



Università di Genova

DIBRIS - Department of Informatics, Bioengineering,
Robotics and Systems Engineering

Unsupervised Rule-Based Out-of-Distribution Detection

by

Selam Gebrehiwot Tewolde

Supervisors:

Dr. Maurizio Mongelli

Prof. Luca Oneto

Co-Supervisor:

Dr. Sara Narteni

In partial fulfillment of the requirements for the degree of

Laurea Magistrale in Computer Engineering

March 24, 2025

I dedicate this thesis to my family for their endless encouragement and patience, and to my friends who have become family along the way. Their constant support made this journey possible.

Declaration of Originality

I, Selam Gebrehiwot Tewolde, hereby declare that this thesis is my own work and all sources of information and ideas have been acknowledged appropriately. This work has not been submitted for any other degree or academic qualification. I understand that any act of plagiarism, reproduction, or use of the whole or any part of this thesis without proper acknowledgment may result in severe academic penalties.

Acknowledgements

I would like to express my deepest gratitude to Dr. Maurizio Mongelli and Dr. Sara Narteni for their unwavering support, encouragement, and guidance throughout this thesis. Their generosity in sharing their expertise, their patience, and their constant availability have been invaluable. They have not only provided me with the freedom to explore and shape this work independently but have also been there whenever I needed help, offering insightful advice and constructive feedback. This thesis would not have been possible without their trust and support.

I am also sincerely thankful to my supervisor, Professor Luca Oneto, for his prompt responses, constructive feedback, and continuous availability. His guidance has been instrumental in refining and strengthening this thesis, always pushing me to improve and think critically.

Furthermore, I am deeply grateful to my friends for their encouragement and for providing both motivation and moments of respite when they were most needed. The discussions and support from my peers at the university have enriched this journey, making it not only a challenging endeavor but also a rewarding and fulfilling experience.

Abstract

Detecting anomalies and out-of-distribution (OoD) data in machine learning remains a critical challenge, particularly in real-world applications where anomalous data often lacks labels and can have severe consequences. This thesis develops an unsupervised framework to address this problem, providing a scalable solution for detecting anomalies in dynamic and complex domains like autonomous systems and healthcare. By perturbing image features, the model generates synthetic data that simulates realistic anomalies without relying on costly labeled data. This allows the system to identify deviations from the patterns seen in the in-distribution data. At the same time, histogram-based analysis monitors the effectiveness of the rules, enabling real-time detection of out-of-distribution (OoD) data. The findings demonstrate that the method is more effective at detecting unknown anomalies in dynamic and complex environments. This work contributes to the advancement of anomaly detection techniques and presents a practical solution for industries facing limited labeled data, with significant implications for improving the safety and reliability of machine learning systems in critical applications.

Keywords: Out-of-distribution, OoD, Out of distribution detection, ODD, synthetic image generation, rule based methods

Notation

In this chapter, a table will be presented which contains the notations to be used throughout this agreement. Technical term abbreviations will be explained upon first use:

| | |
|--------------------|-------------------------------------|
| OoD | Out-of-Distribution |
| ID | In-Distribution |
| ODD | OoD detection |
| ML | Machine learning |
| XAI | eXplainable Artificial Intelligence |
| LLM | Logical Learning Machine |
| TR | Training set |
| OP | Operational set |
| tr_i | i -th training subset |
| op_i | i -th operational subset |
| n_s | number of data samples in a split |
| N_{tr} | Number of training splits |
| N_{op} | Number of operational splits |
| \mathcal{R}_{tr} | Training reference ruleset |
| r_i | i -th rule |
| h_i^j | j -th hit for the i -th rule |
| l_p | l_p norm |
| FN | False Negatives |
| FP | False Positives |
| H&R | Hazard&Robots dataset |
| Δ_i | Variance Percentage |
| R_c | Correlation Matrix |
| MI | Mutual Information |
| SSN | Switching Neural Network |
| HOG | Histogram of Oriented Gradients |
| GLCM | Gray-Level Co-occurrence Matrix |

Contents

| | |
|---|-----------|
| Notations | v |
| 1 Introduction | 1 |
| 1.1 Background and Motivations | 1 |
| 1.2 Contributions | 3 |
| 1.3 Thesis Structure | 4 |
| 2 Related Works | 5 |
| 2.1 Introduction | 5 |
| 2.2 Literature Reviews | 5 |
| 2.2.1 Out-of-distribution Detection | 5 |
| 2.2.2 Data Generation Methods | 6 |
| 3 Methodology | 8 |
| 3.1 Introduction | 8 |
| 3.2 Data Collection | 8 |
| 3.3 Synthetic OoD Data Generation | 9 |
| 3.3.1 Feature Extraction | 9 |
| 3.3.2 Feature Perturbation Parameters | 13 |
| 3.3.3 Correlation Matrix | 14 |
| 3.3.4 Generation of Synthetic Data | 16 |
| 3.4 eXplainable AI (XAI) | 20 |
| 3.4.1 Feature and Value Ranking | 21 |
| 3.5 Out-of-Distribution Detection Algorithm | 22 |
| 4 Experimental Evaluation and Results | 27 |
| 4.1 Introduction | 27 |
| 4.2 Datasets | 27 |
| 4.3 Experimental Design | 29 |
| 4.4 Evaluation | 30 |
| 4.5 Results | 30 |

CONTENTS

| | | |
|----------|--|-----------|
| 4.5.1 | Experiment 1: Synthetic images with Complete Feature Set. | 31 |
| 4.5.2 | Experiment 2: Synthetic images generated excluding dominant features. | 38 |
| 4.5.3 | Experiment 3: Synthetic images generated without corrected covariance matrix. | 40 |
| 4.5.4 | Experiment 4: Evaluation of Anomaly Detection with Multiple Operational Splits ($Nop > 1$) | 42 |
| 4.5.5 | Summary of the results | 43 |
| 5 | Conclusion and Future works | 45 |
| 5.1 | Future Work | 46 |
| A | Appendix | 47 |
| A.1 | Feature Extraction and Synthetic Data Generation | 47 |
| A.1.1 | Relevant Imports | 47 |
| A.1.2 | Feature Extraction | 48 |
| A.1.3 | Generating Synthetic Data | 49 |
| A.2 | Out of distribution detection | 51 |
| A.2.1 | Weighted Mutual Information Calculation | 53 |
| A.3 | Process flow of rule generation in Rulex | 55 |
| | References | 55 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | Difference between anomaly and OoD detection. | 2 |
| 3.1 | Summing the gradient magnitudes at each pixel in the block. . . . | 10 |
| 3.2 | Illustration of GLCM with angle (0°, 45°, 90°, 135°). | 12 |
| 3.3 | Feature extraction workflow | 13 |
| 3.4 | Feature correlation matrix | 16 |
| 3.5 | Example of comparing the Gaussian distribution of the saliency feature | 19 |
| 3.6 | Synthetic Image Generation and rule-based ODD Workflow | 26 |
| 4.1 | Samples of the Hazards&Robots Corridor Images | 28 |
| 4.2 | Sample OoD Cases: Object on Robot | 33 |
| 4.3 | Sample OoD Cases: Saw dust | 34 |
| 4.4 | Sample ID Cases: Normal_4 | 35 |
| 4.5 | Sample ID Cases: Normal_2 | 35 |
| 4.6 | Sample Misclassification Case: Normal Desks | 36 |
| 4.7 | Experiment 1 Feature Ranking (all features) | 38 |
| 4.8 | Experiment 2 Feature ranking (features without HOG and GLCM correlation) | 39 |
| A.1 | RuleX Flow | 55 |

List of Tables

| | | |
|------|---|----|
| 3.1 | Formulas and Descriptions for Texture Features | 12 |
| 3.2 | Training numbers of hits table | 24 |
| 3.3 | Operational numbers of hits table | 24 |
| 4.1 | Feature Variance Percentages used in the Experiment | 29 |
| 4.2 | Baseline Results for Experiment 1: Weighted Mutual Information and Norms | 32 |
| 4.3 | Experiment 1 Anomaly Detection Results with Complete Feature Set | 32 |
| 4.4 | Rules generated with Complete Feature Set | 37 |
| 4.5 | Baseline Results for Experiment 2: Weighted Mutual Information and Norms. | 39 |
| 4.6 | Experiment 2 Anomaly Detection Results without highly attributed features | 40 |
| 4.7 | Baseline Results for Experiment 3: Weighted Mutual Information and Norms. | 41 |
| 4.8 | Experiment 3 Anomaly Detection Results without corrected covariance matrix. | 41 |
| 4.9 | Baseline Results for Experiment 4: Weighted Mutual Information and Norms. | 42 |
| 4.10 | Experiment 4 Anomaly Detection Results with Complete Feature Set | 42 |

Chapter 1

Introduction

1.1 Background and Motivations

In the book *Identification of Outliers*, Hawkins defines an outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [1]. Such observations, often referred to as anomalies, abnormalities, or discordants, are central to understanding data integrity and reliability[2]. Anomalies present a unique challenge for systems, as they must be able to identify and recognize deviations from normal patterns. The study of anomaly detection, sometimes called outlier detection or novelty identification, has been an active research area for several decades, with early exploration dating back to the 1960s[3]. Anomaly detection typically aims to identify deviations from expected behavior within a given distribution, flagging rare or unusual occurrences that may indicate risks or errors. Out-of-distribution (OoD) detection extends this concept by addressing data that originates from conditions or environments not encountered during training. These data samples, described as belonging to a "different distribution", present unique challenges for predictive accuracy and model reliability, as they introduce unfamiliar features or characteristics outside the scope of the model's learned experience [4].

1.1 Background and Motivations

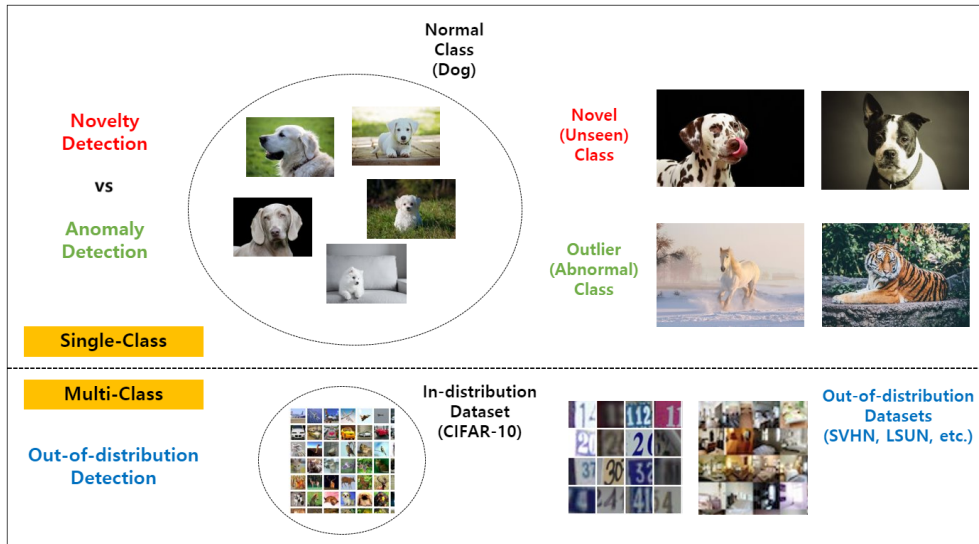


Figure 1.1: Difference between anomaly and OoD detection [5]

To address these challenges, a robust and trustworthy Machine Learning (ML) system must possess the ability to admit uncertainty and say “I don’t know” [6] when confronted with unfamiliar or OoD inputs. This capability ensures that critical decisions can be deferred to human experts rather than relying on error-prone predictions, thereby minimizing potential risks. By integrating mechanisms to handle uncertainty and manage OoD scenarios, these systems can achieve the dual goals of accurate decision-making and enhanced safety. This approach is essential for ensuring the reliability and accountability of ML models, particularly in safety-critical domains where failures can have severe consequences.

Autonomous systems, particularly autonomous vehicles, rely on AI for critical real-time decision-making in dynamic and unpredictable environments. Trustworthiness, built through transparent decision-making processes, is paramount for ensuring safety and user confidence [7]. Out-of-distribution (OoD) detection plays a key role in this context, allowing the system to recognize and manage unfamiliar or unexpected inputs. Robust OoD detection capabilities enable these systems to navigate diverse scenarios more reliably, reducing risks and improving overall safety. Similarly, in medical applications such as disease diagnosis and organ segmentation, OoD detection ensures that AI models can handle real-world clinical data that often deviates from the training data distribution, thus preventing silent failures and misdiagnoses. By recognizing when a model lacks the confidence to make accurate predictions, it can defer decisions to human experts, improving diagnostic accuracy and patient safety [6]. Both domains benefit from integrating OoD detection with related tasks like anomaly detection and un-

certainty quantification, enhancing AI’s adaptability and reliability in complex, real-world environments.

The effectiveness of an OoD detection system relies on pattern recognition, particularly in dynamic and unpredictable environments where the system must be capable of adapting to new, unseen data [8]. The challenge in OoD detection lies in identifying anomalies without requiring fine-grained annotations or access to outliers in the training process. Collecting such fine-grained labels can be both costly and resource-intensive [9]. Many existing state-of-the-art OoD detectors either fail to scale up to complex data modalities like images [10] or assume the availability of fine-grained labels for in-distribution samples[11][12][13].

1.2 Contributions

Motivated by the need for adaptable and robust detection systems, this thesis integrates unsupervised methods into existing frameworks for out-of-distribution detection(ODD). The foundational research rule-based OoD detection [14] focused on labeled datasets, leveraging histograms generated during the training phase as "fingerprints" to evaluate runtime data [15]. However, this dependency on labeled data poses significant challenges, as the existing state-of-the-art OoD detectors, which is obtaining high-quality labeled datasets is often resource-intensive and impractical in real-world applications.

To address these limitations and an extension to the foundational work [14], the proposed approach introduces unsupervised methods that analyze the inherent statistical properties and deviations in the data, enabling OoD detection without predefined labels. This methodology retains the interpretability and efficiency of rule-based systems [14] while simplifying implementation. Eliminating the reliance on labeled datasets enhances scalability and adaptability, making it particularly suited for complex environments such as autonomous systems, healthcare, and robotics.

The proposed approach integrates synthetic OoD data generation with histogram analysis to enhance detection capabilities and ensure resilience against unfamiliar inputs. Synthetic data is generated by perturbing the statistical properties of the in-distribution dataset through controlled variance adjustments while ensuring realistic deviations that mimic potential anomalies. These transformations simulate a wide range of OoD scenarios, enabling the model to encounter and adapt to patterns beyond the training distribution, a critical capability for navigating the complexity of real-world environments.

Complementing this, histogram analysis tracks the frequency of rule activations within a rule-based model during runtime. By comparing these runtime histograms to those generated during the training phase, the system identifies

OoD scenarios through clear discrepancies. This structured tracking method ensures reliable detection, particularly in dynamic and unpredictable contexts, where deviations from expected patterns require swift identification.

To enhance scalability and adaptability, the system incorporates an incremental approach to data handling. New samples are seamlessly integrated into operational batches, while the most distant data points from the previous batch are removed. This maintains a compact, up-to-date dataset aligned with current operations. The recalibration process generates a split collection by updating rule activations, enabling the system to adjust in real-time to shifts in the operational environment.

Finally, the approach is designed to be parameter-free and leverage unsupervised techniques, simplifying its implementation and maximizing flexibility across diverse applications. The system minimizes dependence on labeled data while providing a robust and adaptable framework for OoD detection.

1.3 Thesis Structure

The rest of the thesis is organized as follows:

- **Chapter 2** provides a literature review of out-of-distribution detection methods, focusing on existing techniques and their applications. It also covers data generation methods relevant to synthetic anomaly detection.
- **Chapter 3** details the methodology used in this research, including data pre-processing, feature extraction, synthetic data generation, and rule-based OoD detection.
- **Chapter 4** presents the experimental setup, results of the experiments, discusses the effectiveness of the proposed approach, and compares different configurations.
- **Chapter 5** summarizes the key findings, highlights the contributions of this work, and suggests future research directions.

Chapter 2

Related Works

2.1 Introduction

This chapter provides an overview of the existing literature on out-of-distribution (OoD) detection systems and statistical methods for generating synthetic data. Section 2.2.1 explores the key characteristics and challenges of OoD detection, with a specific emphasis on its applications in safety-critical systems and the integration of explainable AI for enhanced interpretability and decision-making. Section 2.2.2 reviews related works on generating synthetic images, highlighting various tools and techniques that can facilitate this process.

2.2 Literature Reviews

2.2.1 Out-of-distribution Detection

Out-of-distribution detection has been applied to enhance safety in systems like mobile robots, where models must identify unseen or anomalous inputs during runtime. One approach employs variational autoencoders to detect OoD samples in real time, integrated with a YOLO object detection network to ensure concurrent navigation and hazard detection. This enables the system to stop or adapt its behavior when encountering novel data, demonstrating reliability in dynamic environments [15]. While effective, such methods often rely on computationally intensive deep learning models, which can limit interpretability and scalability in resource-constrained applications. In contrast, rule-based methods offer a transparent alternative by producing interpretable "if-then" decision rules, such as those used in models like Switching Neural Networks. These methods maintain computational efficiency while providing clear insights into their decision-making processes, making them particularly suitable for real-time safety-critical systems

where interpretability and reliability are crucial [14].

The foundational paper "Rule-based Out-of-Distribution Detection" [13] lays the groundwork for addressing the critical issue of OoD detection in machine learning systems. The paper introduces a transparent, non-parametric method based on eXplainable Artificial Intelligence (XAI), leveraging rule hits histograms generated during the training phase as a baseline for operational data comparison. This approach avoids reliance on distributional assumptions and tuning-sensitive parameters, offering a robust way to detect OoD samples through metrics like weighted mutual information ($W\mu I$) and rule-based information (RBI). Its validation across diverse scenarios, including predictive maintenance and cybersecurity, underscores its versatility and precision in identifying distributional shifts.

This work enhances the original rule-based OoD detection approach by incorporating unsupervised data into the detection pipeline. This allows for the identification of out-of-distribution instances without the need for labeled operational datasets, addressing scenarios where labeled out-of-distribution samples are unavailable. By utilizing unsupervised data, the method broadens its applicability to more dynamic environments while retaining the key advantages of interpretability and computational efficiency.

2.2.2 Data Generation Methods

Data collection from the autonomous wheelchair involves capturing images of its environment to identify obstacles, such as cables and boxes, that may pose a risk to its navigation. These images serve as the foundation for training the Logic Learning Machine (LLM) and Rulex models to detect OoD instances. In order for the models to process these images effectively, we need to extract relevant, compact, and interpretable features. The extracted features should encapsulate the key information about the images that can help the system recognize when it encounters unfamiliar or hazardous conditions. The goal is to reduce the high-dimensional complexity of raw images while retaining critical details that are necessary for making accurate, real-time decisions.

For this purpose, the Athec library [16] was utilized to extract essential low-level image features, including edges, entropy, and saliency. These features are particularly useful for anomaly detection, as they capture important characteristics of the image's structure and content. The choice of Athec was driven by its ability to provide a set of hand-crafted visual features that are not only relevant and compact but also explainable. By leveraging these methods, the system can efficiently classify images, detect out-of-distribution data, and ensure that the wheelchair operates safely and reliably. Athec's design, rooted in established computational aesthetics and visual psychology, ensures that the features are interpretable, computationally efficient, and aligned with the goals of building an

explainable and trustworthy system.

In the paper [17], the authors explore several prominent methods for generating synthetic images, including Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and the use of 3D modeling software and game engines like Blender and Unreal Engine. GANs and VAEs are powerful tools for generating realistic synthetic data, with GANs using a generator-discriminator approach to create images that resemble real data, while VAEs encode images into a latent space for more probabilistic image generation. 3D modeling and game engines enable the creation of complex virtual environments that can be used for synthetic data generation. However, these methods are computationally intensive, requiring significant hardware resources, large datasets, and considerable training time to achieve high-quality results. As a result, while these approaches offer valuable solutions for data synthesis, they are often less practical for real-time, resource-constrained applications that need fast and efficient data generation.

Chapter 3

Methodology

3.1 Introduction

This chapter describes the systematic approach undertaken to develop a robust out-of-distribution detection system. The methodology integrates feature extraction, synthetic data generation, and rule-based validation to ensure the system’s adaptability and reliability in identifying OoD scenarios. Each component is designed to address the challenges of operating in dynamic environments, emphasizing scalability and efficiency while reducing reliance on labeled datasets.

The methodology is structured into several key phases, beginning with data pre-processing and feature extraction, where critical attributes are derived from image datasets to enhance the model’s understanding. Following this, synthetic image data generation introduces controlled variations, simulating diverse OoD scenarios to improve the robustness of the detection system. The generated features are subsequently analyzed using Rulex, a large language model specializing in rule generation. Rulex formulates detection rules based on extracted attributes, which are evaluated using a rule hits algorithm to measure their effectiveness against OoD inputs. Each methodological choice is guided by the research aim of advancing ODD capabilities while minimizing dependency on labeled datasets and ensuring scalability across diverse applications.

3.2 Data Collection

The dataset used in this research is the Hazards&Robots dataset, an open-access resource designed for visual anomaly detection in robotics. Developed from the University of Lugano (USI-SUPSI) [18], this dataset provides a comprehensive benchmark for training and evaluating visual anomaly detection methods, including those based on deep learning vision models. It consists of 324,408 RGB

frames and their corresponding feature vectors, categorized into 145,470 normal frames and 178,938 anomalous frames across 20 distinct anomaly classes.

The data was recorded using a DJI Robomaster S1 robot equipped with a front-facing camera. The robot traversed university corridors under human control, capturing diverse scenarios to simulate real-world robotic navigation challenges. Anomalies in the dataset include the presence of humans, unexpected objects on the floor, and defects in the robot itself.

3.3 Synthetic OoD Data Generation

3.3.1 Feature Extraction

To generate synthetic OoD (Out-of-Distribution) data, the feature extraction process begins by analyzing the "normal" (in-distribution) data, which in this case corresponds to images of an empty corridor. Since anomalous data is expected to include objects, humans, or other disturbances within the corridor, the goal is to capture these deviations through selected features. To achieve this, we focus on features that are highly sensitive to environmental changes, such as edges, textures, and gradients.

Edge features were selected because they are effective at detecting the boundaries of objects, humans, or irregularities that break the uniformity of the corridor. The Canny edge detector [19], implemented by the Athec library [16], calculates the gradient magnitude to extract edges, making it sensitive to rapid intensity changes, which are characteristic of anomalies. The mathematical formulation of the canny edge detection process [19] involves calculating the gradient at each pixel in the horizontal (G_x) and vertical (G_y) directions, and then combining them to compute the gradient magnitude:

$$\text{Edge Magnitude}(x, y) = \sqrt{G_x^2 + G_y^2} \quad (3.1)$$

where G_x and G_y are the gradients in the horizontal and vertical directions, respectively. This feature highlights areas of significant change, making it particularly useful for identifying potential outliers in the corridor.

Histogram of Oriented Gradients (HOG) features were also included, as they are highly sensitive to variations in shapes and orientations, typical of anomalous objects. HOG calculates the distribution of gradient orientations in localized image regions, making it effective for capturing the structure of objects. The formula for computing HOG features [16] involves calculating histograms of gradient orientations within cells and normalizing them across blocks:

3.3 Synthetic OoD Data Generation

$$HOG = \bigcup_{B \in \text{Blocks}} \frac{H_B}{\sqrt{\|H_B\|_2^2 + \epsilon^2}} \quad (3.2)$$

where:

$$H_B = \sum_{C \in B} \sum_{(x,y) \in C} \|\nabla I(x,y)\| \cdot w_k(x,y)$$

- H_B is the histogram of gradients for a block, computed by summing the gradient magnitudes at each pixel in the block, weighted by the orientation bin $w_k(x,y)$.
- $\|H_B\|_2$ is the L2 norm of the histogram H_B .
- ϵ is a small constant added to the denominator for numerical stability.
- τ is a threshold for clamping the values (e.g., 0.2 in L2-Hys normalization).

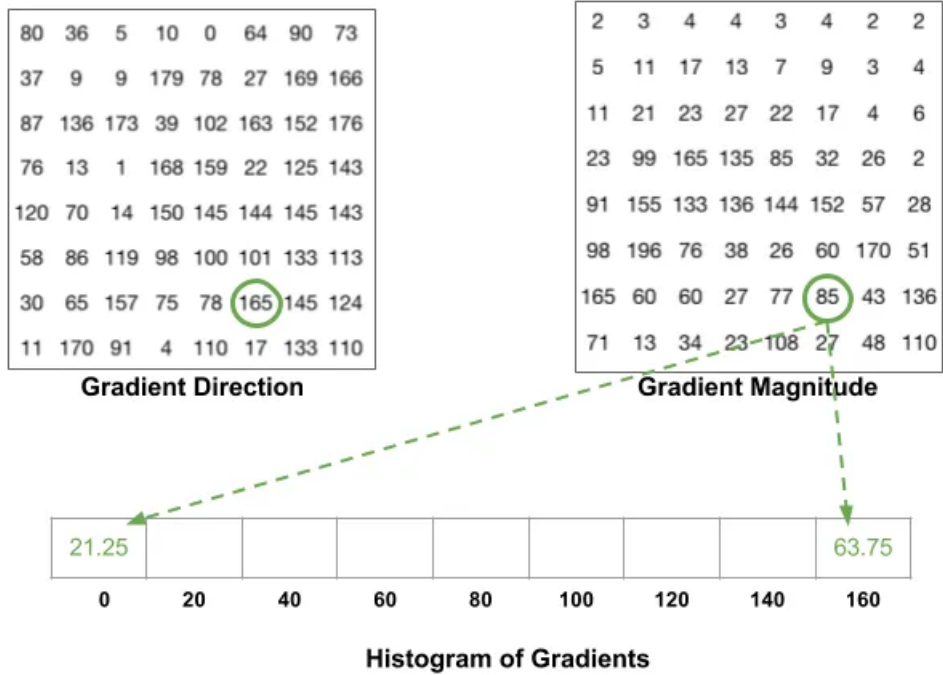


Figure 3.1: Summing the gradient magnitudes at each pixel in the block [20]

In addition, entropy was chosen as a feature because it measures the level of randomness or disorder in an image. Anomalous regions tend to introduce irregular

3.3 Synthetic OoD Data Generation

patterns, which are detected as areas of high entropy. Entropy $H(x)$ is computed using the probability distribution of pixel intensities:

$$H(x) = - \sum_i p(x_i) \log p(x_i) \quad (3.3)$$

where $p(x_i)$ represents the probability of pixel intensity x_i .

Saliency features are used to identify the most visually significant regions of an image, reflecting the parts of the image that attract attention due to their distinctiveness. These features are essential for detecting areas of potential out-of-distribution data, as anomalies often appear as highly salient regions that stand out from the background. In the proposed approach, saliency is computed using the spectral residual method, which identifies salient regions by removing the spectral residuals from the image’s Fourier transform and enhancing the remaining components. The resulting saliency map $S(x, y)$ is typically expressed as:

$$S(x, y) = \frac{I(x, y)}{\hat{I}(x, y)} \quad (3.4)$$

where $I(x, y)$ is the original image intensity at pixel (x, y) , and $\hat{I}(x, y)$ is the enhanced image derived by removing the spectral residual. The saliency map emphasizes anomalous or unexpected areas that are visually distinct and critical for identifying potential outliers or new, unseen patterns in the data.

On the other hand, Gray-Level Co-occurrence Matrix (GLCM) is a powerful texture feature extraction technique that quantifies spatial relationships between pixels with similar intensities. It helps capture the structural and textural characteristics of the image, which are often disrupted by anomalous objects or disturbances. The GLCM is computed by considering pixel pairs in specific spatial orientations (e.g., 0° , 45° , 90° , 135°) and calculating the joint probability of pixel intensity pairs (i, j) separated by a fixed distance.

3.3 Synthetic OoD Data Generation

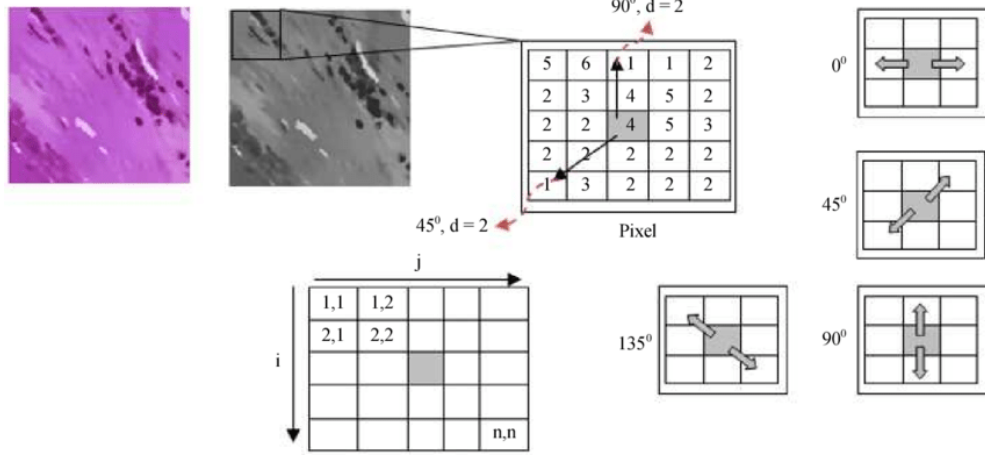


Figure 3.2: Illustration of GLCM with angle (0° , 45° , 90° , 135°) [21]

The four key properties extracted from the GLCM are contrast, correlation, energy, and homogeneity.

- *Contrast* measures the local variations in intensity.
- *Correlation* quantifies how correlated a pixel is to its neighbor in the specified direction.
- *Energy* represents the sum of squared elements in the GLCM, which gives an idea of the uniformity in texture.
- *Homogeneity* measures the closeness of the distribution of elements in the GLCM to the diagonal, reflecting the smoothness of the texture.

| Formula | Description |
|-----------------------|--|
| Contrast(x, y) | $\text{Contrast}(x, y) = \sum_{i,j} (i - j)^2 p(i, j)$ |
| Correlation(x, y) | $\text{Correlation}(x, y) = \frac{\sum_{i,j} (i - \mu_i)(j - \mu_j) p(i, j)}{\sigma_i \sigma_j}$ |
| Energy(x, y) | $\text{Energy}(x, y) = \sum_{i,j} p(i, j)^2$ |
| Homogeneity(x, y) | $\text{Homogeneity}(x, y) = \sum_{i,j} \frac{p(i, j)}{1 + i - j }$ |

Table 3.1: Formulas and Descriptions for Texture Features [22]

3.3 Synthetic OoD Data Generation

These texture features, particularly **contrast** and **correlation**, are vital for distinguishing between normal and anomalous regions in an image. High contrast and low correlation often indicate the presence of anomalies, such as objects or disturbances, that deviate from the typical texture of the empty corridor.

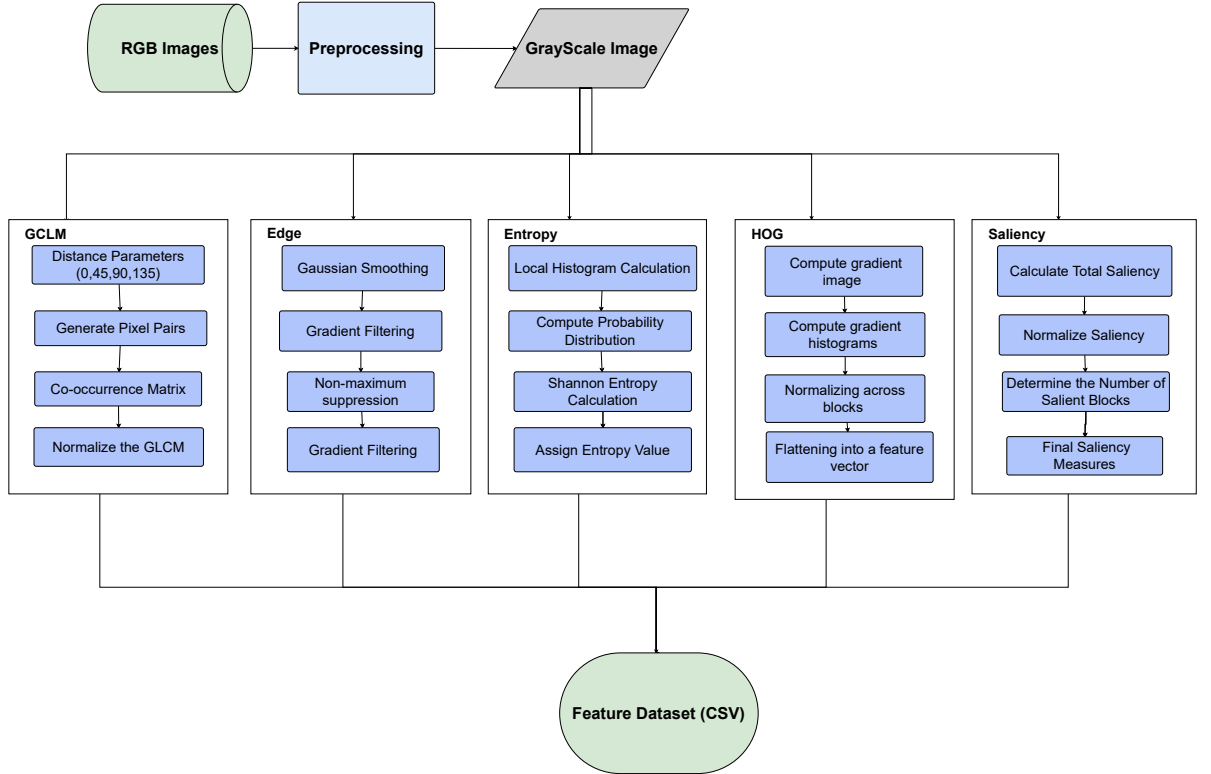


Figure 3.3: Feature extraction workflow: The feature extraction process involves computing edges using the Canny algorithm, entropy to capture randomness, and saliency to highlight significant regions through $n \times n$ block analysis. HOG is used to extract gradient orientations, while GLCM analyzes texture through spatial relationships at specific distances and orientations (0° , 45° , 90° , and 135°).

3.3.2 Feature Perturbation Parameters

The key features outlined in Section 3.3.1—namely edges, entropy, saliency, HOG, and GLCM—are systematically perturbed to simulate anomalies. Each feature is associated with a variance percentage, which controls the degree of perturbation applied. This percentage reflects the expected sensitivity of the feature to anomalies, ensuring that the perturbations are realistic and interpretable.

3.3 Synthetic OoD Data Generation

- **Entropy:** A lower variance range is selected as this feature measures the randomness or texture complexity of the image. Subtle changes in texture caused by anomalies lead to minor deviations, which can be effectively captured without significant alterations to the overall structure.
- **Edges:** A higher variance range is appropriate for edge features, which are sensitive to structural disruptions such as the introduction of new objects or humans. These changes significantly alter the edges in an image, justifying larger perturbations.
- **Saliency:** A moderate variance range is chosen for saliency, as it highlights visually prominent regions in the image. While anomalies can influence these areas, they generally do not drastically affect the image's overall structure.
- **GLCM Contrast:** A moderate variance range is used to capture intensity differences between pixels. Anomalies often introduce variations in texture or shading, making this level of perturbation suitable for simulating such changes.
- **GLCM Correlation:** A lower variance range is applied to correlation, which measures linear dependency between pixel intensities. Anomalies typically have minimal effects on this feature, necessitating only slight perturbations.
- **GLCM Energy:** Moderate variance is used for energy, which represents texture uniformity. Anomalies disrupt these uniform patterns, and this range captures such disruptions without overrepresentation.
- **GLCM Homogeneity:** A higher variance range is selected to simulate significant changes in texture caused by anomalies, such as the appearance of new objects, which disrupt the similarity of pixel intensities.
- **HOG:** A higher variance range is applied, as this feature encodes shapes and orientations. Anomalies often introduce new shapes or modify existing ones, requiring larger adjustments to accurately represent these changes.

3.3.3 Correlation Matrix

When generating synthetic data, especially out-of-distribution data, it is crucial to respect the relationships among features in the original dataset to maintain realism. If these relationships are ignored, the synthetic data may not accurately represent plausible scenarios, reducing its utility in anomaly detection or model validation. For instance:

3.3 Synthetic OoD Data Generation

- In the context of extracted image features, such as entropy and edges, their interdependence often reflects underlying environmental or textural properties.
- Ignoring correlation could lead to generating unrealistic combinations of feature values, undermining the credibility of the synthetic dataset.

Correlation is a measure of a monotonic association between two variables. A monotonic relationship between two variables is one in which either (1) as the value of a variable increases, so does the value of the other variable; or (2) as the value of a variable increases, the other variable value decreases[22]. The Pearson correlation coefficient is a statistical measure that quantifies the linear relationship between two continuous variables. The coefficient value, r , ranges between -1 and 1, where 0 is no correlation, 1 is a perfect positive correlation, and -1 is a perfect negative correlation.

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (3.5)$$

Where:

- x_i and y_i : The data points of the two variables.
- \bar{x} and \bar{y} : The means of the variables x and y , respectively.
- n : The total number of data points.

3.3 Synthetic OoD Data Generation

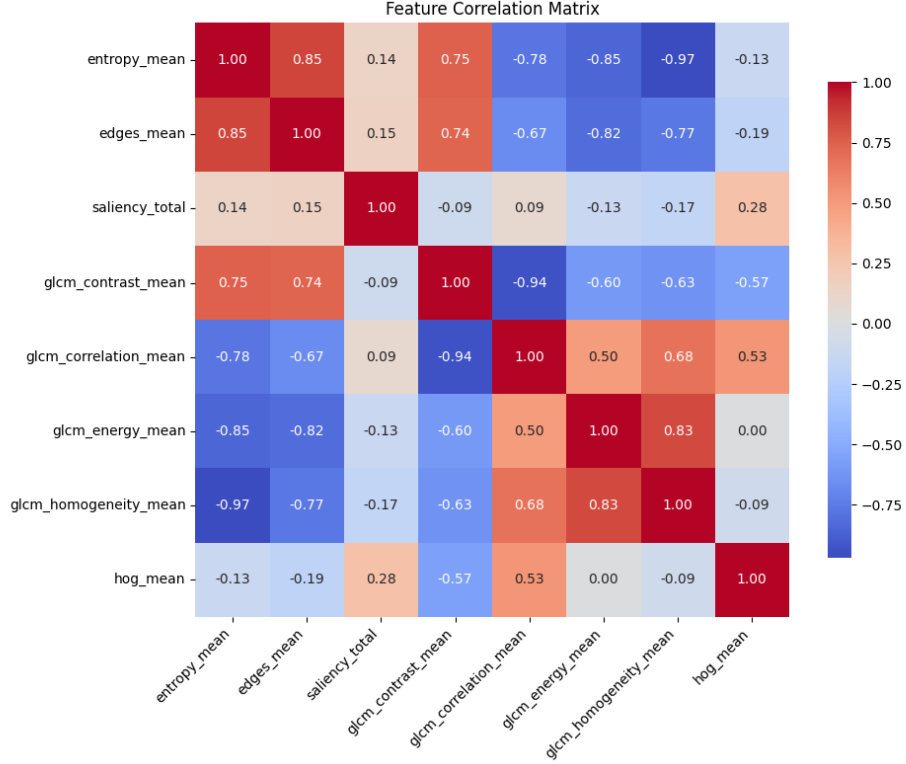


Figure 3.4: Feature correlation matrix: entropy shows a strong positive correlation with edges, suggesting that images with higher entropy values tend to exhibit more pronounced edge features. Meanwhile, glcm_correlation has a strong negative correlation with glcm_contrast. Additionally, glcm_homogeneity and glcm_energy are highly positively correlated, indicating potential dependencies between these features.

3.3.4 Generation of Synthetic Data

The perturbation process aims to simulate anomalies in feature space and generate plausible synthetic OoD samples. The key features extracted earlier (see Section 3.3.1) are perturbed to deviate beyond their observed in-distribution ranges, reflecting potential variations caused by anomalies. These deviations are designed to simulate realistic outliers, ensuring that the synthetic samples maintain the inherent structure and correlations between features. This process helps in generating data that mimics real-world OoD scenarios, where the feature distributions are altered due to anomalies, while still respecting the original relationships among features.

The perturbation process leverages Gaussian distributions to model the devi-

3.3 Synthetic OoD Data Generation

ations introduced in the feature space.

The multivariate normal distribution, also known as the multivariate Gaussian distribution or joint normal distribution, is a generalization of the one-dimensional normal distribution to higher dimensions. It models the probability distribution of a vector of correlated real-valued random variables, each of which follows a normal distribution. Rather than calculating the probability for each variable individually and multiplying, the multivariate normal distribution provides a unified approach to describe the joint probability of all the variables together. In the multivariate normal distribution, the mean vector represents the central location in the feature space where the samples are most likely to occur. This is similar to the peak of the bell curve in the univariate normal distribution. The covariance matrix plays a critical role by capturing the relationships and dependencies between the variables. The diagonal elements of this matrix represent the variances of each feature (i.e., how much a feature varies). In contrast, the off-diagonal elements represent the covariances between different features (i.e., how two features co-vary). A higher covariance indicates that two features tend to change in tandem, while a lower covariance suggests a weaker or no relationship between the features.

The mean is a coordinate in N-dimensional space, which represents the location where samples are most likely to be generated. This is analogous to the peak of the bell curve for the one-dimensional or univariate normal distribution. In many cases, simplifying assumptions are made to approximate the covariance structure, such as using spherical covariance or diagonal covariance. In the spherical covariance case, the covariance matrix is a scaled identity matrix, meaning all features are assumed to have the same variance and are uncorrelated. The diagonal covariance approximation assumes that the features are uncorrelated, with only the variances of individual features represented on the diagonal of the covariance matrix.

Given the feature values and their correlations, the synthetic OoD data for each sample is generated as follows:

1. **Calculating Feature Perturbation:** Each feature value x_i is perturbed using a variance percentage Δ_i . This perturbation factor scales the observed feature value to compute the variance σ_i^2 for each feature, which quantifies the expected deviation or anomaly for that feature:

$$\sigma_i^2 = (x_i \times \Delta_i)^2 \tag{3.6}$$

where x_i is the observed feature value and Δ_i is the perturbation factor for feature i .

3.3 Synthetic OoD Data Generation

2. **Calculating Standard Deviations:** From the variance, the standard deviation σ_i for each feature is computed as:

$$\sigma_i = \sqrt{\sigma_i^2} \quad (3.7)$$

This step converts the variance into standard deviation, which will later be used to adjust the feature values.

3. **Mean Vector Calculation for each Feature:** After calculating the standard deviation σ_i , each feature's observed value x_i is adjusted by applying a deviation factor. This adjustment is made to reflect the expected anomaly in the feature's distribution. The modified feature mean is given by:

$$\mu_i = x_i + (\Delta_i \times \sigma_i) \quad (3.8)$$

where:

- x_i is the original feature value (the observed value).
 - Δ_i is a scaling factor that controls how much deviation to apply to the feature value.
 - σ_i is the standard deviation calculated from the variance of feature i , determined by the perturbation factor Δ_i .
4. **Generate Covariance Matrix:** The covariance matrix captures the linear relationships between features. To generate the covariance matrix, we first compute the correlation matrix, which is structured as follows:

$$\mathbf{R}_c = \begin{bmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{bmatrix} \quad (3.9)$$

where ρ_{ij} represents the correlation between feature i and feature j .

Next, we scale the correlation matrix by the variances of each feature. This is done by multiplying the correlation matrix by the diagonal matrix of the variances σ_i^2 , resulting in the covariance matrix Σ :

$$\Sigma = \mathbf{R}_c \times \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2) \quad (3.10)$$

The covariance matrix Σ incorporates both the individual feature variances and the correlations between features, ensuring that the synthetic data reflects both individual feature variability and feature relationships.

3.3 Synthetic OoD Data Generation

5. **Sampling from the Multivariate Normal Distribution:** After calculating the mean vector μ and covariance matrix Σ , synthetic data is generated by sampling from the multivariate normal distribution.

Using the mean vector μ and covariance matrix Σ , synthetic data points can be generated by sampling as follows:

$$x_i^{\text{synthetic}} \sim \mathcal{N}(\mu_i, \Sigma) \quad (3.11)$$

This means that each feature in $\mathbf{x}_i^{\text{synthetic}}$ is generated such that:

- The feature values follow the distributions specified by the mean vector μ ,
- The relationships (correlations) between features are accounted for by the covariance matrix Σ .

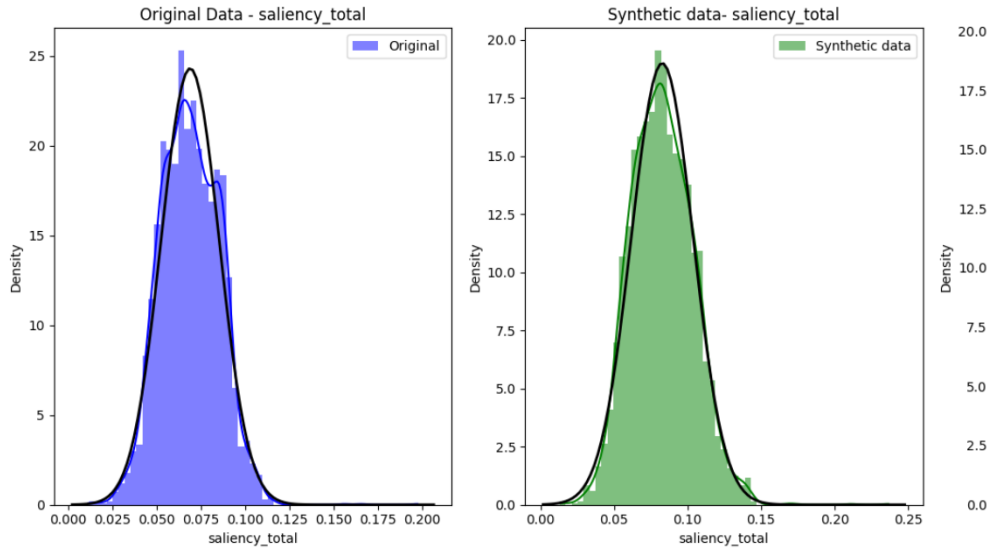


Figure 3.5: Example comparing the Gaussian distribution of the saliency feature normal unperturbed image and synthetically generated image. Here we can see we have shifted the mean while keeping the overall distribution similar.

3.4 eXplainable AI (XAI)

The problem is framed as a canonical supervised task, comparing real and synthetic (potentially anomalous) images. The aim is to generate a model that outlines the behavioral characteristics of real images. Following this, Explainable AI (XAI) is employed through the Logic Learning Machine (LLM) to acquire insights into the behavior of real images and refine the model with the fewest possible rules. This method ensures both effectiveness and interpretability, providing a transparent decision-making process.

While other black-box approaches, such as Support Vector Machines (SVMs) or deep neural networks (DNNs), could address the same task of separating real and anomalous data, they are often difficult to interpret. Reducing the complexity of such models through techniques like model reduction can be challenging because they exploit all features jointly. In contrast, the LLM approach provides a more interpretable solution, without compromising performance.

Logic learning machine (LLM) is a machine learning method based on the generation of intelligible rules. It is an efficient implementation of the Switching Neural Network (SNN) [23] paradigm, developed by Rulex (<https://www.rulex.ai/rulex-explainable-ai-xai/>)¹, a platform created by the Italian National Research Council CNR-IEIIT in Genoa.

An LLM aims to construct a classifier $g(x)$, described by a set of rules structured in the form

if $\langle \textit{premise} \rangle$ **then** $\langle \textit{consequence} \rangle$

where $\langle \textit{premise} \rangle$ is a logical product (AND) of conditions on the input features, while $\langle \textit{consequence} \rangle$ corresponds to the output class.

The model generation goes through three steps:

1. *Latticization (discretization and mapping to a Boolean lattice)* [24]: Each variable is transformed into a string of binary data in a designated Boolean lattice using the inverse only-one binarization. Finally, for each sample, all these strings are concatenated into one large string.
2. *Shadow Clustering*: A set of binary values (implicants) is generated, facilitating the identification of groups of data points associated with specific classes.
3. *Rule generation*: All the implicants are converted into a collection of simple conditions and eventually combined to form a set of comprehensible rules.

¹Rulex: <https://www.rulex.ai/rulex-explainable-ai-xai/>

In rule-based models, conditions for a rule are constructed by considering all the involved variables at the same time. This leads to interdependent conditions within the rules. Specifically for LLMs, implicants (binary strings within a Boolean lattice, as previously explained) are used to build conditions. These implicants define unique data point groups associated with a specific class. The groups of points in the Boolean space are later turned into regulations that combine a portion of joint variables. It is easy to generate a clear rule from an implicant that shows a logical product of threshold conditions found through the discretization phase. When generating implicants using the Shadow Clustering technique in LLMs, which examines the complete training set, the resulting rules may overlap and represent distinct important aspects of the phenomenon under study[25][26].

3.4.1 Feature and Value Ranking

An eXplainable Artificial Intelligence (XAI) model enables the examination of its results through feature and value ranking.

Consider a set of m rules $\mathbf{r}_k, k = 1, \dots, m$, each comprising d_k conditions $cl_k, l_k = 1, \dots, d_k$. Let X_1, \dots, X_n be the input variables, such that $X_j = x_j \in \mathcal{X} \subseteq \mathbb{R}$ for all $j = 1, \dots, n$. Let \hat{y} also represent the class assigned by the rule and y_j the actual output of the j -th instance.

A condition c_{l_k} involving the variable X_j can take one of the following forms:

$$X_j > s, \quad X_j \leq t \text{ or } s < X_j \leq t.$$

where $s, t \in \mathcal{X}$.

For each rule, it is possible to define a confusion matrix that consists of four indices: $TP(\mathbf{r}_k)$ and $FP(\mathbf{r}_k)$, defined as the number of instances (x_j, y_j) that satisfy all the conditions in rule \mathbf{r}_k with $\hat{y} = y_j$ and $\hat{y} \neq y_j$ respectively; $TN(\mathbf{r}_k)$ and $FN(\mathbf{r}_k)$, defined as the number of examples (x_j, y_j) that do not satisfy at least one condition in rule \mathbf{r}_k , with $\hat{y} \neq y_j$ and $\hat{y} = y_j$, respectively.

Consequently, we can derive the following metrics:

$$C(\mathbf{r}_k) = \frac{TP(\mathbf{r}_k)}{TP(\mathbf{r}_k) + FN(\mathbf{r}_k)}$$

$$E(\mathbf{r}_k) = \frac{FP(\mathbf{r}_k)}{TN(\mathbf{r}_k) + FP(\mathbf{r}_k)}$$

The covering $C(\mathbf{r}_k)$ is adopted as a relevance measure for a rule \mathbf{r}_k ; in other words, the greater the covering, the more general the corresponding rule is considered. The error $E(\mathbf{r}_k)$ measures how many data points are incorrectly covered

3.5 Out-of-Distribution Detection Algorithm

by the rule. Both covering and error are used to define feature ranking and value ranking.

Feature ranking (FR) offers a ranking of the features utilized in the rule conditions based on a relevance measure. To obtain the relevance $R(c_{l_k})$ for a condition, we consider rule \mathbf{r}_k in which condition c_{l_k} appears, and the same rule without condition c_{l_k} , denoted as \mathbf{r}'_k . Since the premise part of \mathbf{r}'_k is less restrictive, we deduce that $E(\mathbf{r}'_k) \geq E(\mathbf{r}_k)$, and thus the quantity $R(c_{l_k}) = (E(\mathbf{r}'_k) - E(\mathbf{r}_k)) \cdot C(\mathbf{r}_k)$ can be used as a measure of relevance for the specific condition c_{l_k} . Each condition c_{l_k} pertains to a specific variable X_j and is verified by certain values $\nu_j \in \mathcal{X}$. In this way, a relevance measure $R_{\hat{y}}(\nu_j)$ for every value assumed by X_j is derived using the following equation:

$$R_{\hat{y}}(\nu_j) = 1 - \prod_k (1 - R(c_{l_k})),$$

where the product is computed over the rules \mathbf{r}_k that include a condition c_{l_k} verified when $X_j = \nu_j$. As $R_{\hat{y}}(\nu_j)$ takes values in $[0, 1]$, it can be interpreted as the probability that the value ν_j occurs in predicting \hat{y} . The same argument can be extended to intervals $I \subseteq \mathcal{X}$, thus defining the *Value Ranking (VR)*. Relevance scores are then ranked, revealing the most sensitive interval of the feature with respect to each class [27].

3.5 Out-of-Distribution Detection Algorithm

The algorithm used in this thesis, originally developed in the paper Rule-Based Out-of-Distribution Detection, leverages a rule-based learning approach to identify deviations from normal behavior. It builds on the concept of extracting rules from in-distribution (ID) data to characterize its inherent patterns and uses these rules to detect out-of-distribution (OoD) samples. The rule-set generated by the LLM captures the essential relationships between features, serving as a baseline for comparison during OoD detection.

The following steps outline the core components of the ODD algorithm implemented in this thesis

1. *Rule Creation:* The first step involves creating rules for each output class using the Rulex program as if training a classifier using the training dataset. The program generates a set of rules describing the classes we consider ID. In the case of very simple classes or classes with few images, it might happen that Rulex generates only one rule or two rules for that specific class. Having just one rule could pose problems for the algorithm, as we will see later.

3.5 Out-of-Distribution Detection Algorithm

2. *Rule Hits Tables Creation*: Once rules for ID classes are obtained, rule hits tables are created. Two types of tables are generated: one for training and one for operational purposes. Training tables use the dataset used to generate rules divided by each output class. Operational tables are generated from a dataset containing data that our algorithm must recognize as ID or OoD. For both datasets, random splits are created by randomly selecting data from the respective dataset (randomly selected data can appear in multiple splits but not multiple times in the same split). Rule hits tables are matrices $\mathbf{n} \times \mathbf{m}$, where \mathbf{n} is the number of splits, and \mathbf{m} is the number of rules. The training table for each class is created as follows, starting from the first split:

- (a) For each image in the current split, the algorithm checks if it satisfies each rule generated for that class.
- (b) The algorithm counts how many times each rule is satisfied by the images in the split and divides this number by the total number of images in the split (resulting in 1 if a rule is satisfied by every image in the split).
- (c) If there are multiple images in that split or if there are multiple splits, the process continues with the next image or split.

Operational tables are created similarly to training tables, except that rules are compared with images from the operational dataset. In the end, for each rule class, we will have a training rule hits table and an operational one.

In a more mathematical vein, let \mathcal{R}_{tr} represent a set of rules generated from a training set, where N_r signifies the number of rules it comprises. Let N_{tr} and N_{op} denote the number of splits in the training domain and operational domain, respectively, resulting in a total of $N_h = N_{tr} + N_{op}$ splits. Consider n_s as the quantity of data samples within a split. Within each split, samples may or may not satisfy individual rules a specific number of times, which we term the “number of hits” for that rule. Consequently, N_h vectors are defined, the value of which will be normalised by the split size n_s :

$$\mathbf{h}^j = \{h_i^j\}, \quad h_i^j \in [0, 1], i = 1, \dots, N_r, j = 1, \dots, N_h$$

Each vector \mathbf{h}^j is a table [28]. The structure of training and operational tables is as shown in Table 3.2 and 3.3.

3. *Out-of-Distribution Detection*: Once all the required tables have been acquired, the algorithm moves onto the ODD algorithm. For each class, the

3.5 Out-of-Distribution Detection Algorithm

| | | | |
|-----------|------------------|---------|-------------------------|
| | tr_1 | \dots | $tr_{N_{tr}}$ |
| r_1 | $h_1^{tr_1}$ | \dots | $h_1^{tr_{N_{tr}}}$ |
| r_2 | $h_2^{tr_1}$ | \dots | $h_2^{tr_{N_{tr}}}$ |
| \cdot | \cdot | \cdot | \cdot |
| \cdot | \cdot | \cdot | \cdot |
| r_{N_r} | $h_{N_r}^{tr_1}$ | \dots | $h_{N_r}^{tr_{N_{tr}}}$ |

Table 3.2: Training numbers of hits table. Each column refers to a training split tr_i and each row to a rule $r_i \in \mathcal{R}_{tr}$.

| | | | |
|-----------|------------------|---------|-------------------------|
| | op_1 | \dots | $op_{N_{op}}$ |
| r_1 | $h_1^{op_1}$ | \dots | $h_1^{op_{N_{op}}}$ |
| r_2 | $h_2^{op_1}$ | \dots | $h_2^{op_{N_{op}}}$ |
| \cdot | \cdot | \cdot | \cdot |
| \cdot | \cdot | \cdot | \cdot |
| r_{N_r} | $h_{N_r}^{op_1}$ | \dots | $h_{N_r}^{op_{N_{op}}}$ |

Table 3.3: Operational numbers of hits table. Each column refers to an operational split op_i and each row to a rule $r_i \in \mathcal{R}_{tr}$.

algorithm compares the corresponding training table with the operational table by comparing the divisions of the two tables. If the observed difference surpasses a certain interval, the pair of divisions currently being analyzed is deemed unsatisfactory. If more than half of the analyzed pairs are deemed unsatisfactory, the table is considered unsatisfactory itself.

The ODD algorithm can be split into three phases, commencing from the first class.

- (a) *Interval Determination*: The first step of the algorithm involves finding the interval that determines whether a calculated distance between a split of the training table and the operational table is considered IN or OUT. To do this, both the l_p norm with $p = 1$ or 2 and Weighted Mutual Information ($W\mu I$) were used in parallel:

3.5 Out-of-Distribution Detection Algorithm

1. l_p -Norm Approach:

$$l_p(tr_i, tr_j) = \left[\sum_{r=1}^{N_r} (|h_r^{tr_i} - h_r^{tr_j}|)^p \right]^{\frac{1}{p}}, \quad \forall i, \forall j$$

for each combination of splits within the training table. Once all distances between splits are calculated, the algorithm selects the maximum and minimum.

$$l_p^{base} \doteq [\min_{i,j}(l_p(tr_i, tr_j)), \max_{i,j}(l_p(tr_i, tr_j))]$$

These two values form our ID interval.

2. Weighted Mutual Information ($W\mu I$):

The weighted entropy for each split is computed as:

$$H(tr_i, tr_j) = - \sum_{r=1}^{N_r} [\alpha_{i,j} P(h_r^{tr_i}, h_r^{tr_j}) \cdot \log(\alpha_{i,j} P(h_r^{tr_i}, h_r^{tr_j}))]$$

where:

- $\alpha_{i,j}$: Weight reflecting the difference in hit counts between splits i and j ,
- $P(h_r^{tr_i}, h_r^{tr_j})$: Joint probability distribution of the hit counts.

The mutual information between splits i and j is calculated as:

$$W\mu I(tr_i, tr_j) = [H(tr_i) + H(tr_j) - H(tr_i, tr_j)], \forall i, \forall j$$

After computing the mutual information the algorithm then defines the base line for the ID interval.

$$W\mu I_{base} \doteq [\min_{i,j}(W\mu I(tr_i, tr_j)), \max_{i,j}(W\mu I(tr_i, tr_j))]$$

- (b) *Operational Table Evaluation*: For each operational table, distances between its splits and those of the training table are calculated using the l_p -norm. Each distance is compared against the maximum and minimum bounds of the ID intervals determined during the interval determination phase.

Additionally, the Weighted Mutual Information ($W\mu I$) approach is applied to compute feature correlations between the training and operational splits. The $W\mu I$ values for the operational splits are compared to the ID interval derived from the training data.

A table is classified as out-of-distribution (OoD) if:

3.5 Out-of-Distribution Detection Algorithm

- **For l_p -norm:** More than 50% of the calculated distances fall outside the maximum and minimum bounds.
- **For $W\mu I$:** The majority of $W\mu I$ values for operational splits fall outside the corresponding ID interval.

A simple overview of the flow of the methodology is displayed below in Figure 3.6.

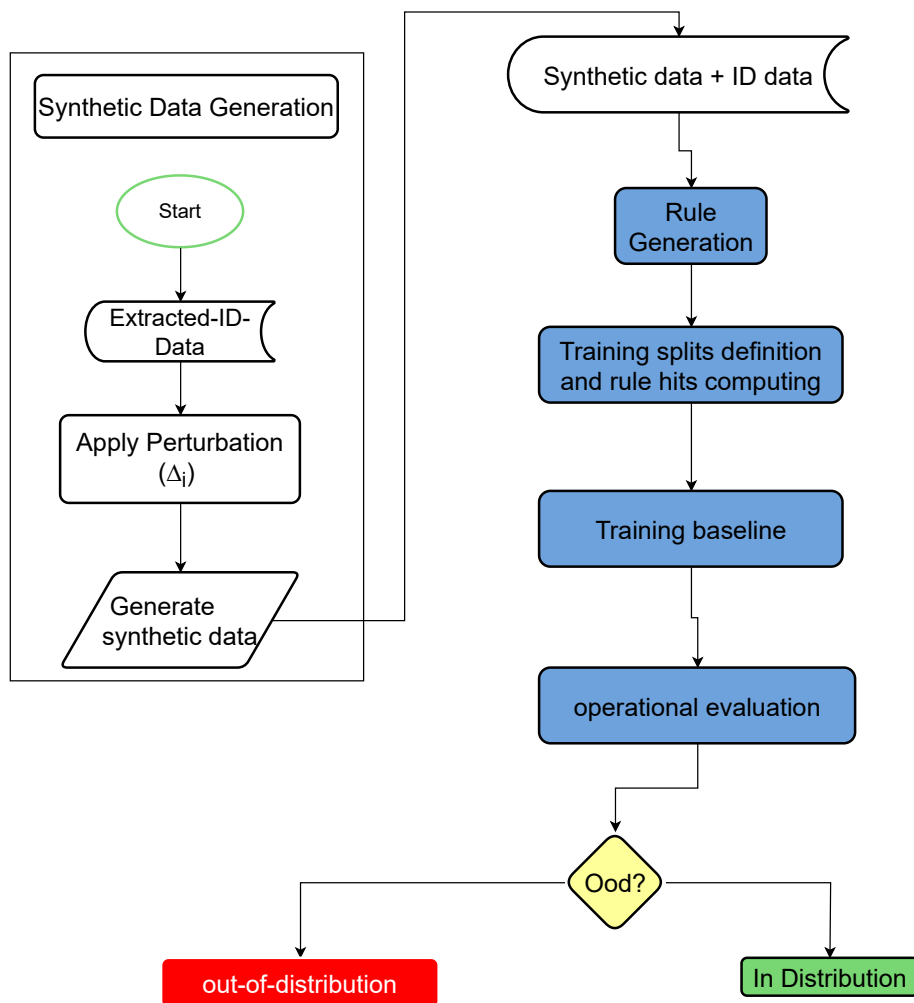


Figure 3.6: Synthetic Image Generation and rule-based ODD Workflow. **Note:** The operational evaluation considers both **Weighted Mutual Distributions and LP Norms**. For more details refer Section 3.5.

Chapter 4

Experimental Evaluation and Results

4.1 Introduction

This section presents the outcomes of rule generation, feature evaluation, and anomaly detection performance using the Logic Learning Machine (LLM). The analysis examines the effectiveness of two distinct rule sets: the initial rules derived from synthetic data and an expanded set generated by excluding specific high-relevance features. Additionally, the impact of feature correlations on detection performance is evaluated using covariance-based synthetic data.

The chapter begins by describing the dataset used for testing, providing details on its composition and relevance to the experiment. The evaluation then explores key performance metrics, such as anomaly detection accuracy, rule specificity, and feature sensitivity. The findings highlight trade-offs between rule-based detection effectiveness and feature selection through comparative analysis, offering insights into the model’s strengths and limitations.

4.2 Datasets

The Hazards&Robots dataset [18] serves as the basis for evaluating the system’s performance, providing both in-distribution and out-of-distribution data for assessing the effectiveness of the OoD detection system. It is divided into 21 classes: one ”normal” class and 20 anomaly classes. The anomaly classes used for testing in this work are box, cable, human, sawdust, water, object on robot, and object on robot 2. For training, 3,000 samples were selected from Environment 1 of the normal class, representing controlled conditions. During testing, additional normal samples exhibiting variations in lighting and surrounding objects were used,

along with samples from the anomaly classes.



Figure 4.1: Samples of the Hazards&Robots Corridor Images [23]

4.3 Experimental Design

The experiment begins with extracting features from the training set. As detailed in Section 3.3.1 and illustrated in Figure 3.3, the extraction process uses the AtheC-Library¹ to compute the mean values of the Edge, Entropy, HOG and Saliency features. The texture components are derived from the GLCM(Gray-Level Co-occurrence Matrix) using the scikit-image library².

This process results in a feature set file which is used as the base for generating synthetic images. The decision for the variance percentage (Δ_i) of each feature depends on each feature characteristics and sensitivity, particularly in the context of corridor images. In this experiment, the following percentages are applied, with the rationale for each choice explained in Section 3.3.2.

| Feature | Variance Percentage (Δ_i) |
|-----------------------|------------------------------------|
| Entropy Mean | 10% |
| Edges Mean | 30% |
| Saliency Total | 15% |
| GLCM Contrast Mean | 20% |
| GLCM Correlation Mean | 12% |
| GLCM Energy Mean | 18% |
| GLCM Homogeneity Mean | 25% |
| HOG Mean | 25% |

Table 4.1: Feature Variance Percentages used in the Experiment

As detailed in Section 3.3.4 of the methodology, the synthetic data generation is based on a Multivariate Normal Distribution. It takes the mean of the features along with the perturbation and covariance matrix as input and generates new data by sampling around the mean.

To generate rules the synthetically generated data merged with the training set, was provided as input to the LLM. During the merging processes an additional field, referred to as the "output," was introduced to distinguish between the two categories: synthetic data labeled as 0 (out-of-distribution) and in-distribution data labeled as 1.

¹<https://github.com/yilangpeng/athec>

²https://scikit-image.org/docs/0.24.x/auto_examples/features_detection/plot_glcmm.html

4.4 Evaluation

The system’s performance was evaluated across four different experiments. Each experiment tested a different approach to generating data and constructing rules. The performance was measured using **False Positive Rate (FPR)** and **False Negative Rate (FNR)** to assess the model’s ability to distinguish between anomalies and normal data. These metrics are computed based on operational splits:

- **FNR:** Measures the percentage of anomalous splits incorrectly classified as in-distribution. This only applies to anomalous splits since normal splits are expected to be classified as in-distribution.
- **FPR:** Measures the percentage of normal splits incorrectly classified as OoD. This only applies to normal splits since anomalies should be classified as OoD.

The evaluation also tested two different configurations for operational splits: single operational split ($Nop = 1$) and multiple operational splits ($Nop > 1$). For both configurations, weighted mutual information and Lp norms were used to handle rule hits across the training and operational datasets.

In the case where $Nop = 1$, the FPR and FNR were effectively binary due to the presence of only one operational split.

4.5 Results

The results of the four experiments, each focusing on different aspects of data generation, rule construction, and operational splits, are discussed in the following sections.

Each experiment introduced specific modifications to the data generation process to explore the impact of feature inclusion and statistical properties on classification accuracy:

- **Experiment 1:** Included all features, serving as a baseline to evaluate the performance of the system when provided with the complete dataset. The covariance matrix was used to maintain consistent feature correlations.
- **Experiment 2:** Excluded dominant features to test the system’s ability to handle data with reduced feature availability and assess the dependency of classification performance on these critical features. The covariance matrix was also used in this experiment to examine the impact of feature relationships on performance.

- **Experiment 3:** Generated data without incorporating the corrected covariance matrix to highlight the importance of maintaining feature correlations when generating data. This experiment aimed to assess the impact of disregarding feature relationships on the system’s performance and classification accuracy.
- **Experiment 4:** Used three operational splits to evaluate the system’s performance with multiple operational datasets. The purpose of this experiment was to test how the system handles data variability and generalization across different operational data splits.

As a reminder of how the results are calculated, the OoD classification process combines multiple methods— $W\mu I$, l_1 -norm, and l_2 -norm—into a unified decision-making framework. Rule hits tables were generated for both training and operational datasets by applying rules derived from the training data to splits of each dataset. These splits were used for training data to compute baseline intervals for the l_1 -norm, l_2 -norm, and Weighted Mutual Information ($W\mu I$), capturing the expected variation within ID data. After calculating the rule hits tables for training and operational splits, each split undergoes evaluation using these metrics against the precomputed ID baselines.

$W\mu I$ Evaluation: Weighted Mutual Information ($W\mu I$) is computed to assess the correlation between features. This approach is preferred over Mutual Information (μI) because it accounts for the varying importance of features, enabling a more accurate evaluation of feature relationships. If the majority of $W\mu I$ values for operational splits fall outside the ID interval, the dataset is flagged as OoD for this metric.

l_1 and l_2 Norms Evaluation: Distances between rule hits for training and operational splits are calculated using l_1 and l_2 norms. These distances are compared against the baseline intervals derived from training splits. If more than half of the calculated distances for operational splits exceed the ID interval in either norm, the dataset is flagged as OoD for that norm.

Final Decision: The system integrates the results from all three metrics— $W\mu I$, l_1 -norm, and l_2 -norm. If any of the metrics classify the dataset as OoD, the final decision is ”**Out-of-Distribution.**” Otherwise, the dataset is classified as ”**In-Distribution.**”

4.5.1 Experiment 1: Synthetic images with Complete Feature Set.

The synthetic images were generated using the complete feature set (GLCM, HOG, edge, entropy, saliency), with the covariance matrix considered. These

synthetic images, along with the training data, were provided to the LLM, which in turn generated **13** rules (Table 4.4). The baselines were then established by calculating intervals for $W\mu I$ and the norms (l_1 and l_2), as shown in Table 4.7, defining the acceptable bounds for ID data distributions.

| | $W\mu I$ | l_1 | l_2 |
|----------|---------------|---------------|---------------|
| Baseline | [0.0, 0.2141] | [0.0, 0.2082] | [0.0, 0.3013] |

Table 4.2: Baseline Results for Experiment 1: Weighted Mutual Information and Norms

| Anomaly | $W\mu I$ | l_1 | l_2 | FPR | FNR | Decision |
|-------------------|------------------|------------------|------------------|-----|-----|----------|
| Box | [0.3165, 0.3528] | [0.4832, 0.9422] | [0.3047, 0.3835] | - | 0 | OoD |
| Cable | [0.1462, 0.2876] | [0.4643, 0.9292] | [0.3004, 0.3889] | - | 0 | OoD |
| Object on Robot | [0.4581, 0.4709] | [0.5204, 0.9785] | [0.3155, 0.4133] | - | 0 | OoD |
| Water | [0.3501, 0.4433] | [0.5235, 0.9801] | [0.3210, 0.4127] | - | 0 | OoD |
| Sawdust | [0.3986, 0.5233] | [0.5104, 0.9751] | [0.3017, 0.3978] | - | 0 | OoD |
| Normal 2 | [0.1103, 0.2135] | [0.1976, 0.1700] | [0.1694, 0.2216] | 0 | - | ID |
| Normal 4 | [0.1204, 0.1439] | [0.1200, 0.1900] | [0.1750, 0.2450] | 0 | - | ID |
| Normal Desks | [0.2100, 0.2123] | [0.1300, 0.1500] | [0.1800, 0.2500] | 1 | - | OoD |
| Object on Robot 2 | [0.3222, 0.4147] | [0.5170, 0.9760] | [0.3086, 0.4049] | - | 0 | OoD |

Table 4.3: Experiment 1 Anomaly Detection Results with Complete Feature Set

Out-of-Distribution (OoD) Cases: Strong Deviations from Baseline

The OoD cases exhibit clear deviations from the baseline, with their weighted mutual information ($W\mu I$) values extending well beyond the upper threshold of 0.2141 (see Table 4.3). This distinction is further reinforced by the l_1 and l_2 norms, confirming their separation from in-distribution (ID) cases.

As shown in Figures 4.2 and 4.3, anomalies such as *Object on Robot*, and *Sawdust* display significantly higher $W\mu I$ values, with ranges like **[0.4581, 0.4709]** for *Object on Robot* and **[0.3986, 0.5233]** for *Sawdust*. Similar patterns are observed for *Box* and *Object on Robot 2*. These anomalies are correctly classified as OoD, demonstrating the reliability of the ODD approach.

Interestingly, *Water* presents a unique case in which its deviation from the baseline is not immediately apparent to the naked eye. Unlike more structurally distinct anomalies, *Water* likely differs due to texture-based features rather than structural one. Given its fine-grained surface patterns and potential reflections,

it is probable that **GLCM (Gray-Level Co-occurrence Matrix) texture features and entropy contribute most to its classification as OoD**. These features can capture localized intensity variations, contrast changes, and irregular texture patterns that distinguish *Water* from ID cases, even in the absence of strong edge-based differences. This reinforces the relevance of the selected image features, confirming that the approach effectively captures both structural and textural differences in the data.

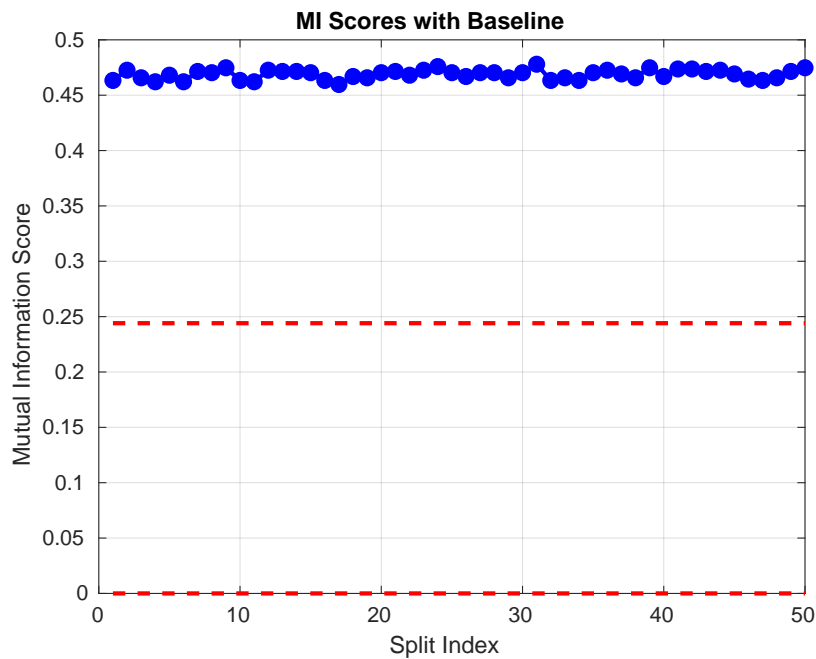


Figure 4.2: Sample OoD Cases: Object on Robot

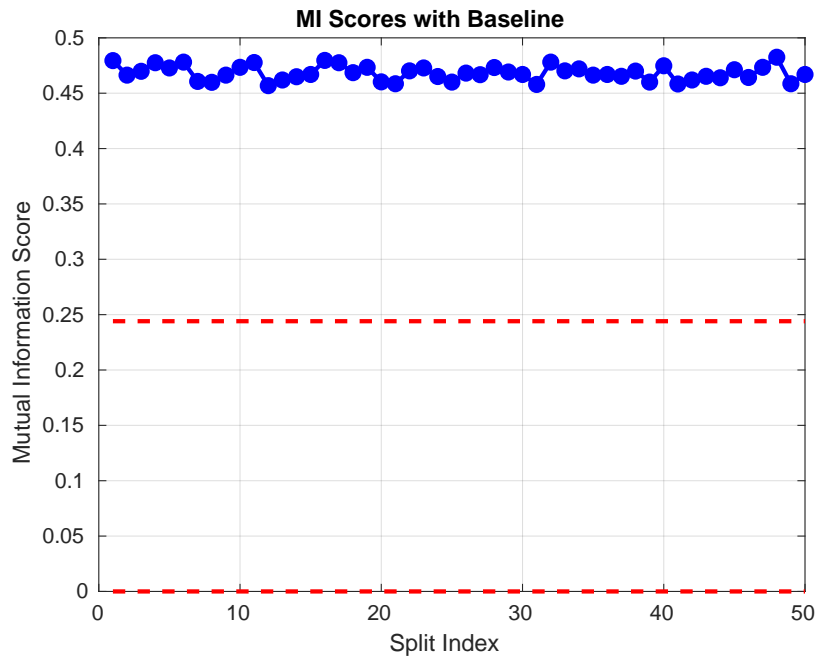


Figure 4.3: Sample OoD Cases: Saw dust

In-Distribution (ID) Cases: Inside the Baseline

The normal cases, shown in Figures 4.4 and 4.5, represent *Normal 2* and *Normal 4*. These samples have mutual information values that remain within the baseline threshold, with *Normal 2* ranging from $[0.1103, 0.2135]$ and *Normal 4* from $[0.1204, 0.1439]$. Both cases align closely with the established baseline in Table 4.7, ensuring that they are correctly classified as ID. The consistency across both $W\mu I$ and the norm-based scores further confirms their placement within the expected range.

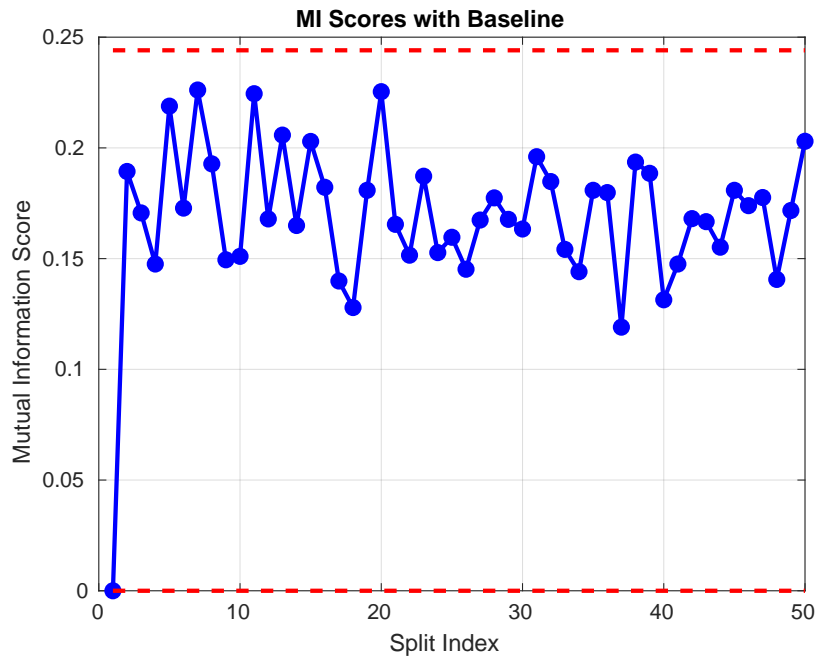


Figure 4.4: Sample ID Cases: Normal_4

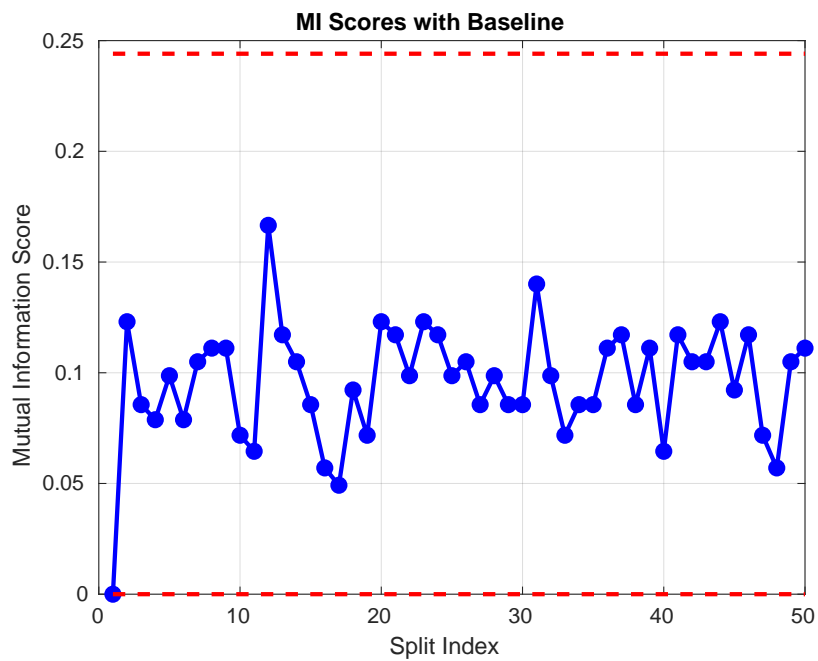


Figure 4.5: Sample ID Cases: Normal_2

Borderline Misclassification: Normal Desks

An interesting case is *Normal Desks*, shown in Figure 4.6. This case has $W\mu I$ values of $[0.2100, 0.2123]$, which are very close to the upper baseline threshold (0.2141). Despite this, it is classified as OoD because the majority of splits fall slightly above the limit. This suggests that while the method is effective in identifying clear-cut anomalies, borderline cases near the threshold may be more sensitive to small variations, leading to potential false positives. Fine-tuning the decision criteria could help reduce misclassifications in such cases.

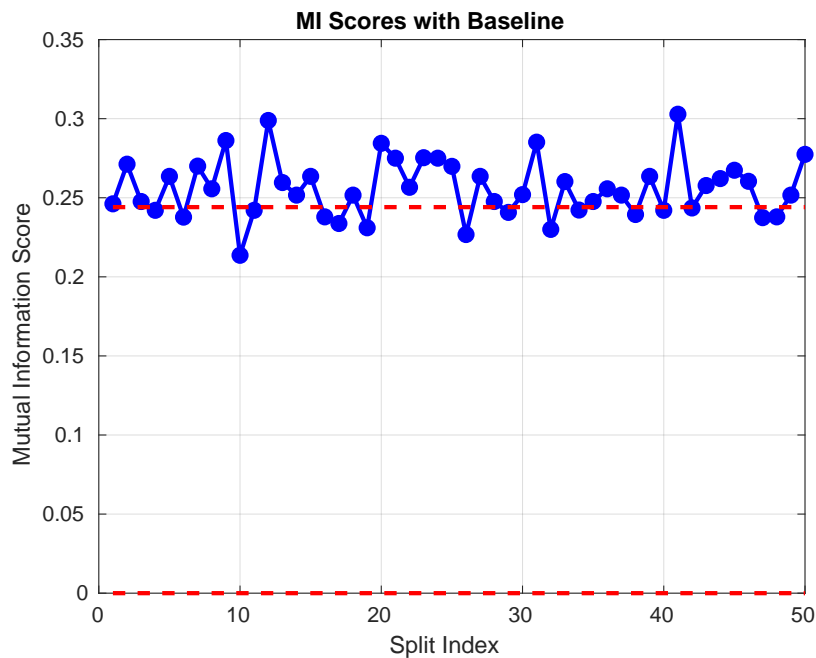


Figure 4.6: Sample Misclassification Case: Normal Desks

| |
|--|
| if glcm_correlation_mean > 0.947839 \wedge glcm_correlation_mean \leq 0.996404 \wedge hog_mean > 0.121639 \wedge hog_mean \leq 0.150611 then ‘Anomaly’ |
| if glcm_correlation_mean > 0.995864 then ‘Normal’ |
| if hog_mean > 0.139809 then ‘Normal’ |
| if glcm_correlation_mean \leq 0.953940 then ‘Normal’ |
| if glcm_correlation_mean \leq 0.973882 \wedge glcm_homogeneity_mean > 0.369726 then ‘Normal’ |
| if glcm_correlation_mean > 0.954613 \wedge hog_mean \leq 0.126453 then ‘Normal’ |
| if glcm_homogeneity_mean > 0.490572 then ‘Normal’ |
| if glcm_homogeneity_mean \leq 0.330240 then ‘Normal’ |
| if entropy_mean > 3.582243 \wedge glcm_homogeneity_mean > 0.397185 then ‘Normal’ |
| if entropy_mean \leq 3.394383 \wedge edges_mean > 6.940611 then ‘Normal’ |
| if glcm_energy_mean \leq 0.026352 then ‘Normal’ |
| if edges_mean > 15.674221 \wedge edges_mean \leq 15.948441 then ‘Anomaly’ |
| if glcm_contrast_mean > 187.486351 \wedge glcm_contrast_mean \leq 215.666177 then ‘Anomaly’ |

Table 4.4: Rules generated with Complete Feature Set

The rules generated using the complete feature set, as presented in Table 4.4, highlight the influence of HOG and GLCM features in the classification process. Observing these rules, it becomes evident that certain feature sets (4.7) contribute more significantly to the decision-making process. This prompted further investigation into whether removing HOG and GLCM correlation features would impact the model’s ability to differentiate anomalies. The motivation behind Experiment 2 was to evaluate the system’s performance when these features were excluded, aiming to determine if a more compact feature set could still achieve effective anomaly detection.

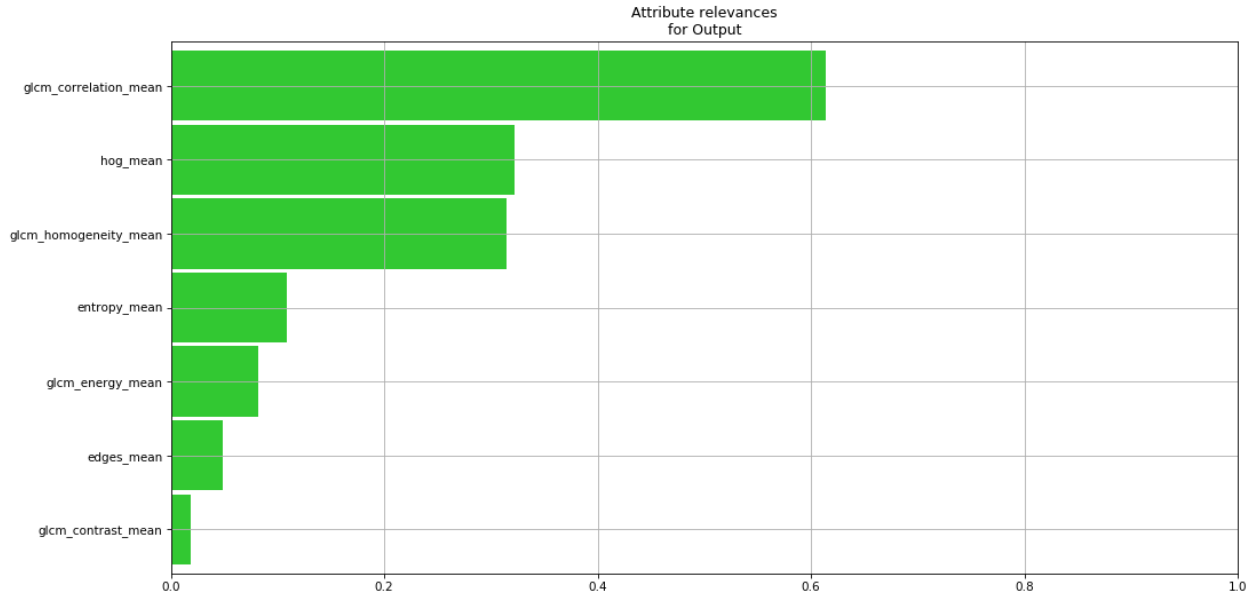


Figure 4.7: Experiment 1 Feature Ranking (all features)

4.5.2 Experiment 2: Synthetic images generated excluding dominant features.

For this experiment, the dominant features, which are HOG and correlation, as shown in the feature ranking in Figure 4.7, were removed to test the adaptability of the rules without these features. This resulted in 27 rules, with the remaining features having a balanced feature ranking.

4.5 Results

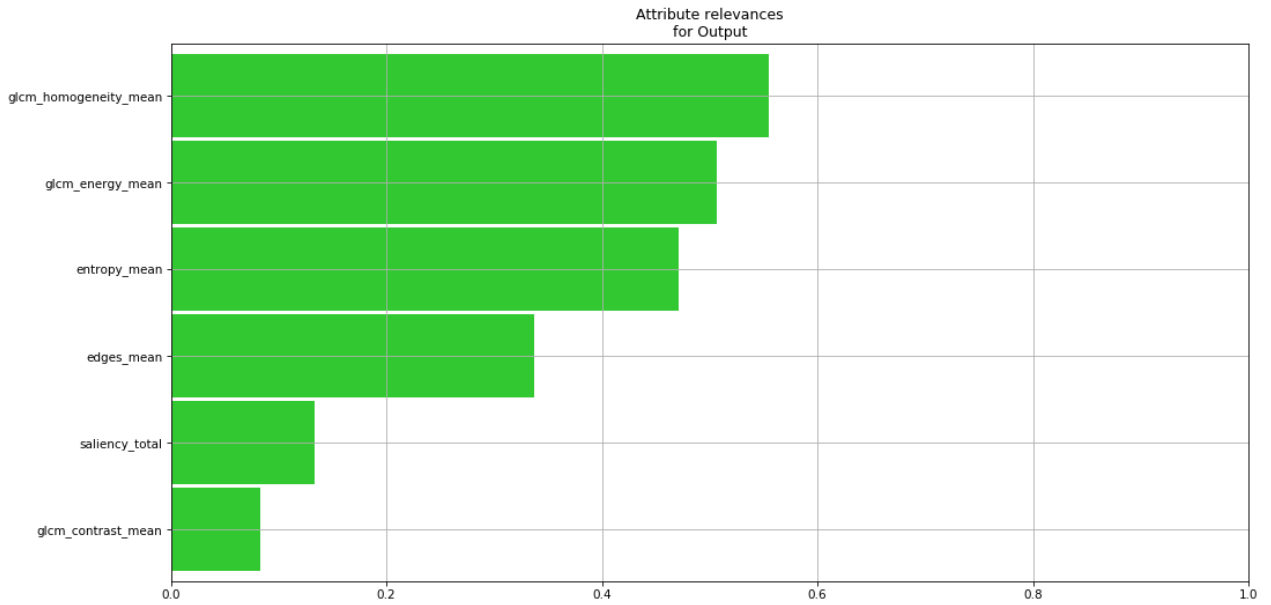


Figure 4.8: Experiment 2 Feature ranking (features without HOG and GLCM correlation)

The same procedure as in Experiment 1(4.5.1) was followed to generate the baselines and the results given below.

| | $W\mu I$ | l_1 | l_2 |
|----------|------------|-------------|------------|
| Baseline | [0,0.2872] | [0, 1.4333] | [0,0.4453] |

Table 4.5: Baseline Results for Experiment 2: Weighted Mutual Information and Norms.

| Anomaly | $W\mu I$ | l_1 | l_2 | FPR | FNR | Decision |
|-------------------|------------------|------------------|------------------|-----|-----|----------|
| Box | [0.3921, 0.5284] | [2.2507, 3.0520] | [0.6726, 0.9232] | - | 0 | OoD |
| Cable | [0.3944, 0.5121] | [3.0647, 3.6467] | [0.9555, 1.1504] | - | 0 | OoD |
| Object on Robot | [0.3821, 0.4946] | [2.8923, 3.7420] | [1.1327, 1.2707] | - | 0 | OoD |
| Water | [0.3501, 0.4433] | [2.5207, 3.0080] | [0.9101, 1.0655] | - | 0 | OoD |
| Sawdust | [0.3986, 0.5233] | [3.5796, 4.1750] | [1.1929, 1.3908] | - | 0 | OoD |
| Normal 2 | [0.4100, 0.5358] | [2.7322, 3.6693] | [0.7905, 1.1402] | 0 | - | ID |
| Normal 4 | [0.5087, 0.6333] | [2.8328, 3.8694] | [0.9905, 1.2406] | 1 | - | OoD |
| Normal Desks | [0.4100, 0.5360] | [2.7329, 3.6695] | [0.6907, 1.1308] | 1 | 0 | OoD |
| Object on Robot 2 | [0.3222, 0.4107] | [2.0100, 2.7760] | [0.6843, 0.9010] | - | 0 | OoD |

Table 4.6: Experiment 2 Anomaly Detection Results without highly attributed features

The results indicate that the model effectively detects all anomalies. However, a closer inspection of the results reveals a notable drawback: the model misclassifies certain normal instances as OoD, leading to an increase in the FPR for normal samples.

This misclassification suggests that the model is overfitting to the learned rules, becoming overly sensitive to minor variations in normal data. Since the model is trained on a large number of rules, it establishes stricter decision boundaries, which results in classifying normal conditions that slightly deviate from the training distribution as anomalies. This is problematic in real-world applications where very small variations in normal conditions should not be flagged as OoD.

One possible reason for this behavior is that the feature space used for classification may contain redundant or highly correlated attributes that the model relies on too heavily. These features may introduce unnecessary complexity, causing the model to enforce rigid classifications even when the variations are within acceptable limits for normal data.

4.5.3 Experiment 3: Synthetic images generated without corrected covariance matrix.

In this experiment, the model was tested on synthetic data generated without considering the covariance matrix, that is, disregarding the natural interdependencies between features found in real-world data. As defined in Equation 3.10, the covariance matrix Σ captures both the individual feature variances and the correlations between features, ensuring that synthetic data accurately reflects the

relationships observed in actual datasets. This experiment was conducted to evaluate whether incorporating feature dependencies through the covariance matrix adds significant value and impacts the model’s performance in detecting OoD cases.

| | $W\mu I$ | l_1 | l_2 |
|----------|---------------|---------------|---------------|
| Baseline | [0.0, 0.4235] | [0.0, 0.7434] | [0.0, 0.2642] |

Table 4.7: Baseline Results for Experiment 3: Weighted Mutual Information and Norms.

| Anomaly | $W\mu I$ | l_1 | l_2 | FPR | FNR | Decision |
|-------------------|------------------|------------------|------------------|-----|-----|----------|
| Box | [0.1165, 0.2638] | [0.5833, 0.6424] | [0.3047, 0.3835] | - | 1 | ID |
| Cable | [0.1120, 0.2846] | [0.3543, 0.2392] | [0.2104, 0.2489] | - | 1 | ID |
| Object on Robot | [0.3185, 0.4609] | [0.5204, 0.9785] | [0.3155, 0.4133] | - | 0 | OoD |
| Water | [0.2503, 0.4133] | [0.5235, 0.9801] | [0.3210, 0.4127] | - | 0 | OoD |
| Sawdust | [0.4086, 0.5233] | [0.6104, 0.9751] | [0.3017, 0.3978] | - | 1 | ID |
| Normal 2 | [0.3100, 0.3358] | [0.4000, 0.4700] | [0.1600, 0.2200] | 0 | - | ID |
| Normal 4 | [0.4032, 0.5439] | [0.3245, 0.4132] | [0.2323, 0.2450] | 0 | - | ID |
| Normal Desks | [0.4100, 0.5360] | [0.4300, 0.5000] | [0.1800, 0.2500] | 1 | - | OoD |
| Object on Robot 2 | [0.3222, 0.4147] | [0.5170, 0.9760] | [0.3086, 0.4049] | - | 0 | OoD |

Table 4.8: Experiment 3 Anomaly Detection Results without corrected covariance matrix.

By excluding this correlation structure, the model’s performance in this experiment is significantly impacted. The absence of the covariance matrix has resulted in a larger baseline range for both $W\mu I$ (0.4235) and l_1 norms (0.7434), weakening the distinction between in-distribution and out-of-distribution cases. As a result, anomalies such as *Sawdust* and *Water*, which were previously correctly classified as OoD, are now misclassified as *ID*, indicating a loss of sensitivity to meaningful deviations.

Furthermore, the misclassification of *Normal Desks* as OoD highlights the model’s increased susceptibility to minor fluctuations rather than true structural differences. Instead of capturing meaningful feature variations, the model establishes unstable decision boundaries, leading to both false positives and false negatives. Without the covariance matrix, the feature relationships essential for

robust anomaly detection are lost, reducing the reliability of the method for real-world applications.

4.5.4 Experiment 4: Evaluation of Anomaly Detection with Multiple Operational Splits ($N_{op} > 1$)

In this experiment, three operational splits were considered, and mutual information and norms were calculated across the splits. The final decision for each operational split was determined by majority voting, where the decision was based on whether more than half of the training splits recognized the operational split as OoD. The baseline values remained the same as in Experiment 1 (4.5.1), as the only modification was the number of operational splits, not the underlying training data. This experiment aimed to evaluate the impact of using multiple operational splits on the final classification decision.

| | $W\mu I$ | l_1 | l_2 |
|----------|---------------|---------------|---------------|
| Baseline | [0.0, 0.2141] | [0.0, 0.2082] | [0.0, 0.3013] |

Table 4.9: Baseline Results for Experiment 4: Weighted Mutual Information and Norms.

| Anomaly | $W\mu I$ | l_1 | l_2 | FPR | FNR | Decision |
|-------------------|------------------|------------------|------------------|--------|--------|----------|
| Box | [0.2073, 0.3021] | [0.4832, 0.9422] | [0.3047, 0.3835] | - | 0 | OoD |
| Cable | [0.2099, 0.3509] | [0.2634, 0.4293] | [0.3203, 0.3489] | - | 33.33% | OoD |
| Object on Robot | [0.4581, 0.4709] | [0.5204, 0.9785] | [0.3155, 0.4133] | - | 0 | OoD |
| Water | [0.5146, 0.5516] | [0.5235, 0.9801] | [0.3210, 0.4127] | - | 0 | OoD |
| Sawdust | [0.4783, 0.5080] | [0.5104, 0.9751] | [0.3017, 0.3978] | - | 0 | OoD |
| Normal 2 | [0.1103, 0.2135] | [0.1976, 0.1700] | [0.1694, 0.2216] | 0 | - | ID |
| Normal 4 | [0.1204, 0.1439] | [0.1200, 0.1900] | [0.1750, 0.2450] | 0 | - | ID |
| Normal Desks | [0.1364, 0.2664] | [0.1300, 0.1500] | [0.1800, 0.2500] | 66.67% | - | OoD |
| Object on Robot 2 | [0.3222, 0.4147] | [0.5170, 0.9760] | [0.3086, 0.4049] | - | 0 | OoD |

Table 4.10: Experiment 4 Anomaly Detection Results with Complete Feature Set

Experiment 4 brought some interesting findings. By introducing multiple operational splits, the model became better at capturing the variability in the data, resulting in more consistent and reliable anomaly detection. For example,

Normal Desks, which was classified as OoD with a very high FPR of 1 (100%) in Experiment 1, showed a much more balanced FPR of **66.67%** in this experiment. This shift indicates that the model, with the help of multiple splits, became less sensitive to small fluctuations, leading to more stable decision-making overall.

Additionally, while multiple splits provided greater stability overall, it might also cause anomalies close to the baseline to be misclassified. This can be seen with the Cable anomaly, where the model, influenced by the majority of splits, became more inclined to misclassify some anomalies as in-distribution, potentially introducing false positives. Specifically, for the Cable anomaly, the FPR was reported as **33.33%**, indicating that, despite being an anomaly, it was incorrectly identified as in-distribution in one out of the three splits.

However, the use of multiple splits does come with a trade-off: an increase in computational complexity. The model now has to process more data and make decisions based on a larger set of operational splits, which requires additional computational resources. This trade-off between accuracy and computational cost becomes particularly relevant in real-world applications, where large datasets and scalability are often concerns.

4.5.5 Summary of the results

The experiments provide several important insights regarding the model’s performance and the impact of different features on anomaly detection. The system performed best when all selected features were used, with a single operational split and the covariance matrix included for realistic perturbations. This configuration allowed the model to accurately detect anomalies, except for a variation of the normal (in-distribution) data labeled "Normal_desks," which was incorrectly classified as OoD. Increasing the operational split size in this setting resulted in largely consistent results. However, some borderline cases, such as Normal Desks, were misclassified due to small fluctuations near the threshold. This indicates that while the model was able to detect anomalies, fine-tuning the decision boundaries might be needed to reduce misclassifications in borderline cases.

In **Experiment 2**, where dominant features like HOG and GLCM were excluded, the model was still capable of detecting anomalies. However, the removal of these features resulted in an increased false positive rate (FPR) for normal data. This suggests that removing these dominant features made the model overly sensitive to minor variations, leading to the misclassification of normal data as OoD. The dominant features (HOG and GLCM) might be important because they capture key structural or texture-based information, which is crucial for distinguishing between ID and OoD data. Retaining these features ensures the model can leverage the full spectrum of relevant data and make more accurate predictions.

In **Experiment 3**, the exclusion of the covariance matrix weakened the model’s ability to differentiate between ID and OoD cases. Without the covariance matrix, the model lost the relationships between features, which are important for identifying meaningful deviations from normal patterns. This reinforces the importance of considering feature dependencies to improve model sensitivity and accuracy.

Finally, **Experiment 4**, which introduced multiple operational splits, provided greater stability in decision making, reducing the FPR for *Normal_Desks*. However, this also led to some anomalies, such as the *Cable* anomaly, being misclassified as in-distribution, as the model relied on majority voting from the splits. Additionally, introducing multiple splits increased the computational complexity, requiring more resources for processing. This highlights the trade-off between stability and computational cost. While multiple splits enhance reliability, their use in real-world applications should be carefully considered, especially in scenarios with large datasets.

In conclusion, the key recommendations are to retain the dominant features which are HOG and GLCM, as they provide valuable information for anomaly detection. Additionally, incorporating feature dependencies through the covariance matrix is crucial for improving model performance. While multiple operational splits enhance stability, the associated computational cost should be considered based on the application’s requirements.

Chapter 5

Conclusion and Future works

This thesis aimed to advance rule-based out-of-distribution (OoD) detection by incorporating unsupervised techniques, building upon a previous work [14]. The motivation stemmed from the limitations of existing methods that rely on labeled datasets, which are costly and impractical for complex data types. By moving beyond labeled data, the proposed approach enhances the adaptability and scalability of OoD detection systems, making them more suitable for real-world applications.

To achieve this, we introduced synthetic data generation methods that leverage perturbations of in-distribution data while preserving statistical properties such as Gaussian distributions and feature correlations. By adjusting the variance and covariance structures, the generated synthetic samples effectively simulate a wide range of OoD scenarios, allowing the model to better generalize to unfamiliar inputs.

The rule-based OoD classification process integrates multiple evaluation methods, including Weighted Mutual Information ($W\mu I$), l1-norm, and l2-norm, into a unified decision-making framework. By establishing baseline intervals for each metric through rule hits tables, the approach ensures a structured comparison between training and operational data. Each operational split is then evaluated against these baselines, with deviations signaling potential OoD instances. The final decision aggregates results from all three metrics, if any of them classify the dataset as OoD, it is labeled as "OoD" ensuring a rigorous detection process.

The experimental results demonstrated the effectiveness of this approach. When trained on synthetic data with all selected features taking their correlations into consideration, the model successfully detected all anomalous test cases, highlighting the importance of the selected features. Retaining dominant features such as HOG and GLCM proved crucial, as their removal led to increased false positive rates, indicating that these features capture essential structural and texture-based information for distinguishing between ID and OoD data. Fur-

thermore, the inclusion of the covariance matrix reinforced the model’s ability to identify meaningful deviations, confirming that feature dependencies play a vital role in realistic synthetic data generation.

A notable trade-off was observed in the use of multiple operational splits. While this approach enhanced stability and reduced false positives for borderline cases, it also introduced computational complexity. This trade-off suggests that the selection of operational split strategies should be tailored to the specific requirements of the application, balancing stability and efficiency.

One limitation in this approach lies in the selection of the variance percentages used to perturb the features. These percentages were chosen to achieve the minimum perturbation required to simulate OoD scenarios. However, this choice lacks scientific backing and is based on heuristic judgments. Future work should focus on empirically determining the optimal variance percentages, potentially using statistical methods or data-driven approaches to ensure that the perturbations closely mimic real-world anomalies.

5.1 Future Work

While the proposed approach has demonstrated promising results, several areas remain open for further exploration:

- **Empirical Optimization of Perturbation Factors:** Investigating more robust statistical methods to define the optimal variance percentages for perturbing features to better simulate OoD scenarios.
- **Optimization for Real-Time Applications:** Given the computational cost of multiple operational splits, optimizing the approach for real-time systems would be a valuable direction for future work.
- **Expanding to More Complex Data Modalities:** The model’s performance on image data suggests its adaptability, but further validation on other complex modalities such as time-series and multi-modal sensor data could broaden its applicability.

Appendix A

Appendix

A.1 Feature Extraction and Synthetic Data Generation

This appendix includes the code used for feature extraction and synthetic data generation in the experiments. It outlines the process of loading images, extracting relevant features, and generating synthetic data based on those features.

A.1.1 Relevant Imports

The Athec library [16] was utilized to extract edge and saliency features from in-distribution images. For HOG and GLCM feature extraction, the scikit-image library was used.

```
1 import os
2 import cv2
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 from scipy.stats import norm
8 from skimage.exposure import exposure
9 from skimage.feature import hog, graycomatrix, graycoprops
10 from skimage.color import rgb2gray
11 from skimage.filters.rank import entropy
12 from skimage.morphology import disk
13 from athec.athec import saliency, misc, edge
```

A.1.2 Feature Extraction

The feature extraction process begins with HOG (Histogram of Oriented Gradients), which is applied to grayscale images to capture texture and shape information. Following this, the GLCM (Gray-Level Co-occurrence Matrix) is used to compute texture properties, such as contrast and dissimilarity. Entropy, a measure of randomness, is calculated using a local entropy filter, while a saliency map is generated using the `saliency.compute_saliency()` function to highlight visually important regions in the image. Edge features are detected using the Canny edge detector to identify prominent edges. These extracted features are then aggregated and stored in a dictionary format, with the corresponding image filename. To ensure numerical consistency, the mean values of entropy, saliency, and edge features are calculated. Finally, the extracted features are converted into a pandas DataFrame and saved as a CSV file in the specified output folder.

```
1 # Extract Features and Save
2 def extract_features_and_save(images, image_files,
3     saliency_folder, output_folder):
4     """
5     Extract features from the given images and save them to the
6     output folder.
7
8     Parameters:
9     - images: List of images
10    - image_files: Corresponding list of image filenames
11    - saliency_folder: Folder containing saliency images
12    - output_folder: Folder to save the extracted features
13
14    Returns:
15    - features: DataFrame of extracted features
16    """
17    features = []
18
19    for i, image in enumerate(images):
20        # Extract HOG features
21        gray = rgb2gray(image)
22        fd, hog_image = hog(gray, visualize=True)
23
24        # Extract GLCM features
25        glcm = graycomatrix(gray, [1], [0], symmetric=True,
26            normed=True)
27        contrast = graycoprops(glcm, 'contrast')[0, 0]
28        dissimilarity = graycoprops(glcm, 'dissimilarity')[0, 0]
29
30        # Extract Entropy features
31        entropy_value = entropy(gray, disk(3))
```

A.1 Feature Extraction and Synthetic Data Generation

```
29
30     # Extract Saliency features (using a hypothetical
31     saliency module)
31     saliency_map = saliency.compute_saliency(image)
32
33     # Extract Edge features
34     edges = canny(gray)
35
36     # Combine features into a list
37     feature_vector = {
38         "image_file": image_files[i],
39         "hog_feature": fd[0], # Assuming the first element
40         is the feature vector
41         "contrast": contrast,
42         "dissimilarity": dissimilarity,
43         "entropy": np.mean(entropy_value),
44         "saliency": np.mean(saliency_map),
45         "edges": np.mean(edges)
46     }
47     features.append(feature_vector)
48
49     # Convert the list of features into a DataFrame
50     features_df = pd.DataFrame(features)
51
52     # Save to CSV (or any other output format)
53     features_df.to_csv(os.path.join(output_folder, "
54         extracted_features.csv"), index=False)
55
56     return features_df
```

A.1.3 Generating Synthetic Data

The code shows the multivariate normal distribution used to generate synthetic out-of-distribution (OoD) data. To create synthetic samples, a mean vector is calculated by adjusting the observed feature values with the corresponding standard deviations and a specified deviation factor. The covariance matrix, which captures the relationships and dependencies between features, is derived from the correlation matrix. This ensures that the synthetic data maintains the inherent feature correlations. Once the synthetic samples are generated, they are clipped to ensure all feature values remain non-negative. The final synthetic data is returned in a structured format with the adjusted and perturbed feature values, simulating plausible anomalies while preserving the original relationships among the features.

A.1 Feature Extraction and Synthetic Data Generation

```
1
2 def calculate_variance(value, percentage):
3     """Calculate the variance based on the percentage deviation.
4     """
5     return (value * percentage) ** 2
6
7 def generate_synthetic_data(features, correlation_matrix,
8     deviation_factor=2):
9     """Generates synthetic OOD data by pushing values beyond the
10     normal distribution."""
11     synthetic_features = []
12
13     # Variance factors (as fractions) for each feature
14     variance_dict = {
15         'entropy_mean': 0.10, # 10% deviation
16         'edges_mean': 0.30, # 30% deviation
17         'saliency_total': 0.15, # 15% deviation
18         'glcm_contrast_mean': 0.20, # 20% deviation
19         'glcm_correlation_mean': 0.12, # 12% deviation
20         'glcm_energy_mean': 0.18, # 18% deviation
21         'glcm_homogeneity_mean': 0.25, # 25% deviation
22         'hog_mean': 0.25 # 25% deviation
23     }
24
25     feature_names = list(variance_dict.keys())
26
27     for feature in features:
28         synthetic_feature = {'file_name': feature['file_name']}
29
30         # Convert the feature values to a numpy array
31         values = np.array([feature[key] for key in feature_names
32             ])
33
34         # Generate variances based on the specified variance
35         # factors
36         variances = np.array(
37             [calculate_variance(value, variance_dict[key]) for
38                 key, value in zip(feature_names, values)]
39         )
40
41         # Calculate standard deviations
42         std_dev = np.sqrt(variances)
43
44         # Generate correlated synthetic samples (deviating beyond
45         # 2 standard deviations)
46         random_samples = np.random.multivariate_normal(
47             values + deviation_factor * std_dev,
```

```
42         correlation_matrix * np.diag(std_dev ** 2), 1
43     ).flatten()
44     # Clip to ensure values are non-negative
45     random_samples = np.clip(random_samples, 0, None)
46
47     # Assign to synthetic feature
48     for key, synthetic_value in zip(feature_names,
49                                     random_samples):
50         synthetic_feature[key] = synthetic_value
51
52     synthetic_features.append(synthetic_feature)
53
54     return synthetic_features
```

A.2 Out of distribution detection

This MATLAB code is designed for **Out-of-Distribution (OoD) detection**, a crucial task in machine learning and anomaly detection. The goal of the code is to determine whether the operational (test) data is from the same distribution as the training data or if it falls outside that distribution (OoD). It does so by analyzing key statistical properties of the data, particularly focusing on **Mutual Information (MI)** and **norms (L1 and L2 norms)**. Here's a high-level breakdown of the approach:

- **Data Loading:** The code loads two datasets:
 - **Training data (dataTR):** Used to learn the patterns or characteristics of normal data.
 - **Operational data (dataOP):** Represents new data that needs to be evaluated for whether it belongs to the same distribution as the training data or is anomalous.
- **Baseline Computation:** It computes baseline values (e.g., statistical properties like mean and standard deviation) from the training data. These baselines help in assessing whether the operational data aligns with the normal patterns.
- **Mutual Information (MI) Calculation:** MI is used to evaluate the dependence between features in the training and operational datasets. A large difference in MI between the two datasets can indicate a potential OoD scenario.

A.2 Out of distribution detection

- **Norm Calculation (L1 and L2):** L1 and L2 norms measure the distance between the data points. By computing these norms for both datasets, the code checks if the operational data deviates significantly from the training data, which could signal an anomaly.
- **Final Decision:** Based on the MI and norm calculations, the code makes a final decision. If the operational data significantly deviates from the training data (as indicated by MI and norm flags), it is classified as **Out-of-Distribution (OoD)**; otherwise, it is classified as **In Distribution**.

In summary, the code integrates statistical methods to identify whether new data (operational data) comes from the same distribution as the training data, a key task for detecting anomalies or outliers in real-world machine learning applications.

```
1 %% Settings
2
3 % number of splits for training and operational datasets
4 n_tr = 50;
5 % usually, we always consider n_op = 1
6 n_op = 1;
7
8 % Boolean value to set the kind of mutual information (MI) used.
9 % for weighted MI, set weighted = true; for MI, set weighted =
   false
10 weighted = true;
11
12 % set to true to use l2
13 useL2 = true;
14
15 % boolean value: true if the test aims at finding false positives
   , false
16 % otherwise
17 compute_fpr = false;
18
19 %% Load the data
20 % dataTR: training domain
21 % dataOP: operational domain
22 if compute_fpr
23     % to compute false positives, we have to apply the ODD method
       by setting the data coming from the
24     % training domain as operational; if the ODD method is robust
       to FPs, we expect that
25     % such data are recognized as In distribution.
26     dataTR = table2array(readtable("MatlabCode\
       extracted_features_training_ruleHits_allFeatures.xlsx"));
27     dataOP = table2array(readtable("MatlabCode\
       Box_Anomaly_allFeatures.xlsx"));
```

A.2 Out of distribution detection

```
28 else
29     % to recognize the Out of Distribution data and compute the
        FNR, we a
30     dataTR = table2array(readtable("MatlabCode\
        extracted_features_training_ruleHits_allFeatures.xlsx"));
31     dataOP = table2array(readtable("MatlabCode\
        Box_Anomally_allFeatures.xlsx"));
32 end
33
34 %% Find the baselines
35
36 [mibaseline,norm_baseline] = ComputeBaselines(dataTR,n_tr,
        weighted,usel2);
37
38 %% Mutual information
39
40 [miflag,miOpRange,tp2_mi] = MI_flag(dataTR,dataOP,n_op,n_tr,
        mibaseline,weighted,compute_fpr);
41
42 %% l1 Norm
43
44 [norm1_flag,norm1OpRange,tp2_n1] = norms_flag(dataTR,dataOP,n_op,
        n_tr,norm_baseline,false,compute_fpr);
45
46 %% l2 norm
47
48 [norm2_flag,norm2OpRange,tp2_n2] = norms_flag(dataTR,dataOP,n_op,
        n_tr,norm_baseline,usel2,compute_fpr);
49
50 %% Final decision
51
52 if miflag || norm1_flag || norm2_flag
53     disp("Out of Distribution");
54 else
55     disp("In distribution");
56 end
```

A.2.1 Weighted Mutual Information Calculation

```
1 function [miflag, miOpRange, MImatrix, tp2] = MI_flag(dataTR,
        dataOP, n_op, n_tr, mibaseline, weighted, compute_fpr)
2
3 % counter of the actual operational splits correctly recognized
        as OoD
4 % (if at least half of the training number of splits)
5 tp2 = 0;
6 min_tr = mibaseline(1);
```

A.3 Process flow of rule generation in Rulex

```
7 max_tr = mibaseline(2);
8 MImatrix = zeros(n_tr, n_op);
9
10 % iterate over operational columns
11 for k = 1:n_op
12     % counter of all splits correctly recognized as OoD (i.e.,
13     % true positives)
14     % for each operational, this counter is initialized to 0
15     tp = 0;
16     % iterate over training columns
17     for i = 1:n_tr
18         % define the couple of vectors
19         hitx = dataTR(:, i);
20         hity = dataOP(:, k);
21         % compute the mutual information
22         MImatrix(i, k) = mutual_info(hitx, hity, weighted);
23         disp(['MI(', num2str(i), ', ', num2str(k), '): ', num2str(
24             MImatrix(i, k))]);
25         % check if the computed mi is outside the baseline
26         if MImatrix(i, k) < min_tr || MImatrix(i, k) > max_tr
27             % increment tp
28             tp = tp + 1;
29         end
30     end
31     % check if the MI for k is out of the baseline for more than
32     % half of the number of training splits
33     if tp > fix(n_tr / 2)
34         tp2 = tp2 + 1;
35         if ~compute_fpr
36             miflag = true;
37         end
38     else
39         if ~compute_fpr
40             miflag = false;
41         end
42     end
43 end
44 min_op = min(MImatrix(:));
45 max_op = max(MImatrix(:));
46 miOpRange = [min_op, max_op];
47 end
```

A.3 Process flow of rule generation in RuleX

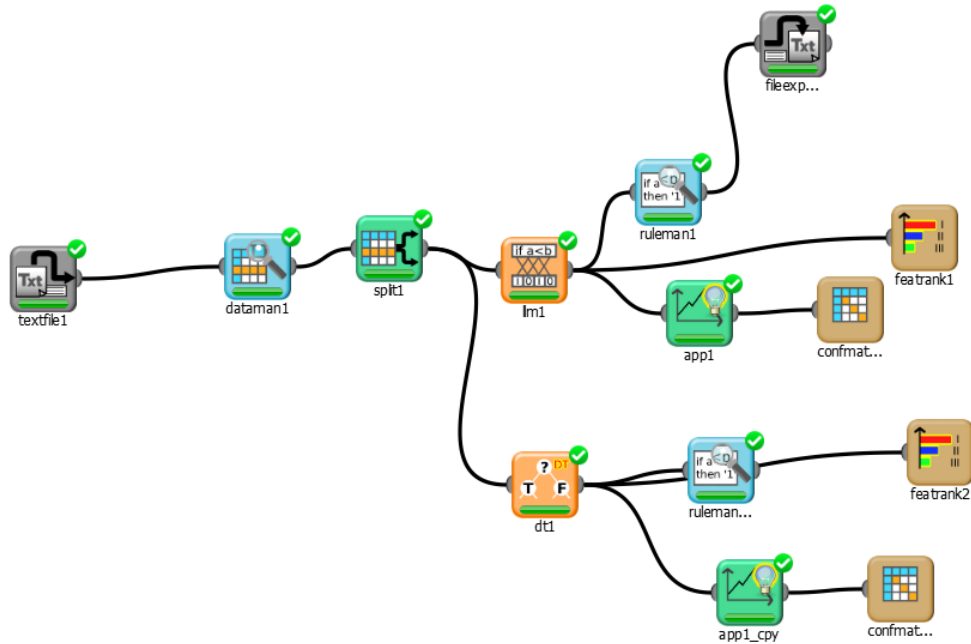


Figure A.1: RuleX Flow

A.3 Process flow of rule generation in RuleX

1. The merged file is imported with an **Import from Text File** task.
2. The **Data Manager Task** helps visualize the data.
3. The dataset is split into test and training sets with a **Split Data Task**.
4. Rules were generated with the **Classification LLM** and with a decision tree for comparison purposes.
5. The **Rule Manager** task is used to analyze the generated rules.
6. The **Feature Ranking Task** allows us to easily visualize the most important attribute in determining the output.
7. Using the **Export File** task, the rules are exported in CSV format.

References

- [1] D. M. Hawkins. *Identification of Outliers*. Monographs on Applied Probability and Statistics. Dordrecht, Netherlands: Springer Netherlands, 1980. ISBN: 978-94-015-3994-4. DOI: [10.1007/978-94-015-3994-4](https://doi.org/10.1007/978-94-015-3994-4).
- [2] Charu C. Aggarwal. *Outlier Analysis*. 2nd Edition. Cham, Switzerland: Springer International Publishing, 2017. ISBN: 978-3-319-47578-3. DOI: [10.1007/978-3-319-47578-3](https://doi.org/10.1007/978-3-319-47578-3).
- [3] Frank E. Grubbs. “Procedures for Detecting Outlying Observations in Samples”. In: *Technometrics* 11 (1969), pp. 1–21. URL: <https://api.semanticscholar.org/CorpusID:120469443>.
- [4] Mohammadreza Salehi et al. *A Unified Survey on Anomaly, Novelty, Open-Set, and Out-of-Distribution Detection: Solutions and Future Challenges*. 2022. arXiv: [2110.14051](https://arxiv.org/abs/2110.14051) [cs.CV]. URL: <https://arxiv.org/abs/2110.14051>.
- [5] Hoya012. *Awesome Anomaly Detection*. Accessed: 2024-12-05. 2024. URL: <https://github.com/hoya012/awesome-anomaly-detection/tree/master>.
- [6] Zesheng Hong et al. *Out-of-distribution Detection in Medical Image Analysis: A survey*. 2024. arXiv: [2404.18279](https://arxiv.org/abs/2404.18279) [cs.CV]. URL: <https://arxiv.org/abs/2404.18279>.
- [7] Xiaowei Huang et al. *A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability*. 2020. arXiv: [1812.08342](https://arxiv.org/abs/1812.08342) [cs.LG]. URL: <https://arxiv.org/abs/1812.08342>.
- [8] L. Lankewicz and M. Benard. “Real-time anomaly detection using a non-parametric pattern recognition approach”. In: *Proceedings Seventh Annual Computer Security Applications Conference*. 1991, pp. 80–89. DOI: [10.1109/CSAC.1991.213016](https://doi.org/10.1109/CSAC.1991.213016).

REFERENCES

- [9] Vikash Sehwasg, Mung Chiang, and Prateek Mittal. “{SSD}: A Unified Framework for Self-Supervised Outlier Detection”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=v5gjXpmR8J>.
- [10] Simon Hawkins et al. “Outlier detection using replicator neural networks”. In: *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2002, pp. 170–180.
- [11] Dan Hendrycks and Kevin Gimpel. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. 2018. arXiv: [1610.02136](https://arxiv.org/abs/1610.02136) [cs.NE]. URL: <https://arxiv.org/abs/1610.02136>.
- [12] Abhijit Bendale and Terrance E. Boult. “Towards Open Set Deep Networks”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1563–1572. DOI: [10.1109/CVPR.2016.173](https://doi.org/10.1109/CVPR.2016.173).
- [13] Marco Muselli. “Switching Neural Networks: A New Connectionist Model for Classification”. In: *Neural Nets*. Ed. by Bruno Apolloni et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 23–30. ISBN: 978-3-540-33184-1.
- [14] Giacomo De Bernardi et al. “Rule-Based Out-of-Distribution Detection”. In: *IEEE Transactions on Artificial Intelligence* 5.06 (June 2024), pp. 2627–2637. ISSN: 2691-4581. DOI: [10.1109/TAI.2023.3323923](https://doi.org/10.1109/TAI.2023.3323923). URL: <https://doi.ieeecomputersociety.org/10.1109/TAI.2023.3323923>.
- [15] Michael Yuhas and Arvind Easwaran. *Demo Abstract: Real-Time Out-of-Distribution Detection on a Mobile Robot*. 2022. arXiv: [2211.11520](https://arxiv.org/abs/2211.11520) [cs.R0]. URL: <https://arxiv.org/abs/2211.11520>.
- [16] Yilang Peng. “AtheC: A Python Library for Computational Aesthetic Analysis of Visual Media in Social Science Research”. In: *Computational Communication Research* 4.1 (May 2022), pp. 323–349. URL: <https://computationalcommunication.org/ccr/article/view/98>.
- [17] Keith Man and Javaan Chahl. “A Review of Synthetic Image Data and Its Use in Computer Vision”. In: *Journal of Imaging* 8.11 (2022). ISSN: 2313-433X. DOI: [10.3390/jimaging8110310](https://doi.org/10.3390/jimaging8110310). URL: <https://www.mdpi.com/2313-433X/8/11/310>.
- [18] Dario Mantegazza et al. “HazardsRobots: A dataset for visual anomaly detection in robotics”. In: *Data in Brief* 48 (2023), p. 109264. ISSN: 2352-3409. DOI: <https://doi.org/10.1016/j.dib.2023.109264>. URL: <https://www.sciencedirect.com/science/article/pii/S2352340923003839>.

REFERENCES

- [19] John Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: [10.1109/TPAMI.1986.4767851](https://doi.org/10.1109/TPAMI.1986.4767851).
- [20] Cronj. *Histogram of Oriented Gradients (HOG): Calculating Image Features*. Accessed: 2025-03-06. 2021. URL: <https://www.cronj.com/blog/histogram-of-oriented-gradients-hog-calculating-image-features/>.
- [21] Toto Haryanto et al. “Multipatch-GLCM for Texture Feature Extraction on Classification of the Colon Histopathology Images using Deep Neural Network with GPU Acceleration”. In: *Journal of Computer Science* 16 (Mar. 2020), pp. 280–294. DOI: [10.3844/jcssp.2020.280.294](https://doi.org/10.3844/jcssp.2020.280.294).
- [22] Patrick Schober, Christa Boer, and Lothar Schwarte. “Correlation Coefficients: Appropriate Use and Interpretation”. In: *Anesthesia Analgesia* 126 (Feb. 2018), p. 1. DOI: [10.1213/ANE.0000000000002864](https://doi.org/10.1213/ANE.0000000000002864).
- [23] Marco Muselli. “Switching neural networks: A new connectionist model for classification”. In: *Italian Workshop on Neural Nets*. Springer. 2005, pp. 23–30.
- [24] Stefano Parodi et al. “Identifying environmental and social factors predisposing to pathological gambling combining standard logistic regression and logic learning machine”. In: *Journal of Gambling Studies* 33 (2017), pp. 1121–1137.
- [25] Stefano Parodi et al. “Differential diagnosis of pleural mesothelioma using Logic Learning Machine”. In: *BMC bioinformatics* 16 (2015), pp. 1–10.
- [26] Stefano Parodi et al. “Logic Learning Machine and standard supervised methods for Hodgkin’s lymphoma prognosis using gene expression data and clinical variables”. In: *Health Informatics Journal* 24.1 (2018), pp. 54–65.
- [27] Marco Muselli. *Switching Neural Networks: A New Connectionist Model for Classification*. Jan. 2005.
- [28] Giacomo De Bernardi et al. “Rule-based out-of-distribution detection”. In: *arXiv preprint arXiv:2303.01860* (2023).
- [29] Jefkine Kafunah et al. “Out-of-Distribution Data Generation for Fault Detection and Diagnosis in Industrial Systems”. In: *IEEE Access* 11 (2023), pp. 135061–135073. DOI: [10.1109/ACCESS.2023.3337658](https://doi.org/10.1109/ACCESS.2023.3337658).
- [30] Chuan Guo et al. “On Calibration of Modern Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1321–1330. URL: <https://proceedings.mlr.press/v70/guo17a.html>.

REFERENCES

- [31] Jingkang Yang et al. *OpenOOD: Benchmarking Generalized Out-of-Distribution Detection*. 2022. arXiv: [2210.07242](https://arxiv.org/abs/2210.07242) [cs.CV]. URL: <https://arxiv.org/abs/2210.07242>.
- [32] Iacopo Masi et al. “Deep Face Recognition: A Survey”. In: Oct. 2018, pp. 471–478. DOI: [10.1109/SIBGRAP.2018.00067](https://doi.org/10.1109/SIBGRAP.2018.00067).
- [33] Greg Brockman et al. “Openai gym”. In: *arXiv preprint arXiv:1606.01540* (2016).