. The columns show the model ID, the dimensionality reduction method, the variables used, the clustering method, the number of clusters, K, the average loss of the models, the standard deviation of the loss and the sum of the loss and the standard deviation.

the dimensionality reduction method, the variables used in the clustering, the clustering method, the number of clusters, K, and the models' average loss and standard deviation.

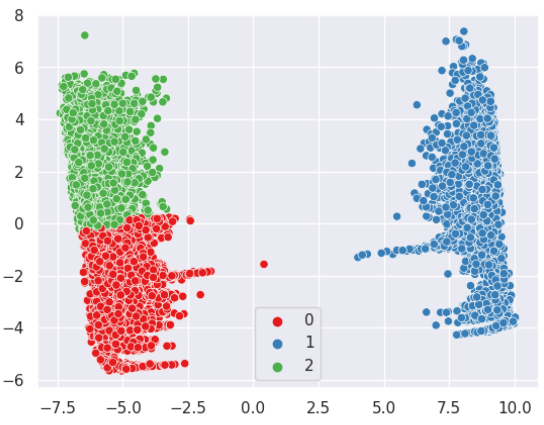
The experimental results (Table 5.9) show that the best models were obtained with the clusterings of the TSNE joint inputs. These divisions were obtained by running the Agglomerative Clustering method with $K = 2$. A closer look at the standard deviation of this model shows that it is the highest of the top 3 models, indicating a large disparity between the losses of both models.

A more in-depth analysis of this outcome with the validity methods from Table 5.8 contradicts the results from this experiment. Despite it having the lowest loss of all the entries, the clustering quality was subpar for this instance. The inappropriate clustering of the dataset might be the reason for the high standard deviation of the models, as the data was not grouped correctly. A visualization of the clustering from this method is presented in Appendix A.1.

The second-best results were obtained with the KMeans clustering of the PCA of the joint inputs. The standard deviation of this model is the lowest of the top three models, which indicates that the losses of the models are very similar. Despite the average loss not being as low as with the first method, it is still not far off. An analysis of the clustering results from the previous section (Table 5.5) shows that the clustering quality was also very good. This indicates that the models could learn the patterns of the clusters and that the clustering was appropriate for the dataset. A visualization of the clusters in Figure 5.8a also corroborates this by dividing the dataset into two clusters.



(a) 2 clusters



(b) 3 clusters

Figure 5.8: Top clustering results for the KMeans applied to the PCA of the joint inputs. Clusters were obtained with the KMeans algorithm on the PCA of the joint inputs with $K = 2$ on the left and $K = 3$ on the right.

The next best method was delined by training a model for each cluster of the same representation as the previous one but with $K = 3$. Although the average loss is higher than the previous two, the standard deviation is also very low, which indicates that the losses of the models are similar, in contrast with the disparity of the models from the second experiment. The clusters for this experiment can be seen in Figure 5.8b.

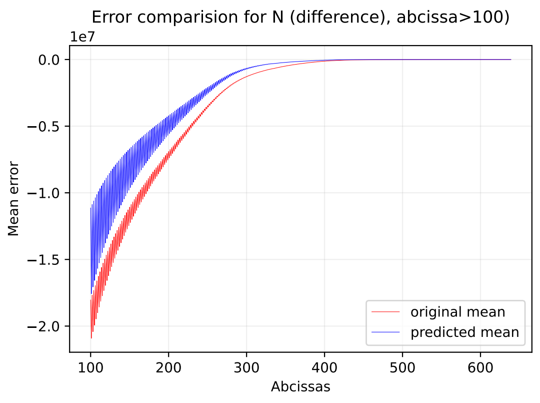
These two experiments serve as the principal contenders for the best models. The results from the previous section were further analyzed to disambiguate the results. As was already concluded, in section 5.4.3.c, the elbow test (Figure 5.5) produced a clear spike at $K = 2$; however, the elbow might be at $K = 3$. The validity metrics (Table 5.5) for the KMeans clustering indicate that the best separation is with two clusters just from the silhouette score and DH index. Despite this, the CH index is slightly lower than with $K = 3$.

Taking all these factors into consideration, the models that were selected were the ones obtained with the KMeans clustering with $K = 3$. In addition to the reasons mentioned above, this clustering provides an equal division of the dataset, in contrast with the other clustering with $K = 2$, which has a very large and small cluster.

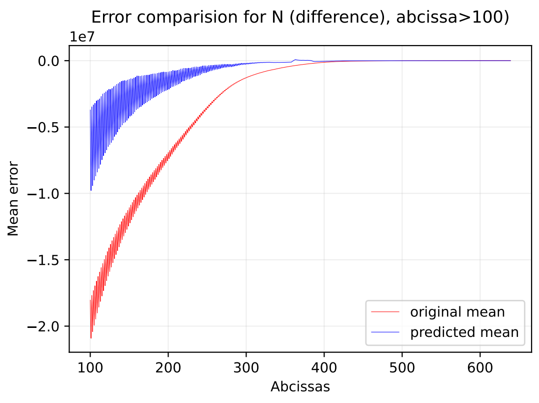
An analysis of the remaining results shows that no model trained on other clustering algorithms produced better results than the ones obtained with the PCA and TSNE. Most of the top results (7 out of 10) were obtained with the clusterings of the joint input variables. In addition, every clustering was obtained by dimensionality reductions of the original dataset. This might indicate that the clustering methods trained on the original dataset could not find the best clusters for the dataset.

5.5.2 MULTI-VP Results

This section presents the results of the MULTI-VP simulation with the initial conditions of the developed prediction models.



(a) Baseline model



(b) Clustering models

Figure 5.9: Abscissa wise estimate error comparison of $n[cm^3]$. (a) comparison of the baseline model and initial expert estimates; (b) comparison of the estimates with the clustering approach and expert predictions.

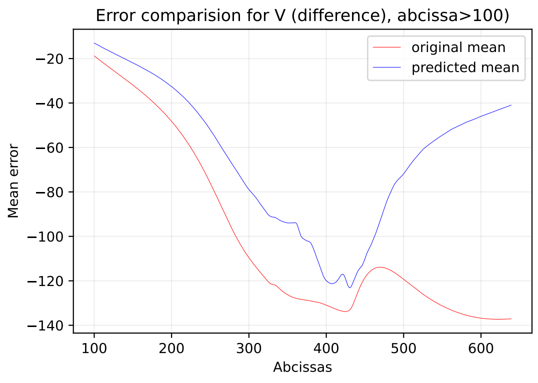
Figure 5.9a shows the abscissa-wise mean error comparison on the $n[cm^3]$ variable. This chart displays the difference between the estimates provided as input for the simulation and its final estimates. The red line displays the error between the initial expert guesses and the final prediction. Similarly, the blue shows the difference between the outputs of the prediction model and the final estimates of the simulation. The results prove that the initial estimates generated by the prediction model in [4] are closer to the expected result of the simulation than the expert estimates.

The results obtained with the three cluster models (Figure 5.10b) show a significant decrease in the mean error of the predictions (blue) when compared to the original estimates (red). In addition, the mean error of the predictions from these new models is lower than that obtained with the baseline model. This shows that the approach resulted in $v[cm]$ values closer to the final MULTI-VP simulation.

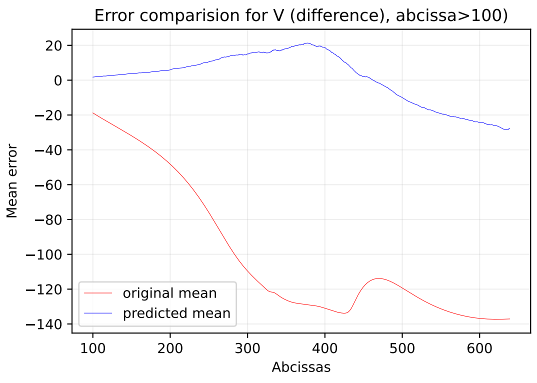
For the $v[km/s]$, the baseline model (Figure 5.10a) shows similar mean error for its predictions and the original expert estimates. In contrast, the cluster-based approach predicted values of $v[k/m]$ closer to the final MULTI-VP simulation, as can be seen by the significantly lower mean error for these when compared to the expert estimates.

Figure 5.11a compares the baseline predictions for the $T[MK]$ variable with the original expert estimates. The baseline shows a slight decrease in the mean error in line with the original guesses. The clustering models (Figure 5.11b) faired much better than the baseline model, with initial estimates for the temperature being much closer to the simulation results.

Despite having provided initial flow estimates closer to the final simulation solution, the new estimates failed to reduce the overall computation time of MULTI-VP. The baseline model had a

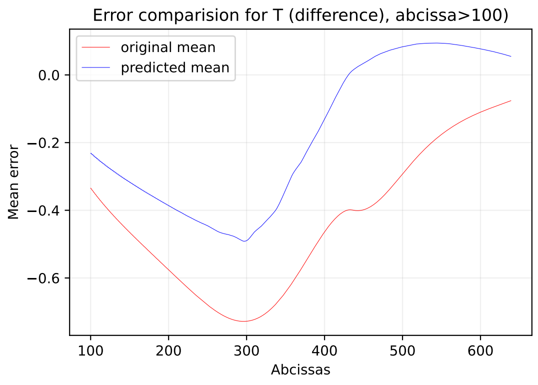


(a) Baseline model

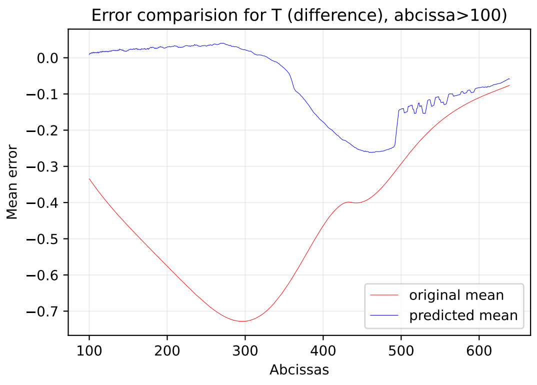


(b) Clustering models

Figure 5.10: Abscissa wise estimate error comparison of $v[km/s]$. (a) comparison of the baseline model and initial expert estimates; (b) comparison of the estimates with the clustering approach and expert predictions.



(a) Baseline model



(b) Clustering models

Figure 5.11: Abscissa wise estimate error comparison of $T[MK]$. (a) comparison of the baseline model and initial expert estimates; (b) comparison of the estimates with the clustering approach and expert predictions.

mean speedup of 1.06 over the initial method of running MULTI-VP with the expert initial flow guesses, while the new clustering approach yielded a speedup of 1.05.

5.6 Summary

This chapter explained the use of clustering methods on solar wind profiles for improving initial flow estimates that were later tested on MULTI-VP. Sections 5.1 and 5.2 provide an overview of the clustering methods and validity measures that were used. The multiple clustering approaches tested for this data are detailed in Section 5.4. The choice of a single approach for testing is explained in 5.5, followed by the training process of the prediction models.

Lastly, Section 5.5.2, shows the results of the MULTI-VP simulation when the predictions from the new method are used as initial flow estimates. From there, it can be concluded that clustering the dataset and then training a separate model for each cluster generated estimates closer to the final solution of MULTI-VP. Unexpectedly, even with initial estimates closer to the final ones, the new approach failed to obtain a higher mean speedup than the baseline.

Chapter 6

Adversarial Anomaly Detection

In recent years, the use of Generative Adversarial Networks (GANs) has been explored in the context of anomaly detection. The main idea is to train a GAN to learn the normal distribution and then use one or all modules to detect anomalies. This chapter presents the main concepts of GANs and how they can be used for anomaly detection in solar wind profiles. Additionally, every experiment that was undertaken with this purpose will be explained.

The chapter is organized as follows. Section 6.1 introduces the GANs training process, common problems and their application for anomaly detection. Section 6.2 presents the experiments undertaken in this thesis to detect anomalies in solar wind profiles. Section 6.3 explains the experiments undertaken with the ML prediction model after anomaly detection in the training data. Finally, Section 6.4 provides an overview of the work done in this chapter.

6.1 Generative Adversarial Networks

GANs were first introduced by Goodfellow et al. in their paper "Generative Adversarial Nets" [54]. Since then, many variations of GANs have surfaced and been applied to different areas like human face generation, image-to-image and text-to-image translation, and semantic generation, among others. The original GAN consisted of two models, a generator G and a discriminator D. The task of the first model was to capture the distribution of the data and generate new examples from that distribution. The function of the discriminator D is to distinguish actual samples from the fake data generated by G. The two components play an adversarial game in which G tries to fool D with increasingly realistic examples, and in turn, G tries to detect the fake samples from G. The authors proposed an analogy that would help the problem's dynamics:
The generative model can be thought of as analogous to a team of counterfeiters trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles. ([54])

The problem is formulated as follows:

$$\underset{G}{min}\,\underset{D}{max}\,V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z))] \tag{6.1}$$

where x represents the data, z the latent space, $p_{data}(x)$ is the distribution of the data and $p_z(z)$ is the distribution of the latent points (usually Gaussian) that G uses to generate new samples. GANs can then be defined as a minimax game where D tries to maximize V, as it tries to recognize generated and real images better; and, on the other hand, G wants to minimize the function V because its goal is to fool G as many times as possible.

6.1.1 Common Challenges in the Training Phase

Earlier GAN architectures were very unstable and hard to train. Despite some proposed solutions to these issues ([66, 67]), GANs are still remarkably difficult to train. Following are some of the main problems experienced during this phase.

Mode collapse. Occurs when the GAN is incapable of reaching Nash equilibrium [1] and is a consequence of poor generalization. It can occur when the generator G only creates samples from a subset of the data distribution or only learns part of the distribution. The leading causes for this issue can be attributed to a poor choice of the objective function ([68]). In other words, G focuses on a small subset of samples that consistently fool the discriminator D.

Vanishing gradients. The discriminator D does not provide enough information for G to update its gradients [69]. D can distinguish real samples from fake ones with high confidence, which in turn causes the loss function of G to decrease towards 0. As D gets better, the gradient of G progressively decreases until virtually none of the layers are updated, and G can't generate samples with new distributions. Some solutions for this problem include batch normalization and clipping.

Evaluation metrics. Due to their wide range of applications, no global evaluation function can be applied to every GAN. The evaluation function varies greatly from the context of the problem in which the GAN was used. In some instances, like image generation, the principal evaluation criteria are still done qualitatively (the outputs are analyzed by humans, who determine their quality). Evaluation functions are an essential part of machine learning and allow for the correct conclusions to be made [68].

6.1.2 Anomaly Detection with GANs

Several approaches can be employed when using GANs for anomaly detection. The simplest one is to train both G and D on the normal data distribution and then use the latter to classify

[1]Can be explained by the following analogy: Two players, Alice and Bob, chose strategies A and B; Alice has no other strategy to maximize her goal better, and Bob has no different strategy other than B to maximize his goal in response to Alice's choice (`https://en.wikipedia.org/wiki/Nash_equilibrium`)

new samples as normal or anomalous. This is not always possible as the discriminator only learns to distinguish between samples from the generator and real ones. There is no guarantee that the discriminator will be able to distinguish between normal and anomalous samples, as it might have only learned specific characteristics of G that are not present in the anomalous samples.

To circumvent this issue, some studies [48, 40] proposed changing the focus of the vanilla GAN to be more appropriate for anomaly detection tasks. Instead of learning the normal distribution G is forced to only generate anomalous samples. By forcing G to generate samples close, but not equal to the normal distribution, the discriminator D will learn to distinguish between normal and anomalous samples. To accommodate this, small changes to the loss functions of D and G are made. The results are promising, but by altering the loss functions, the new implementations might have new issues that were not experienced in other implementations.

Other studies [36, 38, 56] have proposed employing G as a way to detect anomalies. These approaches are usually based on the reconstruction error of G to detect anomalies, which is calculated by measuring the distance of the current sample to its closest reconstruction created by G. Then, an anomaly detection score can be construed with this error, either alone or in combination with the classification of the discriminator. Defective samples will, in theory, have higher reconstruction errors and will be classified as anomalous by the discriminator, as both modules didn't learn to generate or classify them correctly.

More recently, the use of adversarial autoencoders (AAE) has been proposed [38, 45] to simultaneously encode/decode an input sample and constrain its latent space representation to a prior distribution. The encoder and decoder are trained simultaneously with the discriminator, which is trained to distinguish between the latent space distribution and the prior distribution. The encoder is then used to encode new samples to the learned latent space. The discriminator classifies the latent representation of the sample as being real or fake. Similar to all other approaches, the anomaly score can be calculated by using only the reconstruction error of the decoded sample, or by combining it with the classification of the discriminator.

Some of these methods will be employed in the experiments of this thesis. The next section explains the different approaches used to detect anomalies in solar wind profiles with the help of adversarial learning.

6.2 Experiments

This section explains the experiments performed to detect anomalies in solar wind profiles. In total, five different architectures were used for this task. One linear GAN, three RNN-based GANs (two of which are preliminary experiments), and one adversarial autoencoder.

6.2.1 Anomaly Scores

Three anomaly detection methods are used in each experiment to detect anomalies in solar wind profiles. At the end of each experiment, the most suitable anomaly detection method was chosen based on a qualitative analysis of the results. Each approach generates a different normality score for a given sample. The scores are then used to determine a threshold of normality based on the percentage of anomalies in the dataset. This is a hyperparameter that must be determined by the user.

The first and simplest score is the classification of the discriminator, D. The values of the classification can range from 0 to 1, where values equal to 1 indicate a real sample and values closer to 0 indicate a fake label. The classification score is calculated by feeding the sample to the discriminator and obtaining the classification value. The abnormality classification score of a sample, x, is calculated as follows:

$$D_s = 1 - D(x) \tag{6.2}$$

The second score is the reconstruction error of the generator, G, and is based on the reconstruction technique of MAD-GAN [36]. The reconstruction process, expressed in algorithm 1, occurs iteratively for each ith sample, x^i, in the testing dataset. Like in the original method, the optimal latent representation for x^i is obtained by first sampling a latent space variable, z^k. Then, for a predefined set of iterations, j, the generator G is used to generate a batch of reconstructions $G(z^k)$. Contrary to [36], the distance between the original sample x^i and the reconstructed one, $G(z^k)$ is calculated using the MSE loss function. In the next step, the residuals of the MSE functions are averaged and used to update the parameters of z^k. After j iterations, the latent representation, z^k, that results in the best reconstruction of x^i is returned.

```
Algorithm 1: MSE Reconstruction
    Input
        $x^i$        Input Batch
        $n$          Number of iterations
    Output
        $z^k$        Optimal latent representation
        $R_{err}$    MSE Reconstruction error

  Get a sample from the latent space;
  $z^k = random\_sample()$;
  for $j = 0$ to $n$ do
      Reconstruct batch from $z^k$ and calculate loss;
      $R_{err} = MSE(x^i, G(z^k))$;
      Update latent representation based on $R_{err}$;
      $z^k = update(z^k, R_{err})$;
  end
```

The reconstruction error is calculated directly with the MSE loss between the real and reconstructed batches from the optimal latent representation, z^k. The reconstruction error is calculated as follows:

$$R_{err} = MSE(x^i, G(z^k)) \tag{6.3}$$

The next score is a combination of the previous two scores as a way of taking advantage of both the discriminator and the generator in the detection process. The reconstruction process is similar to the previous one, but instead of using only the MSE loss, the classification score, D_s of the discriminator, is also used. The new altered loss function is defined as follows:

$$RD_{err} = MSE(x^i, G(z^k)) \times D_s \tag{6.4}$$

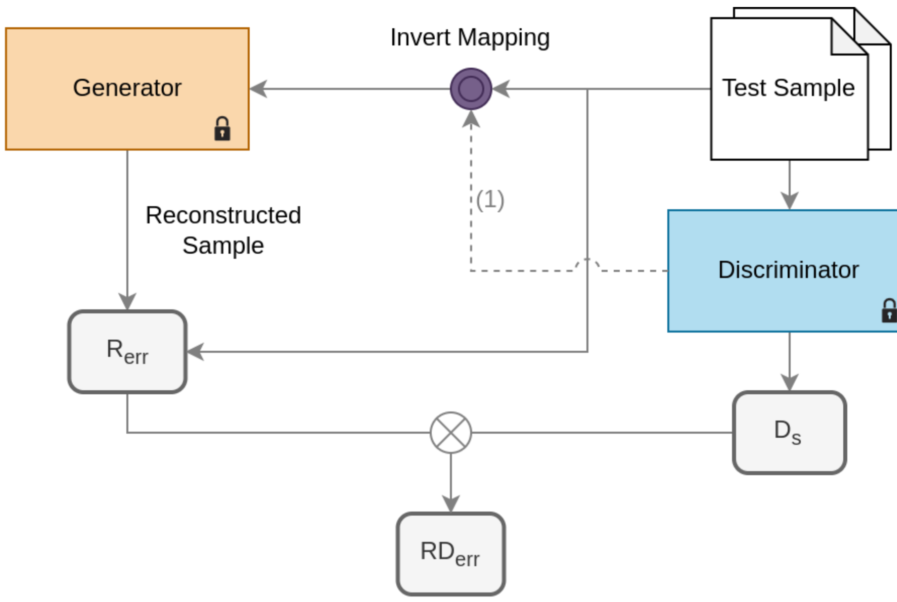
The reconstruction process with the discriminator input is shown in Algorithm 1. It is very similar to Algo. 1, but with the addition of the discriminator classification score to the loss function. In each reconstruction step, j, the discriminator classification, D_s, for the reconstructed batch is obtained. Then, the error between the reconstructed batch and the original data is measured with the MSE loss function. In this step, the results of the discriminator classification are multiplied by the D_s score from the previous step. Finally, the parameters of z^k are updated in accordance with the error that was obtained.

Algorithm 2: MSE-Discriminator Reconstruction

```
Input
    $x$          Input Batch
    $n$          Number of iterations
Output
    $z^k$        Optimal latent representation
    $RD_{err}$   MSE-Discr. Reconstruction error

Get a sample from the latent space;
$z^k = random\_sample()$;
for $j = 0$ to $n$ do
    Obtain D classification for reconstructed batch;
    $D_s = 1 - D(G(z^k))$;
    Reconstruct batch from $Z^k$ and calculate loss;
    $RD_{err} = MSE(x^i, G(z^k)) \times D_s$;
    Update latent representation based on $RD_{err}$;
    $z^k = update(z^k, RD_{err})$;
end
```

A simplified representation of the workflow for obtaining the anomaly scores is shown in Figure 6.1. This process is done by extracting the pre-trained G and D with their layers frozen. Each batch, x^i, is passed through D to obtain the classification score, D_s. The reconstruction process from algorithms 2 and 1 is performed in the invert mapping step to obtain the optimal latent representation, z^k, for the given test sample. After this, R_{err} is calculated by comparing the



[(1)] Only applied to the reconstruction method in algorithm 2.

Figure 6.1: Anomaly detection workflow with each defined reconstruction method. The invert mapping obtains the optimal latent representation, z^k, from the reconstruction error.

reconstructed sample with the original one. The final RD_{err} can be obtained by combining R_{err} with D_s.

6.2.2 Linear GAN

The first architecture was a linear GAN, which is a simple GAN with a linear generator and discriminator. Several experiments were performed with different configurations of the architecture. The use of BatchNorm in either module was tested to try and stabilize the training process. This proved ineffective, making it difficult for G and D to converge. Because of this, it was not included in the final implementation. In addition, the number of intermediate stacked linear layers was varied during the tests to try and find a configuration that worked best for this problem.

Besides these changes, different activation functions were also tested. The activation functions used were ReLU, LeakyReLU, and Tanh, with a combination of them in the intermediate and output and input layers. Like other GAN architectures, the LeakyReLU activation function was chosen to prevent the gradient from vanishing and improve the learning process's overall stability. Most configurations that included the Tanh activation function in the output layer of the generator were discarded, as it proved ineffective during the anomaly detection phase.

The final architecture is shown in Figure 6.2. Both the generator and discriminator are built with linear layers. The generator, G, consists of an input layer that takes as input a latent space variable, z, with L_P features and outputs a sample with the same dimension as the input. The first layer linearly transforms the input noise vector and projects z into lower dimensional space with 640 dimensions. The output of this layer is then passed through a LeakyReLU activation function. Next, the resulting features from this step are passed through two stacked linear layers,

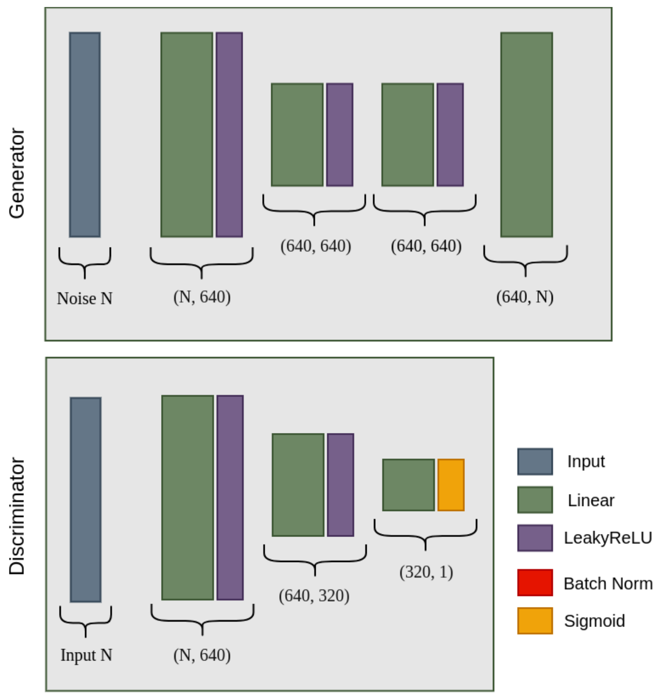


Figure 6.2: Linear GAN Architecture

each with 640 dimensions. In the last layer, the features from the previous step undergo a linear transformation to project them into the same dimension as the input sample. As was previously said, no activation function is used in the output layer of the generator, as it was ineffective during the detection phase.

The discriminator, D, consists of an input layer that takes as input a sample with L_P features and outputs a scalar value. The objective of D is to progressively learn the most important features of the data to perform binary classification of the input sample as being real or fake. The first and subsequent layers transform the input data into increasingly smaller dimensions. In the output layer, the features from the previous step are projected into a single scalar value. This value is then passed through a sigmoid activation function to obtain the final classification score of each sample.

Training

Both modules were trained with the Adam optimizer with a learning rate 0.0001 for the generator and 0.0002 for the discriminator. This was done to prevent the discriminator from overpowering the generator during training. The weights of each linear layer were initialized with the He initialization function from [70].

Two models were trained, the first one, M_i, to detect anomalies in the inputs of the MULTI-VP dataset and the other, M_o, for the outputs (refer to section 4.3). Both models were trained over 300 epochs with a batch size of 128. In both approaches, the training data was scaled with the *MinMaxScaler* to preserve the variation of extreme values in the data. The training data consisted of a matrix with $N \times L_P$ dimension, where N is the number of training samples and L_P is the number of features per profile. Like in the clustering experiments, the selected validation data was excluded from the training process for further use in the ML evaluation step.

From Section 4.3, we know that every variable in a profile has 640 features, so the total number of features per sample can be defined as $L_P = k \times 640$, where k is the number of variables in a profile that is used during training.

In M_i, the input data, x, consists of concatenating the $B[G]$ and $\alpha[deg]$ variables of each profile. The radial coordinate radius, $R[R_{sun}]$, was excluded from the process to reduce the number of features the networks needed to learn. In addition, extreme variations in the input data were removed to prevent both modules from learning noisy features that would hinder the detection process. Following the previous notation, the number of features for each sample is $L_{P_i} = 2 \times 640 = 1280$.

In the outputs model, M_o, every output variable of MULTI-VP is used in the training phase. Each sample consists of the concatenation of the density, $n[10^{10}cm^{-3}]$, the velocity, $v[km/s]$, and temperature, $T[MK]$, with a combined number of features per sample, L_{P_o}, of 1920. Contrary to the input model, no extreme variations were removed from the data, as it performed well without this step.

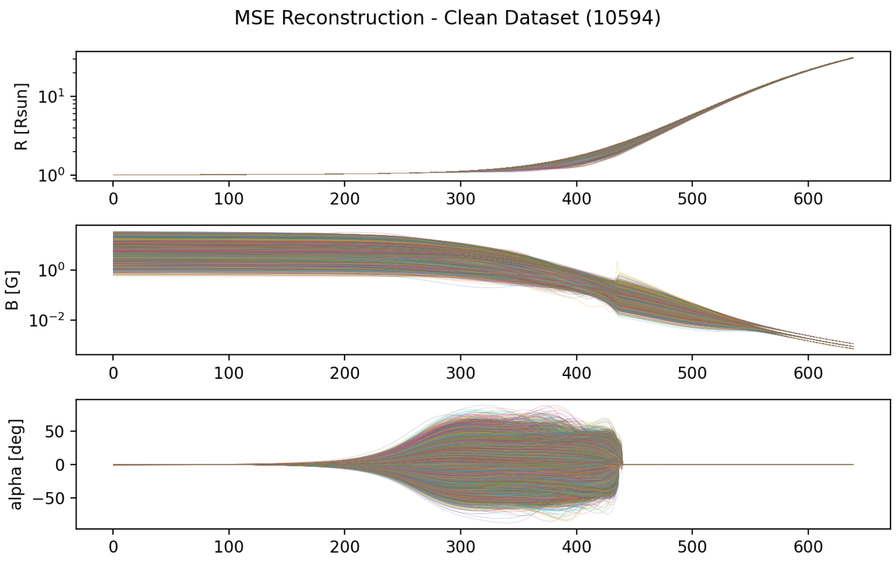
Anomaly Detection

The detection step was carried out for each model with all three anomaly score functions defined at the beginning of this section. Due to the lack of validation metrics, the choice of method was based on the visual inspection of the results. The anomaly scores' stability and the dataset's quality without anomalies were considered.

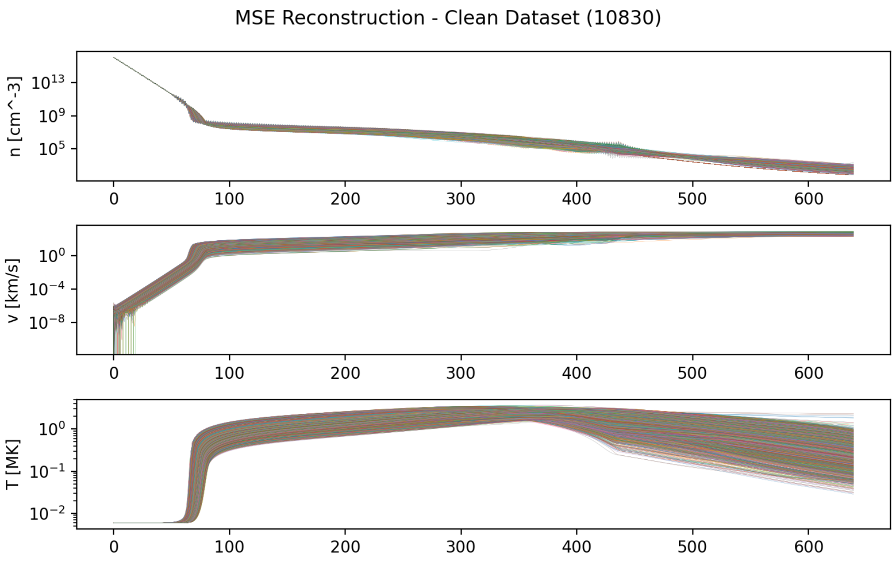
Considering these criteria, the best results were obtained with the reconstruction error, R_{err}. The other two functions, D_s and RD_{err}, were very unstable and were not able to detect the anomalies as well as R_{err}. The filtered input and output variables can be seen in Image 6.3a and 6.3b, respectively.

The first image (Figure 6.3a) resulted from removing 10% of the files from the original dataset based on anomaly scores. As was previously said, the anomaly scores were obtained by training the GAN model, M_i, in the input variables and then using G to calculate the anomaly scores of each sample in the testing dataset. The results show that the data is mostly clean, with some anomalies remaining in the magnetic field, $B[G]$, variable. For the output variables (Figure 6.3b), only 8% of the files were excluded based on anomaly scores. The resulting output variables are mostly clean, with some abnormalities in the density, $n[cm^3]$, variable.

Overall, the experiments with this architecture prove that it can detect anomalies in the input and output variables of MULTI-VP. However, the percentage of files that need to be removed (i.e. the False Positive rate) is still very high. This might indicate that the current architecture might not be ideal for the task, as seemingly "normal" profiles might be excluded from the prediction step.



(a) Input Variables



(b) Output Variables

Figure 6.3: Resulting datasets after the anomaly detection step with the linear GAN architecture on the inputs and outputs of the MULTI-VP dataset.

6.2.3 Preliminary RNN-based GAN Experiments

The next experiments were with RNN-based GANs. These experiments are based on the assumption that consecutive profiles in the MULTI-VP dataset are somewhat similar and have small variances. This can be closely associated with other studies for detecting anomalies in time-series data (refer to Section 3.2).

Several architectures were designed and tested first with the RNN-based GANs. These failed to produce meaningful results as the GANs were very unstable during training, and the anomaly scores could not detect any anomalies in the data. This might be due to the issues associated with traditional RNN-based architectures, such as the vanishing gradient problem [12].

Considering this, the next batch of tests was carried out with GRU-based GANs. GRU NNs, like LSTM NNs, can learn long-term dependencies in the data sequences. However, they are less resource intensive and faster to train than LSTM NNs as they require fewer parameters. The results obtained with the GRU architectures were tested with LSTM ones to ensure that they would not change as per the choice of architecture. The performance of both architectures was similar, with the GRU-based GANs being slightly faster to train. Because of this, the results presented in this section are from GRU GAN architectures, as they are more efficient than LSTM ones, and reach similar outcomes.

Stacked GRU GAN

Before reaching a stable RNN architecture, several experiments were carried out with different architectures. The first attempts employed multilayer GRUs for *G* and *D*. Several iterations were tried with several recurrent layers and hidden sizes. Several tests were carried out with multiplayer GRUs, due to the high number of features for the input data. However, the results were unsatisfactory as the GANs were very unstable during training. *D* would always outperform *G* in this phase, causing the latter to collapse and produce meaningless results. This occurred even when *D* had significantly fewer layers than *G*. In addition, reducing the learning rate of the discriminator didn't seem to affect the instability of the GANs.

With this in mind, other network configurations were tested by reducing the number of features given to the stacked GRU layers. This was done by applying a linear transformation on the input data, significantly reducing its dimensionality. With this layer, in theory, only the most important attributes of the inputs would be retained and passed to the subsequent layers. This approach managed to improve the stability of the GANs during training, but the results of the detection phase were still subpar with the ones from the linear GAN.

Pyramid GRU GAN

In the second batch of experiments, a pyramid GRU GAN was designed to circumvent the dimensionality issues of the stacked GRU GAN. This architecture consisted of a simple generator with three GRU layers and an output linear layer for activation. The first layers would narrow the input size to smaller dimensions to ensure that only the most meaningful features would be retained. Then the last GRU layer would upscale the outputs from the previous layers into a higher dimension. In the output layer, the resulting features would suffer from a linear transformation that transformed the data into the desired output dimension of the generator. This would then be passed through the Tanh activation function.

The discriminator consisted of only two GRU and a linear output layer. Following the same logic as the generator, the first two layers progressively reduced the number of features of the input data. The output layer would reduce the features from the last layer to just one and then pass it through the sigmoid activation function for binary classification.

As in the previous experiments, the training process of the GAN became more stable; however, the results of the anomaly detection still weren't as good as the ones obtained by the linear GAN.

6.2.4 MAD-GAN

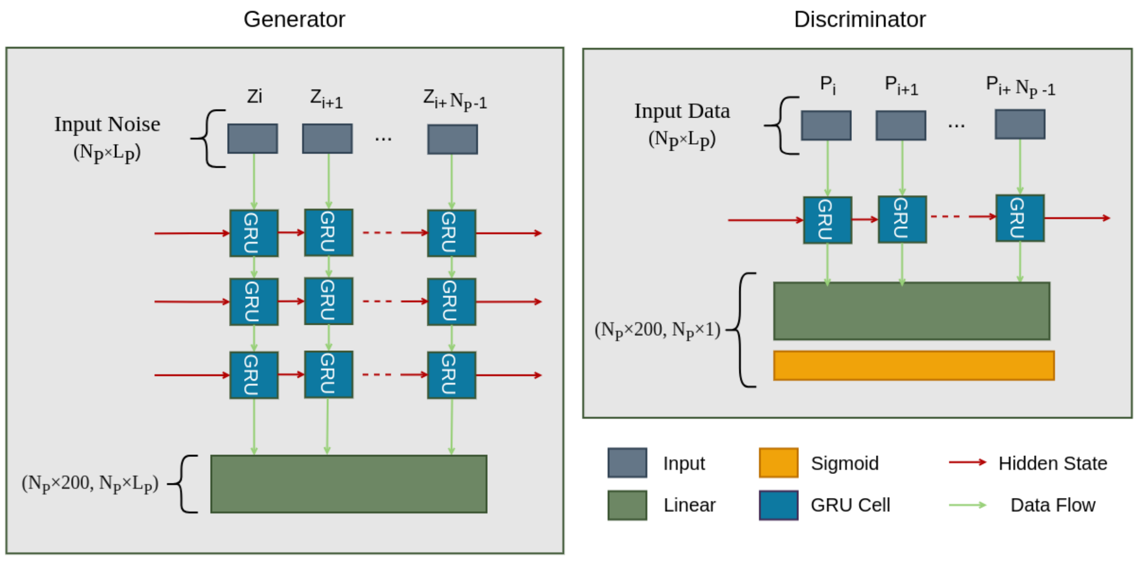


Figure 6.4: Architecture of the MAD-GAN model. The Generator consists of three stacked GRU layers and a linear output layer. The discriminator consists of a single GRU layer and a linear output layer followed by the Sigmoid activation function. Both take as input sequences of size N_P.

The final architecture for this class of GANs was based on MAD-GAN [36]. It was chosen to determine if one of the most famous state-of-the-art RNN GANs for time-series data could be used to detect anomalies in the MULTI-VP dataset. The architecture of the generator and discriminator is shown in Figure 6.4.

Similar to the original implementation [36], the generator, G, consists of three stacked GRU layers and a linear output layer (without an activation function). The discriminator, D, consists of a single GRU layer and a linear output layer followed by the Sigmoid activation function. G takes as inputs windows with N_P latent vectors with L_P features and synthesizes samples with the same dimensions. D takes as inputs windows with N_P real or fake samples with L_P features and outputs a single value between 0 and 1, representing the probability of the input window coming from the dataset or G.

In this implementation, GRU cells were employed instead of LSTM cells used in the original version. It was observed that the choice between the two cell types did not significantly impact the detection capability of the architecture. Considering this, GRU cells were preferred due to their lighter and faster nature compared to LSTMs. Accordingly, each layer of the network was composed of GRU cells, with each cell containing 200 hidden units.

Data Preparation

Unlike the previous architecture, the data is aggregated into windows, W, with consecutive profiles. The size of each window, N_P, is a hyperparameter that needs to be tuned and indicates the number of consecutive profiles, P, fed into the GANs. With this formulation, the ith window, W_i, is defined as the set of the N_P consecutive profiles of the ith profile, P_i, in the dataset, such that $W_i = \{P_i, P_{i+1}, ..., P_{i+N_P-1}\}$.

Each window varies in length (L_P) according to the number of variables used for anomaly detection. Each variable in the dataset has an equal number of features (refer to Section 4.3); therefore, the number of features for a single profile can be expressed as $L_P = k \times 640$, where k is the number of variables being used for the task. From this, the dimensions of the ith window, W_i, can be defined as (N_P, L_P).

As in the previous experiments, two models were designed to detect anomalies in both the input and output variables used by MULTI-VP. In the model trained on the inputs (M_i), only the magnetic field, $B[G]$ variable, was used for the task. This was due to the problems faced in the previous RNN experiments (Section 6.2.3) due to the high dimensionality of the dataset. Additionally, the magnetic field is the variable in the dataset that seems to be most affected by the presence of anomalies. Following the notation adopted in the last paragraph, the dimensions of the ith window, W_i, in the dataset is set to $(N_P, 640)$, where 640 is the number of features L_P of the $B[G]$ variable and N_P is the window size.

In line with the previous experiments, the most extreme values from the input variables were removed from the training process to ensure the best performance of the GAN models. These were then used during the detection phase.

Additionally, the output model, M_o, was trained on the three output variables of the MULTI-VP dataset without removing extreme values. With this, the dimension of each window in the training dataset is set to $(N_P, 1920)$, where 1920 is the number of features L_P of the three output variables and N_P is the window size.

Training

Several configurations were tested for the number of hidden units for the GRU layers. The best results were obtained with 200 hidden units in both modules. The learning rate was set to 0.0001 for G and 0.0002 for D using the Adam Optimizer. The batch size was set to 32, and the number of epochs to 100. The number of profiles per window, N_P, was set to 10. In addition to this, the same training method as in [36] was used. In this method, D is firstly trained for a set number of iterations while G is kept fixed. This ensures that D learns the representation of the real data before training G. After this, G is trained for a set number of iterations while D is frozen. This process is repeated until the end of the training process. The number of iterations was 10 and 5 for D and G, respectively.

As in the previous RNN architectures, the training process was very unstable, and the modules didn't converge due to the high number of input features. Changing the number of hidden features, the learning rate and the number of iterations didn't seem to affect the stability of the training process.

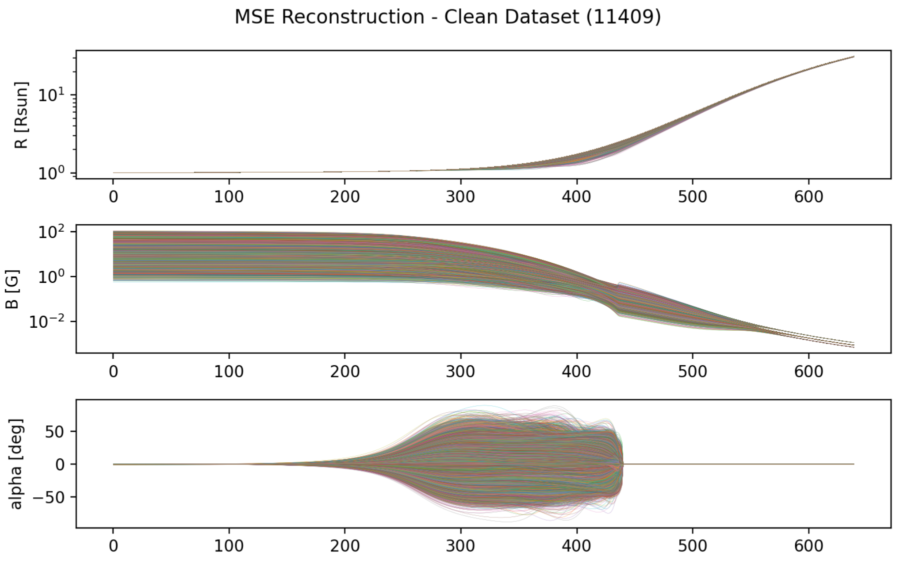
In an attempt to reduce the number of data features and in line with the additional experiments in [36], PCA was applied to the training data. This transformation reduced the number of features, L_P, to a fixed 100 features per profile. For the model M_i, the initial 640 features from the magnetic field variable were reduced to 100 features. For M_o, the 1920 features from all the output variables were reduced to 100 features. With this, the dimension of both training windows was set to $(N_P, 100)$.

Anomaly Detection

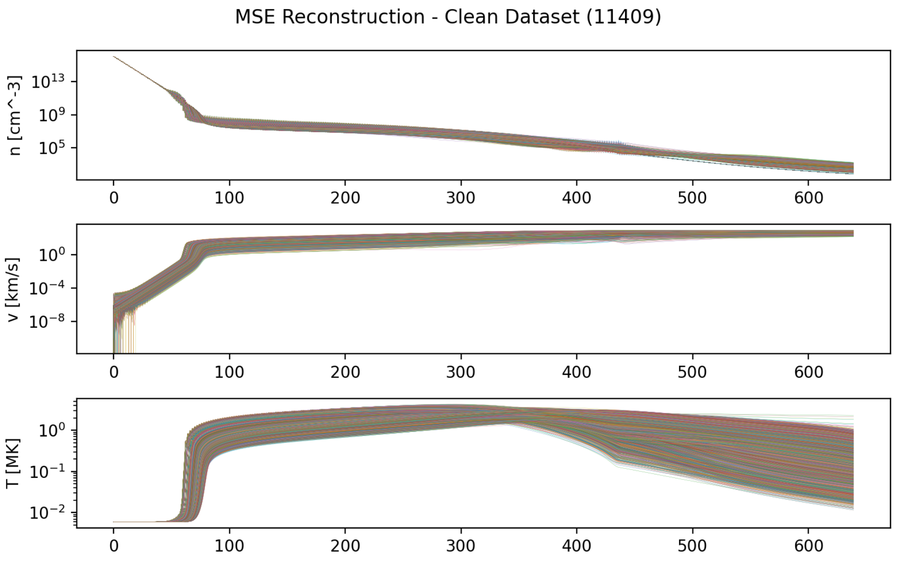
The detection step for both models was carried out with the three anomaly functions defined earlier. Every anomaly score function provided overall good results, showcasing the method's stability. Despite this, the best results were obtained with the R_{err} function. The results of the anomaly detection on the inputs as well as on the outputs are shown in Figure 6.5. These were obtained by training the models with the PCA-transformed data and then applying the anomaly detection step to the testing data with the model from the previous stage.

Figure 6.5a shows the anomaly detection results on the input variables. At first glance, the results are very similar to those obtained with the linear GAN; however, MAD-GAN only required a threshold of 3% top anomalous profiles. This means this model is more sensitive to anomalies in the magnetic field variable than the linear GAN model. The same applies to the model trained on the output variables (Figure 6.5b). These results indicate that MAD-GAN is more sensitive to anomalies in both the input and output variables than the linear GAN model, which translates to a lower False-Positive rate.

Note that these results were only possible because of the use of PCA. This might indicate that the experiments in Section 6.2.3 might have worked if the same method had been employed. A possible issue with this approach is that PCA might be removing important features from the data, which could be used to improve the results of the anomaly detection step.



(a) Input Variables



(b) Output Variables

Figure 6.5: Resulting datasets after the anomaly detection step with the MAD-GAN architecture on the inputs and outputs of the MULTI-VP dataset.

6.2.5 Adversarial AE

In the final experiments, an adversarial autoencoder architecture was used. This architecture was chosen to determine if using an autoencoder could improve the results obtained with the previous GAN architectures. The architecture of the generator and discriminator is shown in Figure 6.6.

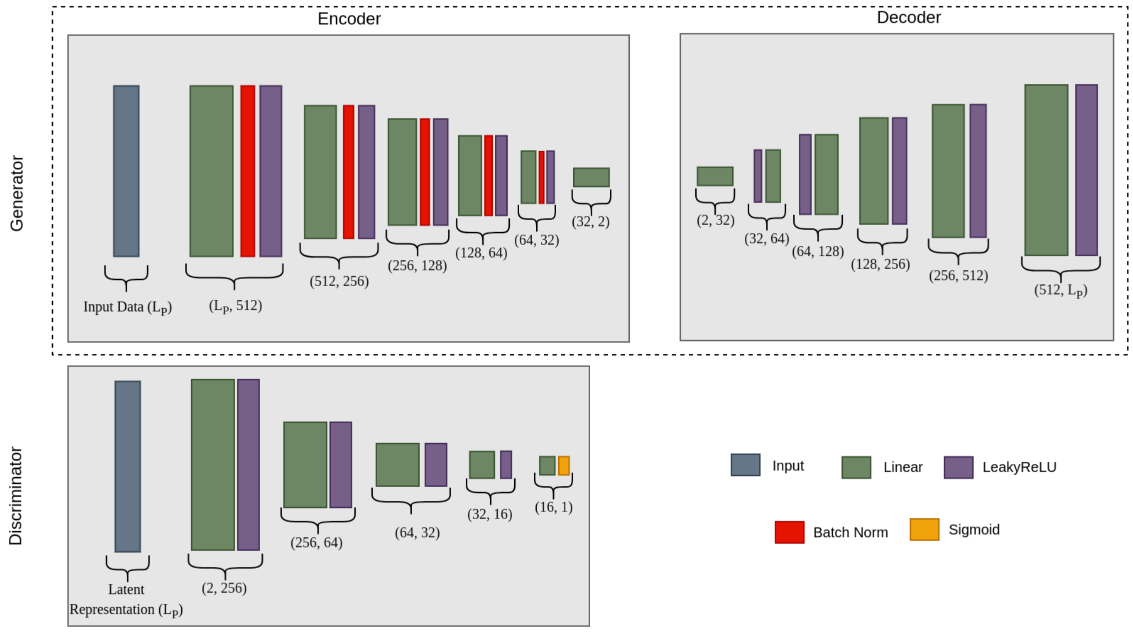


Figure 6.6: Architecture of the adversarial autoencoder. The generator G, consists of an encoder, Enc, and decoder, Dec. The job of the first is to generate a latent representation of the input data, which is then used by the decoder to reconstruct the input. The discriminator, D, takes the latent representation generated by Enc as inputs and outputs a single value between 0 and 1, representing the probability of the input coming from the prior distribution or Enc.

The generator, G, is divided into an encoder, Enc, and decoder Dec. The objective of Enc is to generate a latent representation of the input data. This latent representation is then used by Dec to reconstruct the input data. In adversarial autoencoders, a discriminator model is also introduced to constrain the latent representation to follow a prior distribution. In this case, the discriminator, D, takes the latent representation generated by Enc as input and outputs a single value between 0 and 1, representing the probability of the input coming from the prior distribution or Enc.

Training

The loss function combines the reconstruction loss of the autoencoder (Equation 2.1) with the adversarial loss of the discriminator. The reconstruction loss is the mean squared error between the input to de encoder and the decoder output. The adversarial loss is the binary cross-entropy loss between the output of the discriminator and the prior distribution. The loss function is defined as follows:

$$\mathscr{L}_{AAE} = \alpha \times \mathscr{L}_{AE} + (1-\alpha) \times \mathscr{L}_{ADV} \tag{6.5}$$

where α is a hyperparameter that controls the weight of the reconstruction and adversarial losses.

The training process consists of first training the autoencoder and freezing the discriminator parameters. In this step, the encoder, *Enc*, generates a latent space representation of the original data, which is then reconstructed by the decoder, *Dec*. The parameters of the encoder and the decoder are updated with the equation 6.5. After this, D is trained while the encoder and decoder are kept frozen. In this step, the discriminator is trained to distinguish between the latent representation generated by the encoder and the prior distribution. With this, D is conditioned to learn the prior distribution and to classify the encoded samples more precisely. This process is repeated until the end of the training phase.

The AAE was trained over 400 epochs with the help of the Adam Optimizer, with a learning rate of 0.0002 for D and 0.0001 for G. The batch size was set to 128, and the hyperparameter α was set to 0.999. The prior distribution was set to two distinct 2D Gaussian distributions.

Like in the previous experiments, a model (M_i) was trained for the input variables and another (M_o) for the output variables. For the first, only the magnetic field variable (without extreme values) was used as input, while all the output variables were used for the second.

Anomaly Detection

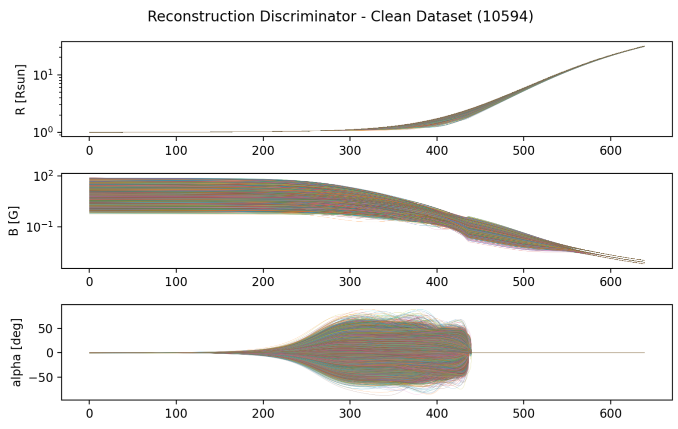
The anomaly detection step was carried out with the same anomaly functions as in the previous experiments; however, the generator reconstruction step is done directly with the autoencoder without the need for the invert mapping in algorithms 2, 1.

The autoencoder proved more than capable of detecting data anomalies without a discriminator. However, as previously stated, this approach aims to take advantage of the reconstruction abilities of the autoencoder along with the discriminator to improve the results of the baseline autoencoder. The results of the anomaly detection step are shown in Figure 6.7.

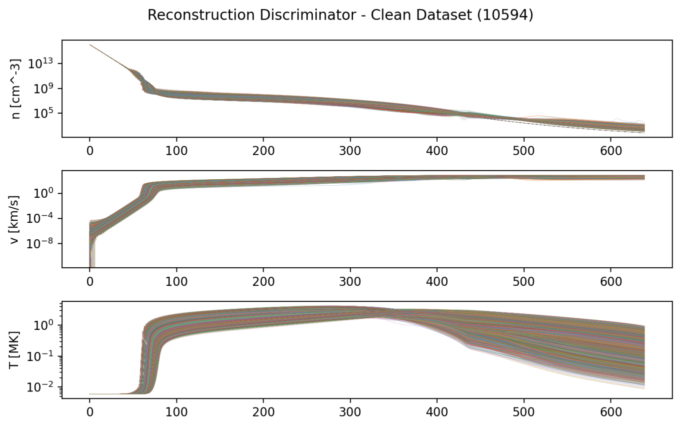
The results show similar outcomes as in the linear GAN implementation. Both images were obtained by removing the top 10% anomalous profiles from the dataset. The results show that the AAE is capable of detecting anomalies in the input and output variables. However, the results are not as good as the ones obtained with the MAD-GAN.

6.2.6 Experiments Summary

A summary of the results obtained in the detection phase with the different GAN architectures is shown in Table 6.1. From an analysis of the results, MAD-GAN cleaned the dataset with the smallest anomaly threshold. The next best architecture was Linear GAN, with the same threshold in the input variables as in AAE but a lower threshold in the outputs. Despite these architectures having the same anomaly threshold for the input variables, some anomalies that resulted from



(a) Input Variables



(b) Output Variables

Figure 6.7: Resulting datasets after the anomaly detection step with the AAE architecture on the inputs and outputs of the MULTI-VP dataset.

	Input Variables			Output Variables		
	Function	Threshold (%)	Anomalies	Function	Threshold	Anomalies
Linear GAN	Rerr	10	1177	Rerr	8	941
MAD-GAN	Rerr	3	352	Rerr	3	352
AAE	RDerr[1]	10	1177	RDerr[1]	10	1177

[1] Direct reconstruction error from the generator ($Enc + Dec$) without algorithm 2.

Table 6.1: Summary of the results obtained with the different GAN architectures.

the linear GAN remained in the dataset. This might indicate that the anomaly threshold for this approach should have been higher than 10%.

In conclusion, MAD-GAN proved to be the best architecture in the anomaly detection step. It managed to clean the dataset with the smallest anomaly threshold, preserving the most normal profiles in the dataset. The resulting datasets from this step will be used in the prediction phase, described in the next section.

Note that the MAD-GAN experiments were only made possible because of the use of PCA on the data. This might indicate that the previous preliminary experiments might also have been capable of performing well if this method had been used.

6.3 ML Experiments

After selecting an appropriate method for anomaly detection, the impact of the chosen approach was evaluated with the MULTI-VP simulation. As previously stated, the set of validation profiles excluded from the training phase was used for this step. Due to time constraints, only the clustering models from Section 5.5.2 were trained without the detected anomalous profiles.

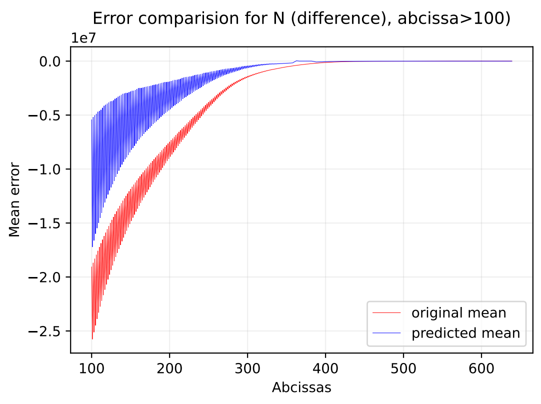
The detection was done with the input and output MAD-GAN models from the previous section. The results from detecting the input and output variables were aggregated into a single file indicating the name of every anomalous profile.

First, a simple experiment was done to assert if excluding the anomalous profiles from the baseline model results would translate to decreased mean errors. Surprisingly this only occurred in the $n[cm^3]$ variable, while the others mainly stayed the same as the ones without anomaly removal. The same approach was tried on the clustering models obtained from the previous experiments (refer to Section 5.5.2). This showed a slight improvement in the mean error in each variable compared to the previous results.

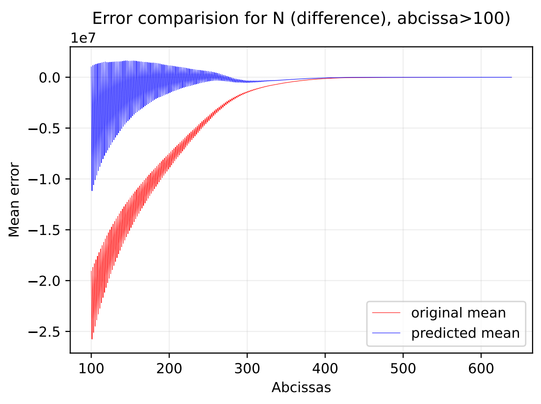
In the second batch of experiments, we intended to discover if anomalies in the training data were hindering the prediction quality. The abovementioned anomalous profiles were excluded from the training dataset and used to train new iterations of the clustering models obtained in the previous experiments. Following the same methodology as before, the predictions of the validation dataset of these new models were fed to the MULTI-VP simulation as initial flow estimates. The anomalous files in the validation dataset were also excluded from the evaluation metrics, which means that the predictions of the new models on anomalous are being ignored in the evaluation.

Figure 6.8a shows the results from the original clustering models on the $n[cm^3]$ variable without the anomalous profiles detected by MAD-GAN. Surprisingly, the results of this experiment are substantially worse than the ones obtained in the clustering experiments without removing anomalous profiles. This can be because the detection method identified anomalies in other variables that did not constitute anomalies in $n[cm^3]$, which might have skewed the mean of the errors.

On the other hand, the predictions obtained with the new clustering model trained on the clean dataset generate better predictions than in both cases. This improvement is evident in Figure 6.8b where the mean error of v is more concentrated around zero. A smaller mean error indicates that



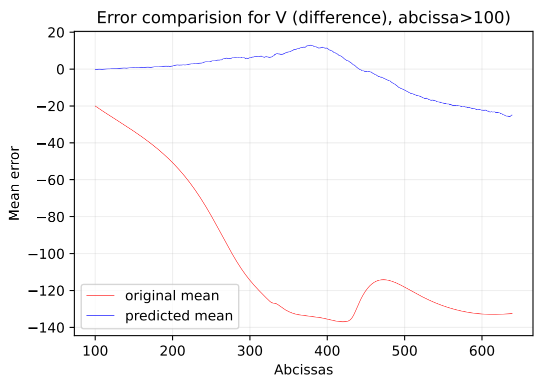
(a) Clustering GAN selected



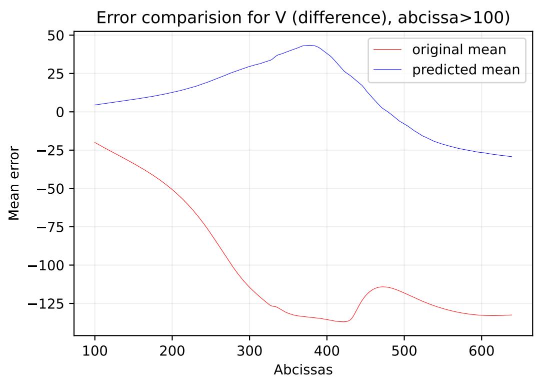
(b) Clustering Clean Dataset

Figure 6.8: Error comparison of $n[cm^3]$. *(a)* shows the mean error comparison of the results from the previous clustering experiments (without anomalous profiles) and the expert estimates; *(b)* is the mean error comparison of the clustering models trained on datasets without anomalies and the original expert estimates.

the predicted values are closer to the actual values, indicating a higher level of accuracy in the forecasting process.



(a) Clustering GAN selected

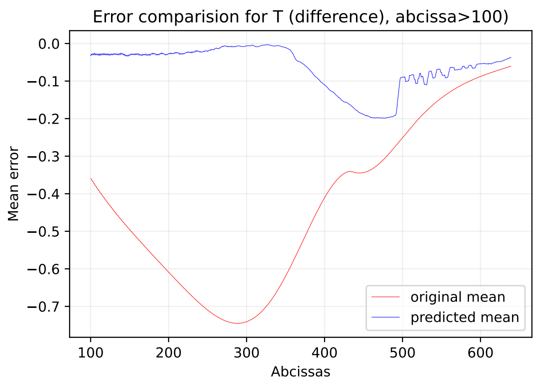


(b) Clustering Clean Dataset

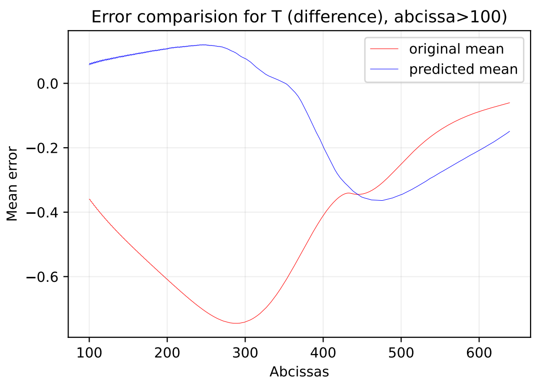
Figure 6.9: Abscissa wise estimate error comparison of $v[km/s]$. *(a)* shows the mean error comparison of the results from the previous clustering experiments (without anomalous profiles) and the expert estimates; (b) compares the results of the clustering models when trained on the clean dataset with the expert estimates.

Figure 6.9a demonstrates a slight reduction in error when compared to the clustering model that did not involve anomaly removal. However, the results obtained from the new clustering models trained on the clean dataset (depicted in Figure 6.9b) show worse performance than the previous models. This discrepancy is particularly noticeable in the range between abscissa values 300 and 400, where the error in the new models is considerably higher than that of the previous approach. Despite this observation, it is essential to note that the new clustering models outperformed the baseline model and the original expert estimates regarding predictive accuracy.

The analysis of temperature data reveals significant differences compared to the results obtained from the initial clustering models (Figure 6.10a). Figure 6.10b presents the outcomes of the new model trained on the clean dataset, showcasing a higher mean error than the first clustering model. This discrepancy is particularly pronounced from abscissa value 300 onwards. The higher mean error in this range indicates that the new model failed to effectively learn the underlying features and patterns of the temperature lines. As a result, its estimates for temperature values in this specific range are even worse than the estimates provided by initial experts. This finding suggests there may be specific characteristics or complexities in the temperature data that the new model could not capture effectively. It is possible that the clean dataset used for training lacked crucial information or representative samples in the range where the model performed poorly.



(a) Cluster models (no anomaly training)



(b) Baseline (no anomalies)

Figure 6.10: Abscissa wise error comparison of $T[MK]$. (a) shows the mean error comparison of the results from the previous clustering experiments (without anomalous profiles) and the expert estimates; (b) is the mean error comparison of the clustering models trained on datasets without anomalies and the original expert estimates.

The findings suggest that training the clustering model on a clean dataset leads to worse predictions, as evidenced by the increased mean error in the velocity and temperature variables. This approach only seemed to work for the $n[cm^3]$ variable, with it having estimates closer to the ones the simulation outputs.

6.4 Summary

In this chapter, the use of adversarial learning was explored for the detection of anomalies in solar wind profiles. Section 6.1 provides background knowledge on generative adversarial networks. Section 6.2 details the various adversarial methods used to filter anomalies from the dataset, with the most efficient one being MAD-GAN.

Finally, the last section shows the results of the MULTI-VP simulation when using the new approach. In it, we show that removing anomalous profiles from the previous results would produce better mean errors for the estimates, which indicates that broken files harm the overall results.

Surprisingly, the clustering models trained without anomalous profiles failed to produce better estimates for the V and T variables, with the ones from the clustering experiments being closer to the final simulation results.

Chapter 7

Final Remarks

This chapter provides an overview of the work carried out throughout this thesis and analyses the obtained results. Section 7.1 evaluates the hypothesis and answers the research questions of this thesis. Section 7.2 provides a broad analysis of the work and results of this thesis, while Section 7.3 details some possible paths for further research.

7.1 Hypothesis Evaluation

Following the work carried out in this thesis, this section intends to evaluate the proposed hypothesis from Section 4.4, which is the following:

> By integrating clustering and adversarial anomaly detection techniques, the initial conditions predicted by RNNs for the MULTI-VP simulator will be closer to the final simulation results and contribute to faster executions.

An analysis of the results from the clustering shows that we have achieved the first part of the hypothesis, as the estimates produced by the new approach were significantly closer to the simulation outputs. However, both experiments seem to indicate that the computation time of the simulation has no direct correlation to the quality of the initial conditions. This is evidenced by the very similar mean speedup of all tested approaches.

The following answers to the research questions help explain these observations:

RQ1 *Are clustering methods capable of detecting characteristics in the dataset that were overlooked by the original RNN and would help with the prediction task?* The clustering experiments' results show that we could generate estimates closer to the simulation outputs using this approach. The models trained on the resulting KMeans clustering of the PCA of the input variables managed to produce better estimates than the single model trained on the whole dataset. This shows that by dividing the dataset into clusters of approximately the same size and training an RNN for each cluster, we could capture previously ignored features of the baseline prediction model.

RQ2 *Do the estimates obtained with clustering-based training significantly improve the simulation's performance?* Despite being able to generate better initial estimates than the ones provided by the experts and the baseline model, this approach didn't significantly reduce the computation time of the MULTI-VP simulation. The impact on the computation time was assessed by comparing the mean speedup of simulations obtained by feeding MULTI-VP simulation with predictions from both the baseline and the devised clustering models. The baseline model achieved a mean speedup of 1.06, which constituted a slight improvement over the expert estimates. On the other hand, the new approach with the clustering models resulted in a mean speedup of only 1.05, which means that there was a marginal increase in the computation time of the simulation when compared to the baseline approach.

RQ3 *Can adversarial learning methods detect anomalies in solar wind profiles?* The results from section 6.2 show that it is possible to use adversarial detection methods in this type of data. Even though the linear GAN and the AAE managed to clean the dataset, the number of detected normal profiles (FN) is still very high compared to MAD-GAN, which might indicate that the previous architectures are not very suited for this task. In addition, we showed that grouping consecutive profiles into windows and then using these on LSMT-based GAN architectures (as MAD-GAN) proved very effective, surpassing the other approaches.

RQ4 *Does the resulting dataset significantly improve the predictive ability of the RNN?* Despite removing most anomalous data from the training dataset, the estimates' quality decreased compared to the previous clustering models. The results of the adversarial experiments showed that the predictions from the clustering models trained on the clean dataset were significantly worse than the earlier approaches (in some cases, worse than the expert estimates). This might indicate that the removed anomalous profiles provided critical features for the RNN training, and excluding them from this phase hindered the predictions' quality. However, it is still important to note that we are using a small portion of the dataset randomly selected by hand, which might not represent the entire dataset.

RQ5 *Does the improved predictive ability of the RNN result in a further reduction of execution time for MULTI-VP?* Even with worse estimates than the previous clustering models, the new approach obtained a better speedup (from 1.05 to 1.06). This disproves the central hypothesis of this thesis that initial partial flow estimates closer to the final simulation ones would reduce the computation time. During the experiments, it was noticed that the model was still predicting extreme values when given anomalous inputs, even without seeing anomalies in the training phase. Further work is needed in this step to be able to reach a better conclusion. A possible approach would be to exclude the anomalous profiles from the simulation to ensure that these are the cause for the increased computation times.

7.2 Conclusions

The need to consistently predict the Sun's conditions that lead to extreme events has become an increasingly important study. However, technological difficulties make obtaining real-time data from the Sun's surface challenging. Multiple numeric simulators have tried to fill this gap by extrapolating these conditions based on limited observations from Earth. In this dissertation, we have explained the problems (Section 1.2) associated with these solutions that severely affect the ability to generate solar estimations promptly. These issues included the long execution time of the simulation models as well as the need for initial expert estimations. Additionally, it was posited that the use of machine learning techniques to predict the original conditions suffered greatly from the data dispersion and anomalies in the training data.

Several experiments were carried out with the existing dataset to address the generalization issues of the baseline RNN models, with widely used clustering algorithms and two data dimensionality reduction methods. These were devised after an extensive analysis of the state-of-the-art approaches for clustering with an emphasis on improving machine learning performance. In the end, a new approach for generating initial flow conditions closer to the final simulation conditions was developed. However, contrary to what was hypothesized, evidence shows that the improved estimates failed to reduce the overall computation time of the simulation.

In the second part of this thesis, many experiments were carried out to determine if the abnormal profiles in the dataset were hampering the predictions' quality. After an extensive analysis of the state-of-the-art approaches for adversarial anomaly detection, three methods were implemented (one of which was an adaptation) and tested on the training dataset, which proved that these methods could detect faulty profiles in the given dataset. From the experiments, we concluded that the state-of-the-art MAD-GAN was the most efficient method for the task. Due to this, it was used to identify anomalies in the input and output variables of MULTI-VP. In this phase, we also show that training the same clustering methods from the previous experiments without anomalies failed to produce more approximate initial and final estimates, but that despite this, the mean speedup obtained in the simulation was superior.

In a final experiment, the outputs of previous MULTI-VP execution were used directly as initial conditions of the simulation. The preliminary findings suggest that there were no notable improvements in the computation time of the simulation. These findings support the notion that initial estimates' proximity to the simulation outputs might not contribute to faster executions. One hypothesis to explain this observation is that the simulation itself has inherent limitations or overhead that prevent faster executions, regardless of the quality of the initial estimates. However, it is important to note that further research is required to draw a more definitive conclusion on this matter.

One issue with the methodology that might have contributed to these results was that the data used in the MULTI-VP simulation only constituted 10% of the entire dataset. We used such a small part of the dataset to evaluate the performance of the approaches with data that was never seen in the training and detection phases. The slow computation time of MULTI-VP was also a key factor for this, as it takes up to two weeks to produce the estimates for this dataset, and we needed to carry out multiple tests during this thesis. The main issue is that the randomly chosen profiles for the validation dataset might not represent the whole dataset, bringing some uncertainty to the results.

From these experiments, it was concluded that we achieved part of the goals defined in this thesis, as we produced estimates significantly closer to the simulation outputs. Despite these improvements, we were unable to reduce the overall computation time, leading us to believe that initial flow conditions closer to the final solutions might not necessarily be linked to the performance of the simulation.

7.3 Future Work

Considering all the work in this dissertation, we concluded that despite having closer initial conditions to the final solution, we could not significantly improve the simulation's computation time. More research on this area is needed in order to reach a possible explanation for these results. This would require a more in-depth analysis of the inner workings of the MULTI-VP simulation to determine why significantly closer initial and final conditions do not lead to computational improvements.

Additional research needs to be conducted to assess the physical coherence of the initial conditions generated by the new prediction models. This can involve examining the conservation of mass, momentum, and energy across each individual solar wind profile. By evaluating the physical feasibility of these predictions, it may be possible to develop a surrogate model that could serve as an early-stage solar wind forecasting system.

In some applications where a higher level of scrutiny is not required, this surrogate model could potentially replace the need for the more resource-intensive MULTI-VP simulation. However, it is important to emphasize that further investigation and analysis are necessary to validate the accuracy and reliability of the surrogate model and its ability to provide physically coherent predictions.

Other approaches, such as applying physics-informed machine learning, could be developed to achieve physically sound initial flow conditions that would be able to replace the MULTI-VP simulation. Furthermore, the developed methodologies could be tested on other MHD simulators to determine if the proximity of initial and final conditions leads to significant computation times that MULTI-VP did not achieve.

References

[1] Yiwei Cheng, Haiping Zhu, Jun Wu, and Xinyu Shao. Machine Health Monitoring Using Adaptive Kernel Spectral Clustering and Deep Long Short-Term Memory Recurrent Neural Networks. *IEEE Transactions on Industrial Informatics*, 15(2):987–997, 2019.

[2] Rui F. Pinto and Alexis P. Rouillard. A Multiple Flux-tube Solar Wind Model. *The Astrophysical Journal*, 838(2):89, 2017.

[3] D. Odstrčil and V. J. Pizzo. Three-dimensional propagation of coronal mass ejections (CMEs) in a structured solar wind flow: 1. CME launched within the streamer belt. *Journal of Geophysical Research: Space Physics*, 104(A1):483–492, 1999.

[4] Ana Filipa Sousa Barros. Initial Condition Estimation in Flux Tube Simulations using Machine Learning, 2021.

[5] Rainer Schwenn. Space Weather: The Solar Perspective. *Living Reviews in Solar Physics*, 3(1):2, 2006.

[6] Rushil Anirudh, Rick Archibald, M. Salman Asif, Markus M. Becker, Sadruddin Benkadda, Peer-Timo Bremer, Rick H. S. Budé, C. S. Chang, Lei Chen, R. M. Churchill, Jonathan Citrin, Jim A. Gaffney, Ana Gainaru, Walter Gekelman, Tom Gibbs, Satoshi Hamaguchi, Christian Hill, Kelli Humbird, Sören Jalas, Satoru Kawaguchi, Gon-Ho Kim, Manuel Kirchen, Scott Klasky, John L. Kline, Karl Krushelnick, Bogdan Kustowski, Giovanni Lapenta, Wenting Li, Tammy Ma, Nigel J. Mason, Ali Mesbah, Craig Michoski, Todd Munson, Izumi Murakami, Habib N. Najm, K. Erik J. Olofsson, Seolhye Park, J. Luc Peterson, Michael Probst, Dave Pugmire, Brian Sammuli, Kapil Sawlani, Alexander Scheinker, David P. Schissel, Rob J. Shalloo, Jun Shinagawa, Jaegu Seong, Brian K. Spears, Jonathan Tennyson, Jayaraman Thiagarajan, Catalin M. Ticoş, Jan Trieschmann, Jan van Dijk, Brian Van Essen, Peter Ventzek, Haimin Wang, Jason T. L. Wang, Zhehui Wang, Kristian Wende, Xueqiao Xu, Hiroshi Yamada, Tatsuya Yokoyama, and Xinhua Zhang. 2022 Review of Data-Driven Plasma Science, May 2022.

[7] D.N. Baker. What is space weather? *Advances in Space Research*, 22(1):7–16, 1998.

[8] Mark Moldwin. *An Introduction to Space Weather*. Cambridge University Press, 2008.

[9] Eric Priest. *The Solar Wind*. Cambridge University Press, 2014.

[10] Sami K. Solanki, Bernd Inhester, and Manfred Schüssler. The solar magnetic field. *Reports on Progress in Physics*, 69(3):563–668, March 2006. arXiv:1008.0771 [astro-ph].

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Number: 7553 Publisher: Nature Publishing Group.

[12] Robin M. Schmidt. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview, 2019.

[13] Charu C. Aggarwal. *Outlier Analysis*. Springer New York, 2013.

[14] Xuan Xia, Xizhou Pan, Nan Li, Xing He, Lin Ma, Xiaoguang Zhang, and Ning Ding. GAN-based anomaly detection: A review. *Neurocomputing*, 493:497–535, July 2022.

[15] Priyanga Dilini Talagala, Rob J Hyndman, and Kate Smith-Miles. Anomaly Detection in High Dimensional Data, 2019.

[16] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies, oct 2004.

[17] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[18] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[19] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras. A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279, 2014.

[20] Gang Kou, Yi Peng, and Guoxun Wang. Evaluation of clustering algorithms for financial risk analysis using mcdm methods. *Information Sciences*, 275:1 – 12, 2014. Cited by: 630.

[21] Feiping Nie, Xiaoqian Wang, and Heng Huang. Clustering and projected clustering with adaptive neighbors. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 977–986. ACM, 2014.

[22] Daniel Granato, Jânio S. Santos, Graziela B. Escher, Bruno L. Ferreira, and Rubén M. Maggio. Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective. *Trends in Food Science & Technology*, 72:83–90, 2018.

[23] Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. Clustering-based undersampling in class-imbalanced data. *Information Sciences*, 409–410:17–26, 2017.

[24] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences*, 465:1–20, 2018.

[25] John Paparrizos and Luis Gravano. K-Shape: Efficient and Accurate Clustering of Time Series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1855–1870. ACM, 2015.

[26] Mingjing Du, Shifei Ding, and Hongjie Jia. Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowledge-Based Systems*, 99:135–145, 2016.

[27] Amita Malav, Kalyani Kadam, and Pooja Kamat. PREDICTION OF HEART DISEASE USING K-MEANS and ARTIFICIAL NEURAL NETWORK as HYBRID APPROACH to IMPROVE ACCURACY. *International Journal of Engineering and Technology*, 9(4):3081–3085, 2017.

[28] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. ClusterNet: Deep Hierarchical Cluster Network With Rigorously Rotation-Invariant Representation for Point Cloud Analysis. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4989–4997. IEEE, 2019.

[29] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised Deep Embedding for Clustering Analysis, 2016.

[30] Fateme Fahiman, Sarah M. Erfani, Sutharshan Rajasegarar, Marimuthu Palaniswami, and Christopher Leckie. Improving load forecasting based on deep learning and K-shape clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4134–4141, 2017.

[31] Tanmay Kumar Behera and Suvasini Panigrahi. Credit Card Fraud Detection: A Hybrid Approach Using Fuzzy Clustering & Neural Network. In *2015 Second International Conference on Advances in Computing and Communication Engineering*, pages 494–499, 2015.

[32] Jinjun Tang, Fang Liu, Yajie Zou, Weibin Zhang, and Yinhai Wang. An Improved Fuzzy Neural Network for Traffic Speed Prediction Considering Periodic Characteristic. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2340–2350, 2017.

[33] Hamidreza Jahangir, Hanif Tayarani, Saleh Sadeghi Gougheri, Masoud Aliakbar Golkar, Ali Ahmadian, and Ali Elkamel. Deep Learning-Based Forecasting Approach in Smart Grids With Microclustering and Bidirectional LSTM Network. *IEEE Transactions on Industrial Electronics*, 68(9):8298–8309, 2021.

[34] Chenshuang Zhang, Guijin Wang, Jingwei Zhao, Pengfei Gao, Jianping Lin, and Huazhong Yang. Patient-specific ECG classification based on recurrent neural networks and clustering technique. In *2017 13th IASTED International Conference on Biomedical Engineering (BioMed)*, pages 63–67, 2017.

[35] Yandong Yang, Wei Li, T. Aaron Gulliver, and Shufang Li. Bayesian Deep Learning-Based Probabilistic Load Forecasting in Smart Grids. *IEEE Transactions on Industrial Informatics*, 16(7):4703–4713, 2020.

[36] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11730 LNCS:703–716, 2019.

[37] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient GAN-Based Anomaly Detection, 2018.

[38] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. Adversarially Learned Anomaly Detection. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 727–736, 2018.

[39] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M.A. Zuluaga. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3395–3404, 2020.

[40] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He. Generative Adversarial Active Learning for Unsupervised Outlier Detection. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1517–1528, 2020.

[41] Shuokang Huang and Kai Lei. IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. *Ad Hoc Networks*, 105:102177, 2020.

[42] W. Jiang, Y. Hong, B. Zhou, X. He, and C. Cheng. A GAN-Based Anomaly Detection Approach for Imbalanced Industrial Time Series. *IEEE Access*, 7:143608–143619, 2019.

[43] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial Feature Learning, 2017.

[44] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 33–43, 2020.

[45] X. Wang, Y. Du, S. Lin, P. Cui, Y. Shen, and Y. Yang. adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection. *Knowledge-Based Systems*, 190, 2020.

[46] Andrew Blance, Michael Spannowsky, and Philip Waite. Adversarially-trained autoencoders for robust unsupervised new physics searches. *Journal of High Energy Physics*, 2019(10):47, 2019.

[47] J. Wu, Z. Zhao, C. Sun, R. Yan, and X. Chen. Fault-Attention Generative Probabilistic Adversarial Autoencoder for Machine Anomaly Detection. *IEEE Transactions on Industrial Informatics*, 16(12):7479–7488, 2020.

[48] Phuc Cuong Ngo, Amadeus Aristo Winarto, Connie Khor Li Kou, Sojeong Park, Farhan Akram, and Hwee Kuan Lee. Fence GAN: Towards Better Anomaly Detection. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 141–148, 2019.

[49] Ilias Siniosoglou, Panagiotis Radoglou-Grammatikis, Georgios Efstathopoulos, Panagiotis Fouliras, and Panagiotis Sarigiannidis. A Unified Deep Learning Anomaly Detection and Classification Approach for Smart Grid Environments. *IEEE Transactions on Network and Service Management*, 18(2):1137–1151, 2021.

[50] Shaowei Liu, Hongkai Jiang, Zhenghong Wu, and Xingqiu Li. Data synthesis using deep feature enhanced generative adversarial networks for rolling bearing imbalanced fault diagnosis. *Mechanical Systems and Signal Processing*, 163:108139, 2022.

[51] S.K. Lim, Y. Loo, N.-T. Tran, N.-M. Cheung, G. Roig, and Y. Elovici. DOPING: Generative Data Augmentation for Unsupervised Anomaly Detection with GAN. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, volume 2018-November, pages 1122–1127, 2018.

[52] Chuanlong Yin, Yuefei Zhu, Shengli Liu, Jinlong Fei, and Hetong Zhang. An enhancing framework for botnet detection using generative adversarial networks. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 228–234, 2018.

[53] Yuxuan Yuan, Kaveh Dehghanpour, Fankun Bu, and Zhaoyu Wang. Outage Detection in Partially Observable Distribution Systems Using Smart Meters and Generative Adversarial Networks. *IEEE Transactions on Smart Grid*, 11(6):5418–5430, 2020.

[54] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, 2014.

[55] Fanhui Kong, Jianqiang Li, Bin Jiang, Huihui Wang, and Houbing Song. Integrated Generative Model for Industrial Anomaly Detection via Bidirectional LSTM and Attention Mechanism. *IEEE Transactions on Industrial Informatics*, 19(1):541–550, 2023.

[56] Md Abul Bashar and Richi Nayak. TAnoGAN: Time Series Anomaly Detection with Generative Adversarial Networks. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1778–1785, 2020.

[57] Chu Wang, Yan-Ming Zhang, and Cheng-Lin Liu. Anomaly Detection via Minimum Likelihood Generative Adversarial Networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1121–1126, 2018.

[58] Qinghua Xu, Shaukat Ali, and Tao Yue. Digital Twin-based Anomaly Detection in Cyber-physical Systems. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 205–216, 2021.

[59] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[60] Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding, and Chin-Teng Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017.

[61] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[62] 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.

[63] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[64] Giuseppe Vettigli. Minisom: minimalistic and numpy-based implementation of the self organizing map, 2018.

[65] François Chollet et al. Keras. `https://keras.io`, 2015.

[66] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2016.

[67] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 214–223. JMLR.org, 2017.

[68] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans): Challenges, solutions, and future directions. *ACM Comput. Surv.*, 54(3), may 2021.

[69] Claire Little, Mark Elliot, Richard Allmendinger, and Sahel Shariati Samani. Generative Adversarial Networks for Synthetic Data Generation: A Comparative Study, 2021.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015.

Appendix A

Clustering - KMeans Results

Results of clustering method that was used to train the initial wind flow prediction RNN. PCA and TSNE were applied to the $R[Rsun]$, $B[G]$ and $\alpha[deg]$ input variables of MULTI-VP. Next, a KMeans clustering was performed on the resulting representations. Only the PCA solution was used, as the KMeans applied to the TSNE of the joint inputs (Figure A.1) failed to produce a clear data division.

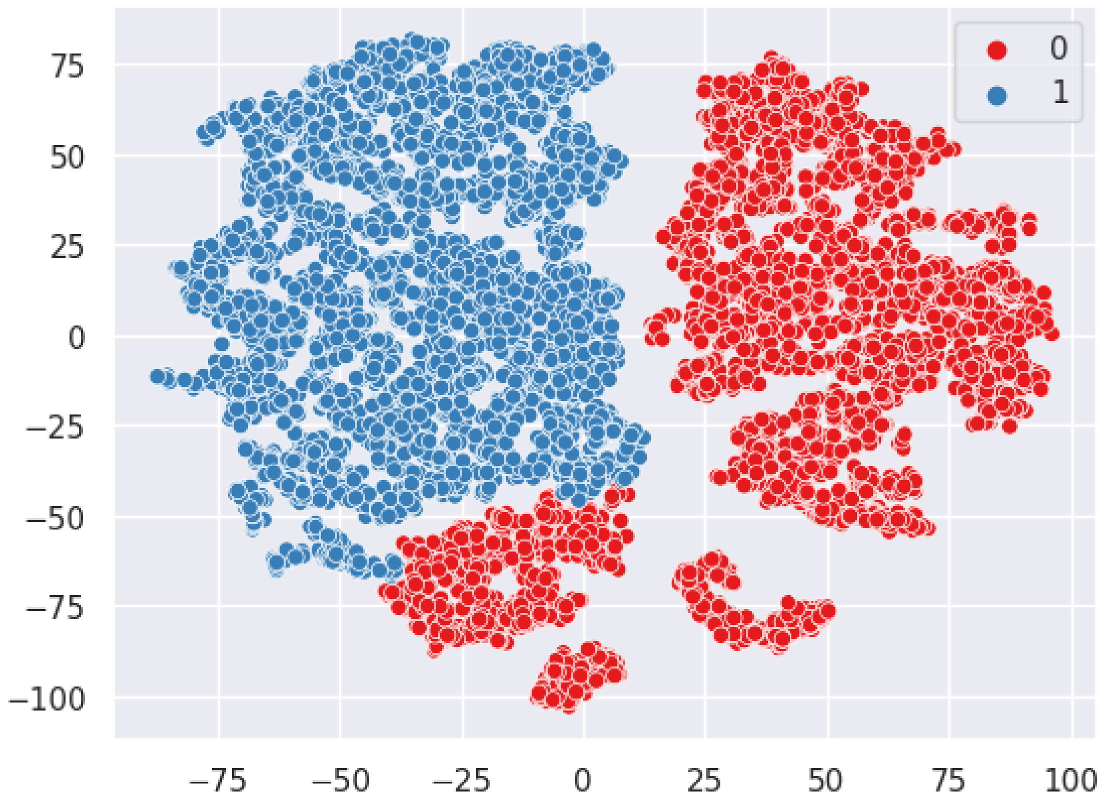


Figure A.1: Visualization of the clusters obtained with the TSNE of the joint inputs. Unlike the results from the PCA on the joint inputs (Figure A.3), this method didn't produce a good division of the data as points that would be more suited in cluster 1 were assigned to cluster 0.

A.1 PCA Results

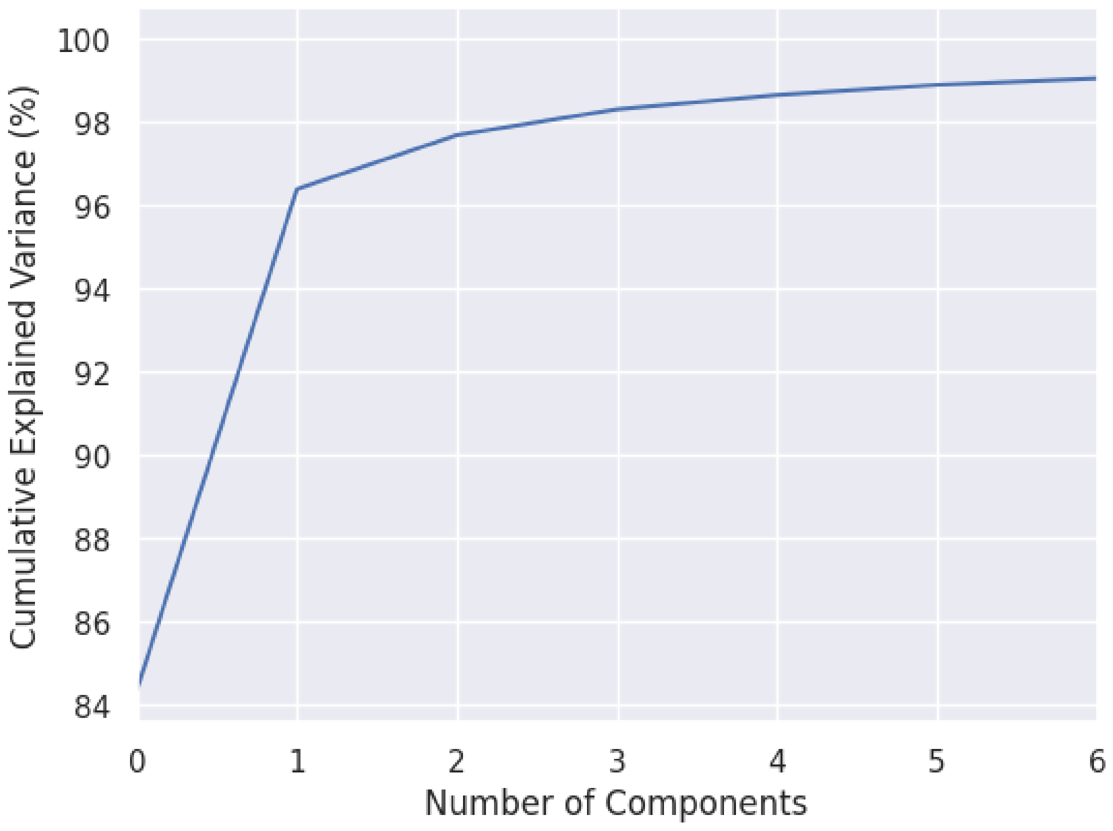


Figure A.2: Cumulative Explained Variance for the PCA of the input variables

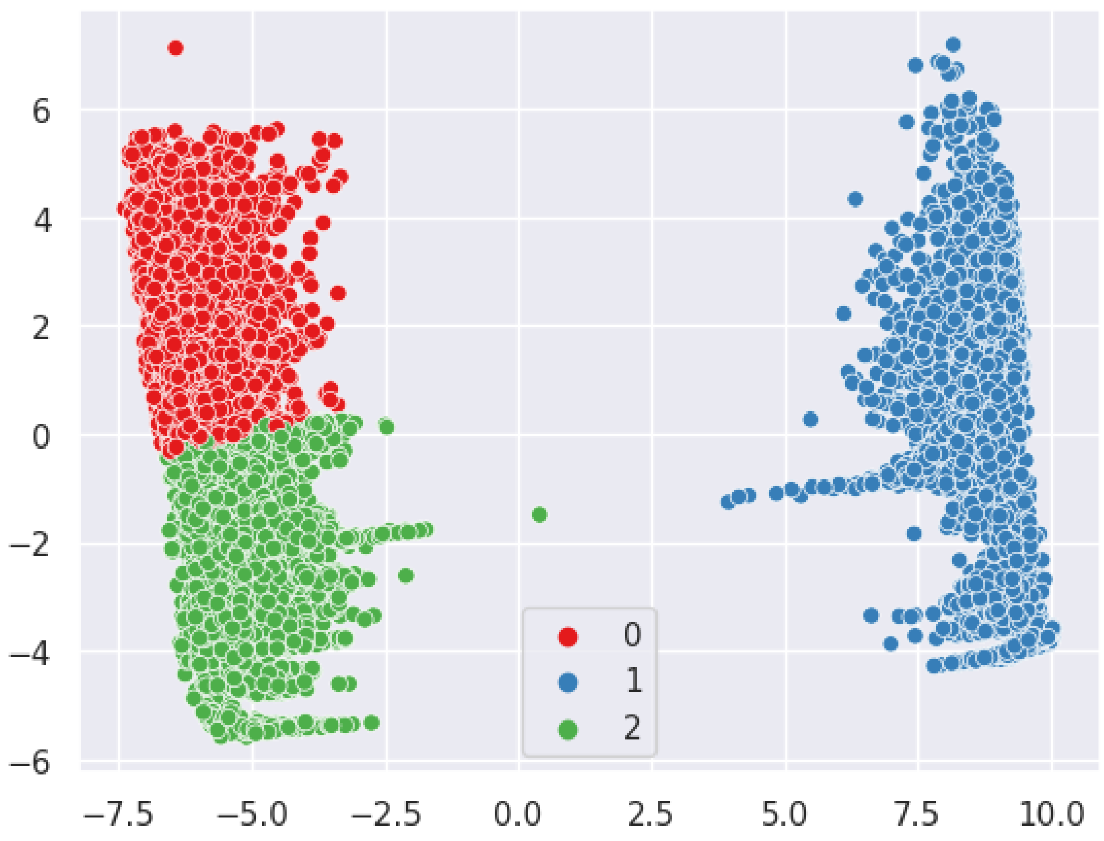


Figure A.3: KMeans clustering of the PCA of the input variables

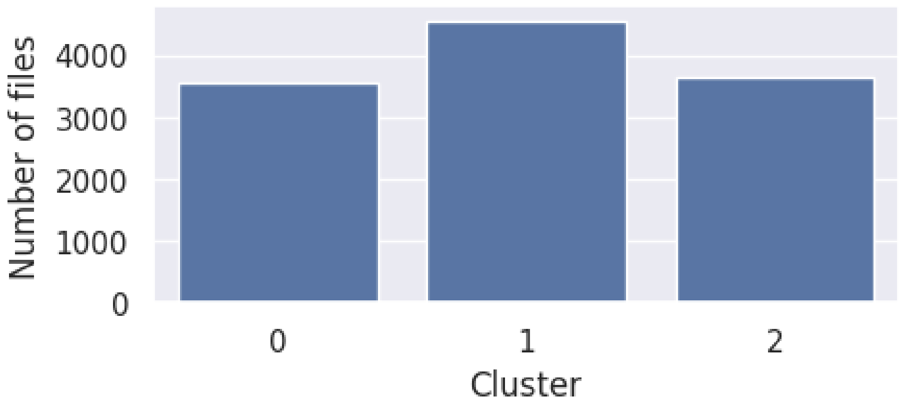


Figure A.4: Number of profiles per cluster

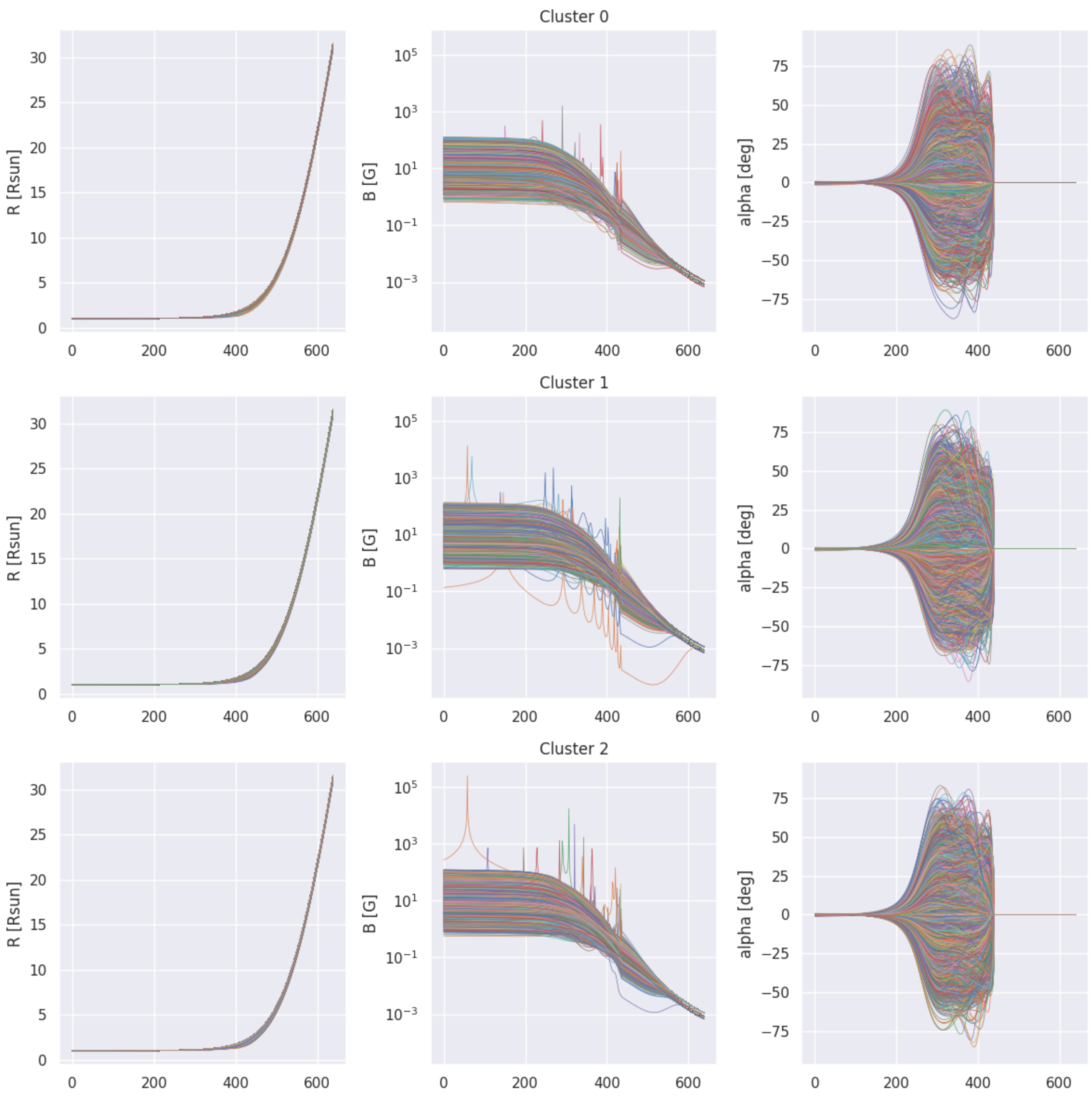


Figure A.5: Data division based on the clustering results

Appendix B

MAD-GAN Results

Results of the MAD-GAN approach for anomaly detection in solar wind profiles. Two models were created, one for detecting input variables and the other for the outputs. Each model produces an anomaly score for every file in the dataset and a defined percentage is considered anomalous. In these experiments the threshold is set at 3%.

B.1 Input Model

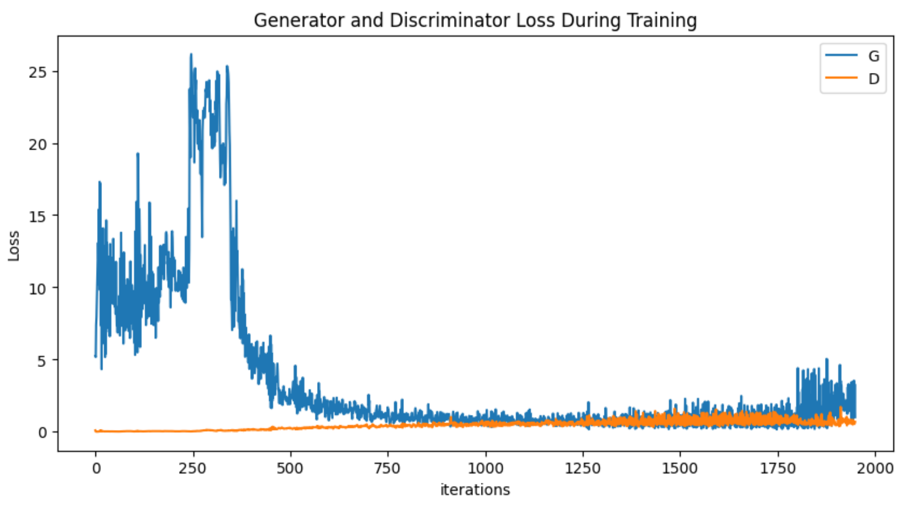


Figure B.1: MAD-GAN input model training history.

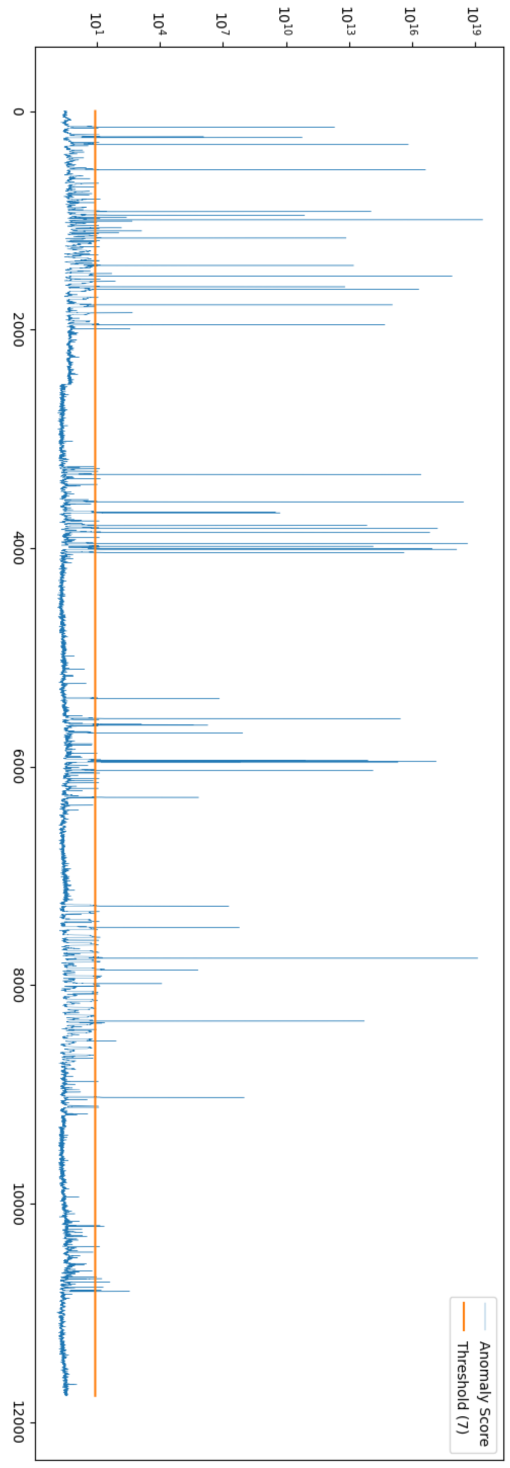


Figure B.2: Input Anomaly Scores. MAD-GAN Anomaly Reconstruction Scores for each profile in the input dataset. Profiles with scores above the orange line are considered anomalies.

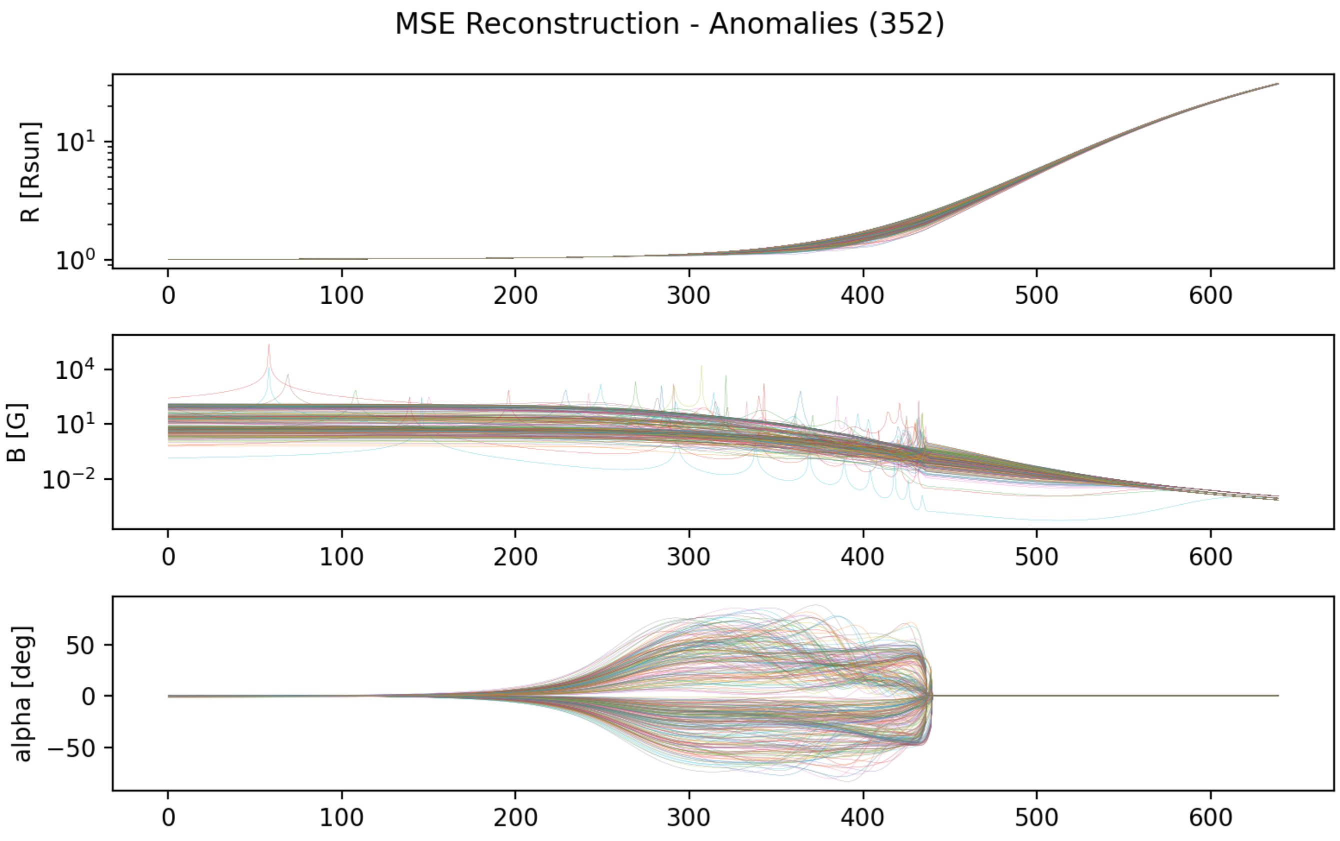


Figure B.3: Anomalies detected with the MAD-GAN input model.

B.2 Output Model

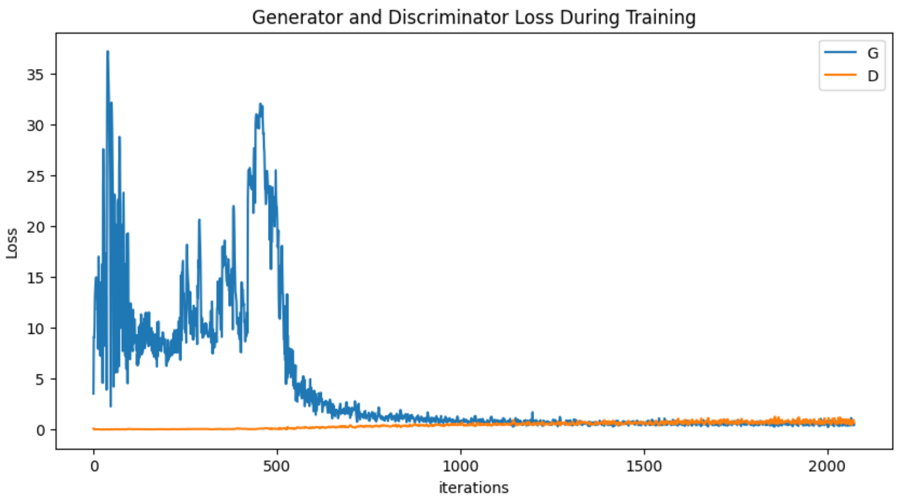


Figure B.4: MAD-GAN output model training history.

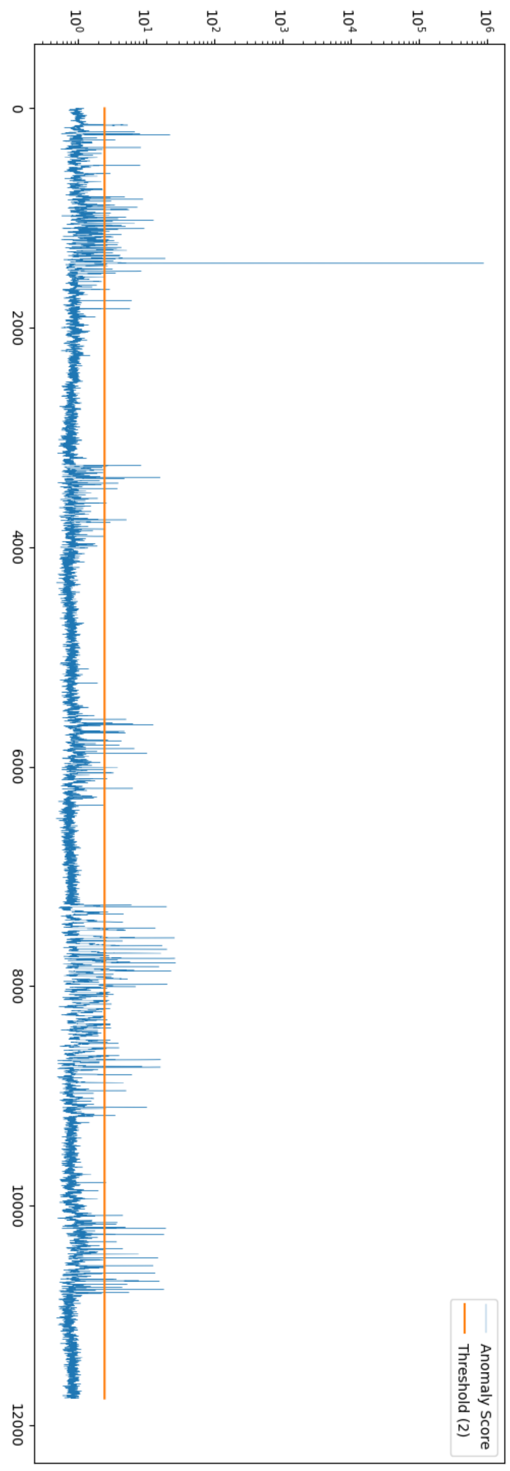


Figure B.5: Output Anomaly Scores. MAD-GAN Anomaly Reconstruction Scores for each profile in the output dataset. Profiles with scores above the orange line are considered anomalies.

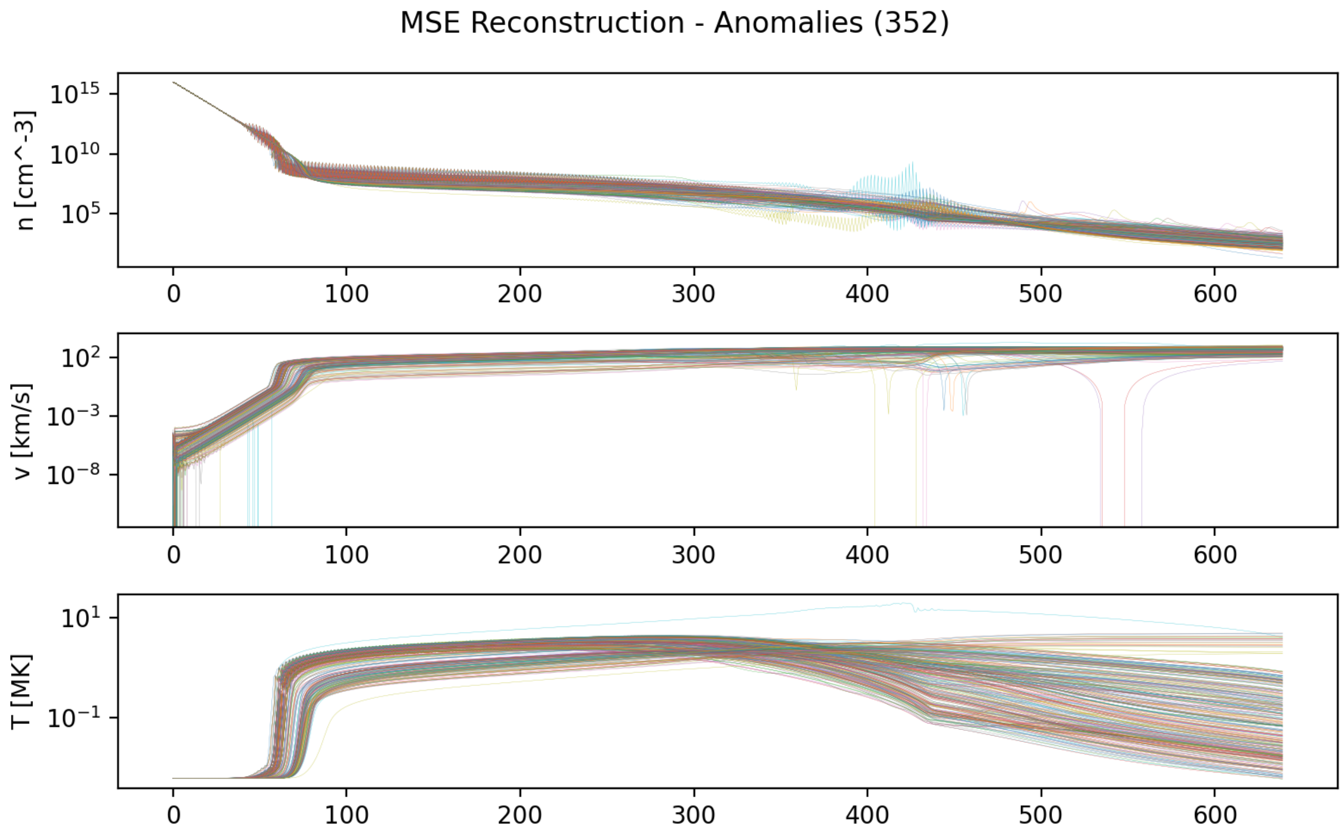


Figure B.6: Anomalies detected with the MAD-GAN output model.