UNIVERSITY OF ŽILINA IN ŽILINA

FACULTY OF MANAGEMENT SCIENCE AND INFORMATICS

Towards Explainable Pattern Recognition

DISSERTATION THESIS

28360020233004

Study Program:	Applied Informatics
Field of Study:	Informatics
Workplace:	Department of Information Networks
	Faculty of Management Science and Informatics,
	University of Žilina in Žilina
Supervisor:	Prof. Ing. Martin Klimo, PhD.

Žilina, 2023 **Ing. Jaroslav Kopčan**

Declaration

I sincerely declare that this thesis named *Towards Explainable Pattern Recognition* has been composed solely by myself and according to the methodological guidelines of the university, as well as according to the professional advice of my supervisor. To the best of my knowledge, it has not been submitted, either in part or in whole, in any prior application for a degree. I affirm that I have cited and listed all the sources used in my research in the bibliography section. Moreover, I have actively applied the knowledge and expertise gained during my doctoral studies to enrich my research work.

Author signature: _____

Acknowledgements

Firstly, I would like to express my profound gratitude to my supervisor, Prof. Klimo, for his invaluable guidance, support, and remarkable patience throughout my PhD. study. Also, I would like to express my sincere gratitude to my girlfriend for her unwavering support and understanding. I am deeply grateful for her constant love which has helped me immensely. Lastly, I thank my colleagues and fellow PhD. students for their enthusiasm and camaraderie. They have made the study experience more enjoyable and inspirational.

Abstract

KOPČAN Jaroslav, Ing.: Towards Explainable Pattern Recognition. [Dissertation Thesis] University of Žilina in Žilina. Faculty of Management Science and Informatics. Faculty of Information Networks. - Thesis supervisor: prof. Ing. Martin Klimo,PhD. - Žilina: FRI ŽU, 2023, 162 pages.

Research in this dissertation hones in on the explainability of predictions obtained by deep learning classifiers. As the mining of input data may generate novel features that are not readily interpretable within the application domain, our main goal is to establish a system that provides explanations of the prediction and from those, help the end-user infer the interpretability of involved features. To sustain the high performance of a non-linear system, we concentrate on a post-hoc explanation strategy, which involves constructing an interpretable model based on fuzzy logic which provides explanations of the decision-making of the black box. To overcome the challenge of interpreting the vast number of non-linear functions, we decompose the DNN into two components: the feature extractor and the classifier. The exact interpretation of features is a tough task-dependent discipline that has the nature of scientific work. To facilitate this process, we introduce a tool to help researchers and users comprehend newly extracted features that may not be known in the specific application domain.

Following an evaluation of the related work in this field, we have concluded that the current explainability mechanism fails to consider the possibility of anomalies. This means the interpretation method may attempt to explain unexplainable inputs, and users remain uninformed of such misuse of recognizers. To safeguard users against these issues, we suggest integrating an anomaly detector into the explainability system. This work examines an innovative approach to anomaly detection that does not rely on measuring the similarity between the tested pattern and those in the training set as conventional methods. Instead, the proposed method aims to compare the underlying mechanisms responsible for forming the tested image with those used for forming images in the training set. By focusing on the creation mechanisms, the proposed method can more accurately identify anomalous patterns.

Keywords: explainability, interpretability, anomaly detection, deep generative modeling, deep learning, feature extraction, pattern recognition, fuzzy logic

Abstrakt

KOPČAN Jaroslav, Ing.: Vysvetliteľné Rozpoznávanie Vzorov. [Dizertačná práca] Žilinská univerzita v Žiline. Fakulta riadenia a informatiky. Katedra Informačných Sietí. - Vedúci dizertačnej práce: prof. Ing. Martin Klimo,PhD. - Žilina: FRI ŽU, 2023, 162 pages.

Výskum v tejto dizertačnej práci sa zameriava na vysvetliteľnosť predikcií nelineárnych klasifikátorov hlbokého učenia. Keďže systém môže zo vstupných dát extrahovať príznaky, ktoré nie sú ľahko interpretovateľné v rámci aplikačnej domény, hlavným cieľom je vytvoriť systém, ktorý poskytuje vysvetlenia predikcií. Z nich má následne používateľ možnosť odvodiť interpretovateľnosť príslušných príznakov. Pre zachovanie výkonnosti nelineárnych systémov sa sústredíme na post-hoc prístup, ktorá znamená vytvorenie interpretovateľného modelu založeného na fuzzy logike. Logický model poskytuje vysvetlenie predikcií čiernej skrinky. Aby sme prekonali problém interpretácie obrovského množstva nelineárnych funkcií, konceptuálne delíme nelinárny klasifikátor na dve časti: extraktor príznakov a klasifikátor. Presná interpretácia príznakov je náročná disciplína, ktorá má charakter vedeckej práce. Na uľahčenie tohto procesu predstavujeme nástroj, ktorý pomáha výskumníkom a používateľom porozumieť novo extrahovaným príznakov, ktoré nemusia byť v konkrétnej aplikačnej doméne známe.
Na základe hodnotenia súčasného stavu vysvetliteľnosti konštatujeme, že sa neberie do úvahy možnosť dátových vstupov, ktoré sú anomálne voči trénovaciemu dátovému súboru. Teda systém vysvetliteľnosti sa bude snažiť vysvetliť nevysvetliteľné. O takomto nesprávnom používaní rozpoznávačov nie sú používatelia poučení, preto by bol detektor anomálií vhodnou súčasťou. V práci navrhujeme detektor anomálií ako možnú súčasť vysvetliteľného konceptu. Pre detekciu anomálií je skúmaných viacero metód, ktoré nehodnotia podobnosť testovaného vzoru so vzormi v trénovacej množine, ale sa snažia porovnávať mechanizmy vzniku testovaných obrazov s mechanizmom vzniku obrazov v trénovacom súbore.

Kľúčové slová: vysvetliteľnosť, interpretovateľnosť, detekcia anomálií, hlboké generatívne modelovanie, hlboké učenie, fuzzy logika, extrakcia príznakov

List of Abbreviations

AE Autoencoder.

AI Artificial Intelligence.

ARIMA Autoregressive Integrated Moving Average.

CART Classification and Regression Tree.

cGAN Conditional Generative Adversarial Network.

CNN Convolutional Neural Network.

DAD Distributed Anomaly Detection.

DenseNet Densely Connected Convolutional Network.

DGM Deep Generative Modeling.

DL Deep Learning.

DNF Disjunctive Normal Form.

DNN Deep Neural Net.

DRL Deep Reinforcement Learning.

ELU Exponential Linear Unit.

FCNN Fully Connected Neural Network.

FGSM Fast Gradient Sign Method.

FIR Finite Impulse Response.

FLF Fuzzy Logical Function.

GAN Generative Adversarial Network.

GMM Gaussian Mixture Models.

GOFAI Good Old-Fashioned Artificial Intelligence.

GRU Gate Recurrent Unit.

IoU Intersection over Union.

LOF Local Outlier Factor.

LRP Layer-wise Relevance Propagation.

LSTM Long Short-Term Memory.

ML Machine Learning.

MNIST Modified National Institute of Standards and Technology.

MSE Mean Squared Error.

NLP Natural Language Processing.

NN Neural Networks.

OCSVM One-Class Support Vector Machine.

PCA Principal Component Analysis.

ReLU Rectified Linear Unit.

ResNet Residual Neural Network.

RNN Recurrent Neural Network.

SGD Stochastic Gradient Descent.

SotA State of the Art.

SVDD Support Vector Data Description.

UI User-Interface.

VAE Variational Autoencoder.

VGG Visual Geometry Group.

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Deep learning (DL), a subset of artificial intelligence, is a rapidly growing field that has led to significant advancements in many areas such as computer vision, natural language processing, and speech recognition. It involves the use of neural networks with multiple hidden layers, which can learn and extract complex patterns and representations from vast amounts of data.

This area is one of the most promising trends for the future, and even nowadays is used in a wide range of applications. These implementations assist the end-user in decision-making or can directly make decisions and perform actions for the user itself. They are used for everyday tasks such as customer selection based on shared interests, weather forecasting, image recognition, natural speech recognition, autonomous driving, text translation, etc. As the autonomy of DL models increases, so does the risk that we want to eliminate because even a small number of bad decisions can have very serious consequences. The great advantage of the deep learning area is technological development, which enables the enhancement or replacement of human abilities across a variety of industries. Models like this can be used in any sector to create new opportunities and increase efficiency. However, there are still a few milestones to be reached before AI algorithms are deployed en masse in any field. One of the most important ones is the explainability of these systems.

So, neural network models are due to their complexity, notoriously difficult to explain and give mostly no or very poor explanations to the end user. This state of affairs is not ideal for both practical use and future research and development, as the models are mostly black boxes. It is considered a major drawback, especially in areas such as medical diagnosis, finance, and autonomous systems, where decisions based on deep learning models would have a significant impact. Explainability in deep learning refers to the ability to understand, interpret, and provide justification for the decisions made

by these models. The lack of explainability can create a barrier to the wider adoption of deep learning models in various applications, as it makes it challenging for users to trust the predictions made by these models. However, this problem is not straightforward as the robustness of the model increases with increasing model performance, which means that the model has a huge number of parameters and the information in the models is difficult to track while lacking a clear representation. The second reason is the non-linearity of these systems where the activation functions can transform the space in a way that is incomprehensible to the human brain.

Another significant problem is that out-of-distribution data are often not considered when it comes to explainability. This can lead to ambiguities in interpretation when the model essentially tries to explain unexplainable data. Due to this, an anomaly detection system should be an integral part of the solution. Anomaly detection is another crucial task in many areas, including cybersecurity, finance, and healthcare, where identifying unusual or suspicious events is crucial for ensuring the safety and stability of systems. But traditional approaches to anomaly detection, such as rule-based systems and statistical methods, have limitations in handling complex and dynamic data patterns. On the other hand, Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can learn and represent complex patterns and relationships in data. This makes them well-suited for detecting anomalies in high-dimensional, dynamic, and non-linear data sources.

The research presented in this dissertation aims to address the issue of explainability in DL and highlights the challenges faced in applying DL to anomaly detection. The experiments aim to demonstrate the potential of deep learning to reach adequate accuracy and robustness of anomaly detection systems and provide a better understanding of how the novel design can be applied in real-world scenarios. The next objective is to develop a method that can enhance the explainability of deep learning models while maintaining their accuracy. The contribution to the field of anomaly detection by providing insights into the capabilities and limitations of deep learning models for this task is the second part of this work. The main part, is a contribution to the explainability domain, to provide a better understanding of how DL models make their predictions, thereby fostering trust in their use in various applications.

Chapter 1

Dissertation Thesis Objectives

Motivation

The concept of "Good Old-Fashioned Artificial Intelligence" (GOFAI) operates on the principles of a specific scientific discipline, wherein logical operations derive decisions from knowledge-based concepts. The scientific discipline tasked with explaining these concepts is the same discipline within which the decision-making system is applied. The system user is assumed to have a fundamental understanding of the applied scientific field. In contrast, machine learning systems extract the necessary features to make decisions from vast amounts of input data, based solely on their need for successful decision-making. The output generated by these systems is akin to a black box for the user, which makes it necessary to develop a method to enable users to understand the extracted features. By understanding the features generated by systems with higher prediction success than humans, a user with knowledge in the application area will also understand the meaning of newly obtained features that he did not know about before. An often ignored issue with the explainability of decision-making systems is the possibility that the input samples could be data on which the system was not trained, leading to attempts to explain the inexplicable. This means inevitably failing to interpret results. Thus, we view the protection against anomalous data as a crucial aspect of an explainable decision-making system. Given the practical applications and existing datasets, this work will also concentrate on the threat of anomalous data (outliers) in inference processes.

Objectives:

- The primary objective of this work is to develop a method for post-hoc explaining the decisions made by a deep learning system, specifically for image recognition.
- The secondary objective is to develop an anomaly detection technique that avoids the explanations of anomalous inputs. relies on deep generative modeling. Such a system could serve as an extension of explainability methods, thereby enhancing the robustness and reliability of systems. Alternatively, it could also be implemented as a standalone application.

Chapter 2

Theoretical Foundations

2.1 Deep Learning

Today's computers are mostly Turing machines, which are extremely fast at executing instructions. For example, adding digits at high speed - thousands, possibly millions per second - is impressive, but it is not enough nowadays when the complexity of tasks is increasing. It is extremely challenging for the human brain to perform arithmetic operations with high numbers at high speed, but the important fact is that the actual process of such calculations does not require above-average intelligence, it only requires knowledge of the algorithm.

On the other hand, some tasks are very difficult for traditional computing devices, but the human brain, on the contrary, can process and solve them easily. A human looks at an image with human faces, animals, etc., and in most cases, can identify the object immediately. The entire process is really fast, with a minimum of effort and with very high accuracy. This is because the brain can process the large amount of information that the image contains and can identify very successfully what is in the image. These types of tasks are not easy for computational devices because we do not know the solution algorithm and they require a high level of parallelization. Because of this, there has been a requirement for computing devices to be able to solve cognitive tasks in addition to algorithmic ones and to continuously improve in this area. It could be said that a certain level of intelligence is necessary for these types of tasks. "Good old-fashioned artificial intelligence"(GOFAI) approaches problem-solving by using human knowledge of the problem and proceeding to a solution in logical steps.

Work in this cognitive area began shortly after World War II and the name itself (Artificial intelligence - AI) was coined in 1956. However, the very idea that a machine might have the ability to exhibit intelligent behavior indistinguishable from that of a human was there before. Alan Turing, considered by many to be the father of artificial intelligence, in his work [1] from 1950 asked the question:

"Can machines think?"[1]

However, he also introduced another way of answering this difficult question: **by creating a Turing test**. A computing device-machine can pass the test if a person, after asking several written questions, cannot tell whether the written answers come from a person or a computer. To pass a designed test requires unusual effort and the machine would have to master disciplines such as:

- **Knowledge Representation and Reasoning** - the specific form of information that a computer system can use to solve complex problems, answer questions and draw new conclusions.[1]
- **Machine Learning** - The ability to adapt to new circumstances, detect new features, and change behavior according to them.[1]

However, the base version of the test does not take physical interaction into account. Turing deliberately avoided disciplines like that, because the physical simulation of a person is not necessary for intelligence. However, an extension of this test exists - (**Absolute Turing Test**) where the test includes a video signal so a human supervisor can test the subject's perception, as well as the ability to handle objects. In order to pass this extended version, the machine must satisfy even more disciplines[2]:

- **Natural Language Processing (NLP)** - the ability to communicate successfully,
- **Computer Vision** - the ability to perceive/see objects,
- **Robotics** - the ability to manipulate objects.

These 5 disciplines make up the majority of machine learning and deep learning research to this day, and the Turing Test is still relevant even 70 years later.[2].

In our opinion, even passing the Turing Test gives no answer to the actual question: **"Can machines think?"**
But rather it gives an answer to the question: **"Can machines imitate thinking?"**
One of the highlights of GOFAI is the results achieved by IBM's Watson system [3]. The system was especially popularized in 2011 in the TV knowledge quiz "Jeopardy", in which Watson won against two of the best human participants up to that time. If GOF-AI [4] concludes data with the use of logical operations and expert knowledge, it should be possible to make conclusions directly by extracting them from the input data. This idea is implemented in the form of a neural network (NN). The identification of NN parameters is based on the optimization of neuron weights and biases in a way that cost function (e.g. classification error) is at a minimum. The described approach to identifying the right set of system parameters is called Machine Learning (ML) in general, and it is one of the main designs in developing Artificial Intelligence (AI) systems. The subset of ML when we are dealing with NN is called Deep Learning (DL). Machine Learning spreads over a huge range of fields, from abstract areas (learning, explainability, ...) to more specific ones, such as games, mathematical theorems, creating art, autonomous driving, and diagnosing diseases. Nowadays it is a field relevant to every intellectual task. [5]

Overall AI domain is about extracting knowledge from data. It is also known as Predictive Analytics or Statistical Learning. The use of these methods in everyday life has increased in the last 20 years, thanks to the digital age and immense computational resources (applied mainly in computer vision and NLP). And so, these methods are gradually becoming ubiquitous. Many modern websites and computing devices have implemented these intelligent systems in one way or another, and in many cases, these algorithms can also work effectively locally in real-time. Outside of the commercial realm, Machine and Deep Learning methods have had a huge impact on the research field. It is effectively used as a tool, for various scientific problems of the present time. In the early days of "intelligent" applications, many systems used sets of pre-defined rules that consisted of various 'if' and 'else' statements when processing data or modifying user input. As an example, a simple spam filter moves incoming unwanted emails to the SPAM folder. The plain solution is to create and program certain keywords in a way, that if an incoming email contains such words, the spam filter will automatically

mark the email and classify it as spam. Such an implementation can be considered a textbook example of a model that works according to pre-designed and explicitly pre-defined rules.[6] Engineering the decision rules by hand is suitable for some kinds of applications, especially in areas that are well-known in terms of the modeling processes of the system itself. (see Chapter 3)

However, the need for a pre-defined set of rules, according to which the system should work has two main disadvantages:

- The logic behind the decision-making process is specific to one particular domain and task. Even a small change in the task can result in the need to create the system from scratch,
- During the process of model creation, defining the decision-making rules requires a deep understanding of how decisions should be made, for a given specific task. Also, they should be implemented as if they were being created by an expert, specialized in the given field.

A perfect example, where the procedure of manually defined rules failed, is the problem of detecting and classifying objects in images. Nowadays, smartphones and digital cameras can detect different objects in photos and pictures very accurately, but this relatively simple by today's standards task was unsolved for a long time [7] [8] [9]. How the rule-based systems "see" an image and its pixels is diametrically different from how images are perceived by the human brain. It is this difference in the perception of an image, and its parts that have made it impossible to come up with a good set of rules for describing what constitutes objects like a face in a digital image. On the other hand, the use of deep learning in such a task is a suitable solution. A sufficiently large data set with a collection of face images is enough for the algorithm to find out what characteristics - features are necessary for the identification of faces in the image. The formal definition of ML/DL can be formulated as:

Machine learning gives computing devices the ability to solve any given task without the need for explicit programming. [10].

A key characteristic of this field is precisely **the ability to learn**. This refers to the implementation of systems capable of detecting patterns found in data and

improving performance based on data and empirical information, all without human-defined instructions. [5]. Machine and Deep learning can also be defined as the process of solving a specific problem in two main steps:

1. data collection,
2. parameters identification by the learning algorithm, from previously collected data.

Such a model is then used as a tool to solve a given specific problem, contained in the collected data. It should be taken into account that only data is not enough. It is also highly important to choose the right procedure for how the model should approach collected data. Various problems require setting up the right learning environment in order to make the right choice depending on the task at hand and the available data.

Models in general may be split into **static** and **dynamic**. Static ones represent a non-linear transformation and this dissertation deals exclusively with this type. Static models of Deep Neural Networks (DNNs) are also called feed-forward DNNs. There are several ways to train static models, but the basic ones are:

- *Supervised learning*
- *Unsupervised learning*
- *Semi-supervised learning*

There is also a specific learning algorithm for **dynamic** systems which is called Reinforcement learning

Supervised Learning

The main concept is based on the fact that the training dataset is a collection of labeled data samples,$\{(x_i^j, y_i^j)\}_{i=1}^N$. Each input sample x_i in the set N is defined by coordinates (descriptors) in a descriptor vector, *(otherwise known as a feature vector. In this work, the features refer only to the vectors extracted from the descriptors during the training process).* Such a vector has values $j = 1,....,D$ that somehow describe the given sample. For example, if we had a database of people in which each sample represents one person, then the first coordinate (element) $x_i^{(1)}$ can carry information about the person's height.

Then the second one $x_i^{(2)}$ can contain information about weight, the third $x_i^{(3)}$ can carry information about the gender of the given person, and so on. Important is also the fact, that for all samples in the database, the specific element in the descriptor vector always contains the same type of information across all other data samples. That means, if an element $x_i^{(2)}$ carries information about a person's weight in kilos, in a specific sample x_i, then $x_k^{(2)}$ carries the same kind of information in every other sample $x_k, k = 1,...,N$. The data label y_i can be an element belonging to a finite number of classes $\{1,2,...,C\}$, a real number, or even a more complex structure such as a vector, matrix, and so on. Classes are considered categories, to which individual data samples belong. Assuming the problem of detecting spam emails, we would have exactly two {`spam, not_spam`}.

The grand design of Supervised learning is to train on labeled data, meaning the data has a known outcome or target value. The main goal of supervised learning is to build a model that can make predictions on new, unseen data, based on the patterns it has learned from the training data during the training phase. [11]

Unsupervised Learning

In this type of training algorithm, the database consists of a collection of unlabeled data samples $\{(x_i)\}_{i=1}^{N}$. That means the target or outcome values are not known. Again, in this approach, x represents a vector that describes the given sample. The goal of unsupervised learning is to find patterns or structures in the data without being told what the patterns mean. Some common examples of unsupervised learning algorithms include clustering, a method where the model groups similar data points together based on their vector coordinates or returns an `identifier` to a certain cluster for each input vector x. Another example is dimensionality reduction, where the goal is to reduce the number of features in the data while retaining as much of the original information as possible. In anomaly detection, data points that are significantly different from the rest of the data could indicate an outlier/anomalous sample. So, unsupervised learning should be used whenever the goal is to find patterns in the data without knowing the outcome. [12]

Semi-Supervised Learning

Data samples based on hybrid learning algorithms contain both labeled samples and unlabeled samples. Usually, the frequency of unlabeled data is much higher. The goal of such learning is the same as in the case of learning with a teacher. The added value should be that both types of data are in the interaction, and the learning process can achieve better results in terms of generalization. At first glance, it may seem that we add more uncertainty to the learning process instead of improving the model. However, if we add unlabeled data we bring more generalization to our problem. The larger amount of data better reflects the probability distribution from which the labeled data come. In theory, the learning algorithm should be able to use this additional information. [13]

Reinforcement Learning

Reinforcement learning algorithms are used to train an agent to make decisions in dynamic environments, where the decision made by the agent affects the subsequent state of 'the environment' and the reward received. The agent interacts with the environment by taking action and observing the resulting state and reward. The agent's goal is to learn a policy that maps states to actions and maximizes the cumulative reward over time. The policy is learned through trial and error, where the agent receives feedback in the form of rewards for its actions. The learning process involves the agent discovering for itself the optimal sequence of actions to maximize the reward. Overall, reinforcement learning is a powerful tool for solving decision-making problems in complex, dynamic environments. [14]

2.2 Understanding the domain of Explainability in Man-Machine Systems

The field of cybernetics has established a foundation for the automated control of processes in machines, organisms, and society [15], and computers have become a widespread tool for its implementation. Over the past seventy years, computers have been utilized in various applications, replacing human decision-making in many cases. The creation of computer programs follows a scientific knowledge-based process (the epistemological triangle [16]), in which the problem to be solved is first understood and then translated into a decision-making algorithm using the principles of Aristotle's logic (implemented in digital computers through Boolean algebra). This results in explainable computer decisions, within the limitations of current scientific knowledge. Any errors in these conclusions can be traced and corrected.

While solving complex problems and creating large programs can result in errors, even computer decisions can be incorrect in exceptional circumstances. With the advent of machine learning and its use in creating non-linear models, such as deep neural networks (DNNs), the epistemological triangle method has been set aside. Instead, the parameters of the non-linear model are determined directly from data using machine learning techniques. This form of decision-making, which is also inherent in humans and evolved prior to logical thinking, is utilized when quick decisions are needed and there is not enough time for contemplation. The use of non-linear models, particularly DNNs, has been instrumental in the success of machine decision-making in recent years. However, this approach comes with the drawback of being unexplainable, which is an issue not only for computer-generated decisions but also for human decisions. Although we may tolerate this lack of explanation in standard operating conditions of automated control systems with no history of significant errors, the same is not true in areas such as medicine where human intervention is often required in the final decision-making process [17]. Therefore, it is imperative that decisions made through the use of computers as assistants be transparent and explainable. This issue has garnered significant attention in the field of neural network science, as DNNs have been effective in augmenting human decision-making [18] [23].

Human decision-making is often similar to decision-making based on logical reasoning,

such as decision trees, if-then rules, logical expressions, or linear models, like regression models, PCA [24], or linear predictive models (FIR, ARIMA[25]). Aristotle's axioms for correct thinking have been standardized and implemented in digital computers using Boolean logic. To explain the behavior of non-linear systems, humans have approached it by locally linearizing the systems under the assumption of only small and slow changes over a short period. However, the general user is unfamiliar with the new properties of non-linear dynamical systems such as chaotic behavior, multiple attractors, and sensitivity to initial conditions. Additionally, they lack experience with static non-linear transformations. Despite this, non-linear systems, in the form of deep neural networks, have demonstrated superior performance in decision-making compared to optimal linear systems. Hence, the issue of the explainability of these systems becomes increasingly relevant.

2.2.1 Explainability vs Interpretability

As Artificial Intelligence continues to develop and permeate society, there has been an increasing prevalence of systems that involve collaboration between human users and agents. These systems, commonly referred to as Human-Agent Systems, have evolved from theoretical constructs to practical applications, such as recommendation systems, chat-bots, planning systems, or self-driving cars. A significant consideration in the development of such systems is the quality and type of information exchange required during interactions between human users and agents [26] [33].

One of the primary components of human-agent interaction is the level of internal explainability that agents using complex algorithms must possess in order to account for their decision-making. Nevertheless, there is currently no consensus on the precise definition of explainability. This lack of clarity is likely compounded by the fact that the terms "explainability" and "interpretability" are frequently used interchangeably, while other researchers make distinctions between these concepts and implicitly define them in distinct ways [34] [39].

What is Explainability ?

Explainability in a general sense refers to the capacity to present a clear and comprehensible explanation of a model's decisions to end users in a manner that is both comprehensible and credible.

This requires transforming the technical intricacies of a model into a format that can be easily understood by non-technical audiences. Explainability frequently addresses the impact of a model's predictions on society, including its potential for prejudice or discrimination, and is vital for fostering trust and accountability in the model's application. The paper [40] proposed the mathematical definition of Explainability. It is based on the data samples and features utilized for training within a system, the supervised output that requires identification, and the machine learning algorithm used by the agent. Where the algorithm in use is denoted as L, which is formed from a set of training data samples R. Each sample, $r \in R$ holds values for a set of descriptors D. Each descriptor is identified as $d \in D$. Therefore, the entire training set is comprised of $R \times D$. For instance, if the existing descriptors are $d1 = age$ (in years), $d2 = height$ (cm), and $d3 = weight$ (kg), then D is represented as $\{age, height, weight\}$. An example of a sample, $r \in R$ could be $d(r) = (35, 160, 70)$. Although this example model is commonly utilized with tabular data, it can also be applied to other forms of input such as text, in which d represents strings, or images, in which d represents pixels, etc. The primary aim of the algorithm, L is to accurately fit $R \times D$ with respect to the ground truth - labeled targets, $l \in L$.

So, the goal of any system is to ensure explainability, which involves presenting an explanation E that facilitates human understanding of systems decisions(predictions) T. To create the explanation, the user must first grasp the relationships between the supervised inputs $R \times D$ and the targets L. The user's understanding of an interpretation function I forms the basis for creating the human-friendly explanation E. The function I takes $T, R \times D$, and L as inputs and returns a representation of the underlying logic within T that is readily comprehensible. Now, it is plausible to define the explanation in the following manner:

$$E = I(T(R \times D, L)) \tag{2.1}$$

Quite important is to note that the term "explainability" does not signify "interpretability".

Then what about Interpretability ?

There is currently no universally accepted quantifiable definition of interpretability in artificial intelligence (AI), but a qualitative definition can be given as follows:

Interpretability is the degree to which a human can consistently predict the impact

of the input or internal model variables on the final decision [41] [42].

Interpretability, in contrast, represents the capacity to comprehend the inner workings of a model and the methods by which it generates its predictions. This entails scrutinizing the model's internal mechanisms, such as its hidden layers, activations, and weights, in order to gain an understanding of its decision-making process. Interpretability is frequently centered on the technical aspects of a model, including its architecture and parameters, and is critical for comprehending the reasoning behind a model's predictions and how it can be enhanced.

2.2.2 Attributes of Explanations

In terms of the explainability of systems, we always aim to explain the predictions made by the black-box model. In order to accomplish this objective, we used an explanation method, which constitutes an algorithm designed to produce explanations. In the paper, [43] and [22] were proposed properties of explanations that may serve as a basis for evaluating the quality of an explanation method or the explanation itself.

Performance Measures of Individual Explanations

- **Accuracy** - To what extent does an explanation predict unseen data? If the explanation is utilized for predictions instead of the black box, high precision is particularly crucial. If the accuracy of the black box model is low and the aim is to decode the workings of the black box, low precision may suffice(fidelity would be the only critical factor).

- **Fidelity** - To what extent does the explanation approximate the predictions of the black-box model? This is a crucial attribute of an explanation, as one with low fidelity is incapable of explaining the workings of the black-box model.

- **Comprehensibility** - To what extent do humans comprehend the explanations? It is a crucial factor, but quite hard to define and measure.

- **Stability** - To what extent are the explanations for similar instances comparable?

- **Consistency** - To what extent do the explanations vary between models that have undergone training for the same task and generate comparable predictions?

- **Certainty** - Does the explanation incorporate the level of certainty associated with the machine learning model? It is worth noting that certain machine learning models solely offer predictions without explicitly indicating the degree of confidence in the accuracy of their predictions.

- **Novelty** - Does the explanation account for the possibility that the given data sample could originate from an out-of-distribution region significantly distant from the distribution of the training data? In such circumstances, the model's accuracy may be compromised, rendering the explanation futile.

- **Degree of Importance** - To what extent does the explanation accurately convey the significance of the features or components of the explanation?

2.2.3 Taxonomy of Explainable Methods

The explainability of deep learning models can be categorized into several groups proposed in [22], [43], [40] including:

- **Intrinsically (built-in) Explainable Models.**
  Explainability can be attained by constraining the complexity of the model. This type of explainability pertains to models that are deemed explainable due to their simple design (decision trees, sparse linear models).

- **Post-hoc explainable methods.**
  Explainability can be accomplished by utilizing techniques that analyze the model after its training. Post-hoc methods can also be used to improve the explainability of models that are already inherently explainable.

- **Model-Specific methods**
  Explanation methods like this are used for explaining models that are specific to particular model classes. They are not universally applicable.

- **Model-Agnostic methods**
  They are universal and follow a post hoc approach. These agnostic methods generally examine the input and output pairs of the features. Also, these methods cannot access the model's internal parameters.

- **Local explanation.**

  A method is considered local if it provides an explanation for an individual prediction.

- **Global explanation.**

  A method is considered global if it explains the overall behavior of the model.

In the realm of deep learning and neural networks, model-specific methods are commonly used, which are restricted to particular types of models, including models with inherent explainability. A lot of sources like [44] - [52] address the issue of explainability.

2.3 Neural Network Interpretation

In order to generate predictions using neural networks, the input data must traverse multiple multiplication layers with learned weights and non-linear activation functions. The complexity of a single prediction, dependent on the network architecture, involves millions of mathematical operations. As a result, it is infeasible for humans to trace the precise mapping of input data through the network to the output. To comprehend the predictions of a neural network, it would be necessary to consider the intricate interaction of millions of weights. Thus, the use of specific interpretation methods is necessary to interpret the behavior of the model. It is worth noting that conventional model-agnostic methods may also be used. There are two reasons for the utilization of methods specifically developed for neural networks:

- Firstly, neural networks learn features and concepts in their hidden layers, requiring the application of special methods for interpretation.

- Secondly, the gradient can be utilized to implement interpretable methods, making them more computationally efficient compared to model-agnostic methods.

2.3.1 Adversarial Examples

An adversarial example refers to an instance in a dataset that has undergone small, intentional modifications to its features, leading to an incorrect prediction by a machine learning model. It is similar to Counterfactual Explanations, with the distinction being that the goal of adversarial examples is to mislead the model, rather than interpret it. There are various techniques for creating adversarial examples, which generally involve minimizing the distance between the instance and its adversarial counterpart while ensuring that the prediction outcome changes to the desired result. Some methods require access to the gradients of the model, which is limited to gradient-based models like neural networks, while others only require access to the prediction function, making them model-agnostic. In the case of image data, adversarial examples involve intentionally manipulated pixels, designed to trick the model during runtime.

In the paper on adversarial examples [53], a gradient adjustment optimization approach was used to discover adversarial examples for Neural Networks. These patterns were generated by minimizing the following loss function with respect to r [22]:

$$loss(\widehat{f}(x+r),l) + c \cdot |r| \tag{2.2}$$

In this equation, x represents an image as a vector of pixels, r represents the modifications made to the pixels to generate an adversarial image (such that the result, $x + r$, yields a new image), l represents the intended outcome class, and the constant c is employed to balance the distance between the images and the distance between predictions. The first component in the equation calculates the distance between the predicted outcome of the adversarial example and the desired class l. Meanwhile, the second component measures the distance between the adversarial example and the original image. It is worth noting that this formulation closely resembles the loss function utilized in the generation of counterfactual explanations. Additionally, there are restrictions placed on r to ensure that the pixel values remain between 0 and 1.

An alternative option to consider is the Fast Gradient Sign Method (FGSM) [54] for producing adversarial images. The FGSM leverages the gradient of the underlying model to identify adversarial examples. This approach involves perturbing the original image x by adding or subtracting a small error, denoted by ϵ, to each pixel. The determination of whether to add or subtract ϵ is based on the sign of the gradient (positive, negative)

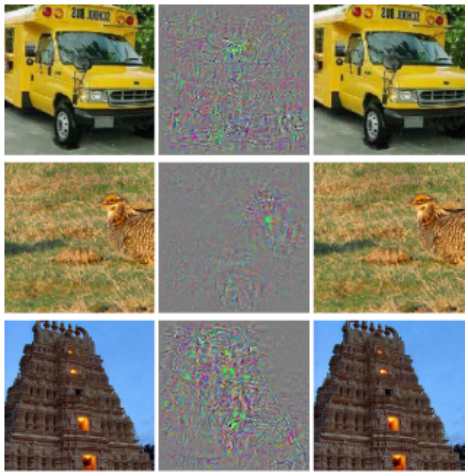


Figure 2.1: (left) correctly classified sample,
(middle) the magnified error added to the sample,
(right) adversarial example [53].

for that particular pixel, with the direction of the gradient determining the intentional alteration of the image so as to induce a failure in the model's classification. The core of this method can be mathematically defined as [54] [22]:

$$x' = x + \epsilon \cdot sign(\nabla_x J(\theta,x,y)) \tag{2.3}$$

where $\nabla_x J$ represents the gradient of the model error function with respect to the actual input x (pixel vector), y is the actual label, and θ includes the model parameters. The gradient vector (which has the same length as the vector of input pixels), is important only for the sign of the gradient. This sign is positive (+1) when an increase in pixel intensity results in an increased number of errors made by the model. Conversely, the sign is negative (-1) when a decrease in pixel intensity increases the loss. The vulnerability of this method arises when neural networks treat the relationship between an input pixel intensity and the class score linearly. It is mainly specific network architectures, such as LSTMs, networks with ReLU activation units, and logistic regression, which favor linearity. Thus, even partial linearity leads to vulnerability to extreme values, so the model can be fooled by shifting the pixel values of the image to regions outside the data distribution. Besides that, it might be tempting to believe that these adversarial examples are unique to a specific neural network architecture. However, the reality is that adversarial examples can be reused to deceive networks with different architectures trained on the same task.

Thus, the gradient orientation method requires all pixels to change by at least a small amount. Nonetheless, it is also possible to deceive the model by altering only a single pixel in an image [55],[22]. The 1-pixel attack resembles counterfactual interpretations because it also seeks a modified example $x\,'$ that is close to the original image x but results in a different classification. However, this time the notion of proximity is defined differently: only a single pixel may be altered. This method uses the so-called differential evolution to determine the pixel to be altered and the extent of alteration. A certain number of pixels considered necessary to achieve a solution are recombined until the correct solution is attained. Each of the possible solutions encodes a pixel modification and is represented by a vector of 5 elements - x and y coordinates and RGB (Red, Green, Blue) values. The search starts with a different number of possible solutions (e.g 500 suggestions of pixel modification), where each solution represents a certain pixel modification and also creates a new generation of solutions (new solutions - children of parent modifications) such as [55]:

$$x_i(g+1) = x_{r1}(g) + \alpha \cdot (x_{r2}(g) - x_{r3}(g)) \tag{2.4}$$

where each x_i represents an element in the candidate solution (i.e. x-coordinate, y-coordinate, or R.G.B.). The g attribute is the actual generation of solutions. Next, α is a scaling parameter, and $r1$, $r2$, and $r3$ are different random numbers. Each subsequent child is a solution with a 5-element vector and each of those attributes is a mixture of three random parent pixels. The creation of new possible solutions (children) is stopped if any of the actual provided solutions are successful(adversarial example) or the maximum number of iterations specified by the user is reached.

Other approaches to creating adversarial examples involve physically constructing contradictory cases, as demonstrated in works such as [56] and [57]. In [56], the authors developed a special label that, when placed beside the object under classification, can mislead the classifier into believing the object is a toaster. Similarly, in [57], the authors created a 3-D turtle that appeared as a gun to a deep neural network from nearly every viewpoint. These methods diverge from above mentioned adversarial methods as they do not require the adversarial image to closely resemble the original. Instead, they substitute part of the image with a label of any shape that has been optimized for different images and positions (such as shifted, enlarged, reduced, rotated, or mirrored).

Adversarial attacks present a formidable challenge for achieving explainability in

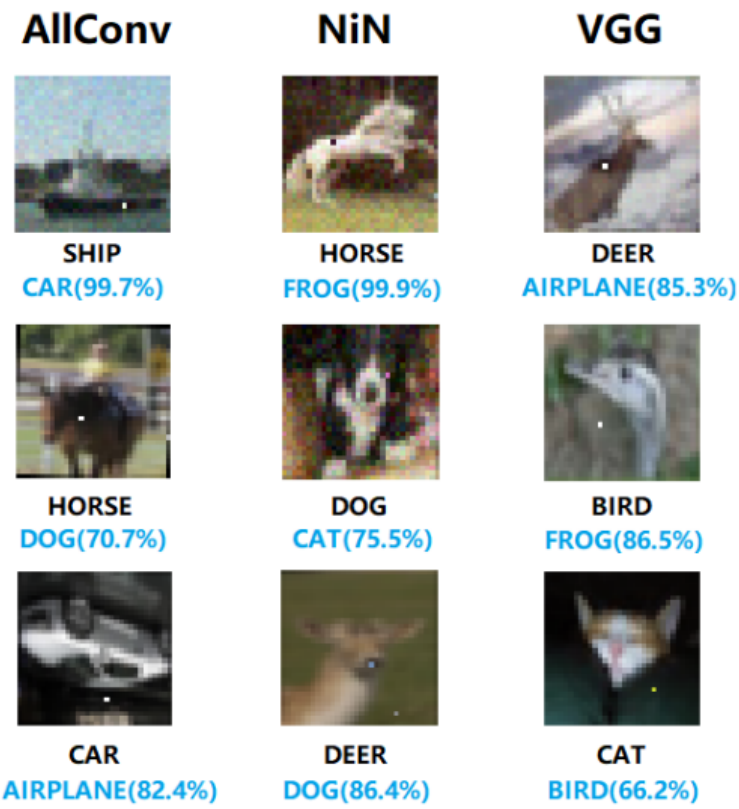


Figure 2.2: Showcase of one-pixel attacks [55].

machine learning. A key issue is how to provide a clear and comprehensive explanation of the change in classification that occurs when an adversarial noise is introduced. In our view, a potential solution to this challenge is analogous to that of anomaly detection. Specifically, the inclusion of a filter that screens out samples containing adversarial noise can be an effective approach to improve explainability. This topic requires further research to fully comprehend its feasibility and effectiveness.

2.3.2 Hierarchical Learned Features

Deep Neural Networks (DNNs) possess a significant advantage over other model options due to their capability of learning complex features in the hidden layers without the need of engineered them by hand. When an image is fed into a Convolutional Neural Network (CNN) [58], the network processes it multiple times through its convolutional hidden layers. During this process, the network acquires and enhances increasingly complex features as the image progresses through the network. Finally, the transformed image information is passed through fully-connected (dense) layers, resulting in a

classification or prediction of the model. The possibilities of how to interpret these learned hierarchical features are usually through visualization. Therefore they are generally called feature-visualization methods. [59]

Visualization of Activations

The method proposed in [60] is centered around the visualization of activation maps from a selected layer. This provides insight into how individual components of the input image (referred to as receptive fields) influence the values of individual neurons. Consequently, it is possible to examine, for instance, the neurons with the highest activation through additional methods for further investigation.

Feature visualization through optimization

The method of activation maximization could be applied to various components of a neural network, including neurons, activation maps (convolution channels) of a convolution layer, entire convolution layers, and even predictive neurons at the output. The fundament of the method is based on searching for an input that leads to the maximum activation of the selected unit. While individual neurons represent the smallest structural component of the network and offer the most comprehensive information, this approach is not practical due to the complexity of neural networks, which can contain millions of neurons, and the time-intensive nature of the operation. As an alternative, feature maps (convolution channels) in convolution networks or entire convolution layers may be selected for activation maximization, providing a balance between the level of detail and computational efficiency. [61] [62] Mathematically, activation maximization can be viewed as an optimization problem. With the parameters of the neural network fixed (the network is considered to be trained), the objective is to find an image that maximizes the average activation of a selected network unit, such as a neuron. [22]

$$img^* = \underset{img}{\operatorname{argmax}}\, h_{n,x,y,z}(img) \tag{2.5}$$

The equation represents the activation of a network unit, denoted by h, as a function of the input image (img), the layer specified by n, and the spatial information provided by x and y. The index of the convolution channel is represented by z. If the entire convolution channel is taken into consideration, the calculation of the average activation would be:

$$img^* = \underset{img}{\operatorname{argmax}} \sum_{x,y} h_{n,x,y,z}(img) \tag{2.6}$$

In this scenario, each neuron in the feature map z carries equal weight, yet there exists the potential to explore the interactions among neurons within the map. Another approach is to maximize individual connections at random, resulting in neurons being multiplied by varying parameters, even in a negative direction. This not only maximizes activation but also allows for its minimization. An intriguing observation is that when the activation is maximized in the negative direction, distinct symptoms arise for the same network unit [63]. The concept of feature visualization bears a strong resemblance to adversarial examples as both approaches aim to maximize the activation of the neural network unit. However, in the case of adversarial examples, the goal is to achieve maximum activation of the neuron for an incorrect (contradictory) class. Another distinction lies in the initial images used. Adversarial examples require an image that needs to be a distorted copy of the original, whereas in feature visualization probably the best approach is to start with random noise. A slightly worse alternative to generating new images is to search through the training images and select those that maximize the activation.[64][65] While this approach is valid, it has the potential drawback that elements within the images may be correlated, making it difficult to determine what is truly important for a given neural network.

Network Dissection

Convolution layers operate by learning new features through their individual channels, which are also called feature maps. However, simply visualizing these features is insufficient to verify if a Network Unit has learned a specific concept. Additionally, a metric is necessary to evaluate how accurately the network unit detects the learned concept. While some feature maps may display recognizable concepts to the end users (such as different objects or human contours - depending on the task), this may not be the case across the board. To address this problem, the Network Dissection method was proposed [66]. The Network Dissection method quantifies the interpretability of a Convolution network unit by linking highly activated areas of feature maps with human concepts like objects, textures, and colors. However, for this method to work, it is based on the concept that CNNs could learn disentangled features, meaning that specific network units detect individual concepts. Conversely, if the network lacks individual units that detect specific concepts and instead it relies on collective contributions for recognition, it is not possible to determine a decision-making process

for the network (a state that is uninterpretable). As a result, CNNs with disentangled features are considered highly interpretable since the detection of individual concepts enables monitoring the whole decision-making process. But as always, it is not that easy, and CNNs are not perfectly disentangled. The Network Dissection consists of 3 main steps (assuming the network has been trained and the weights are fixed)[66]:

- Establish an annotated dataset of visual concepts with varying degrees of abstraction (ranging from simple to more complex).
- Assess the activation of CNN feature maps for the annotated data.
- Quantify the extent to which the activation of feature maps aligns with the annotated concepts (through the IoU metric - intersection over union).

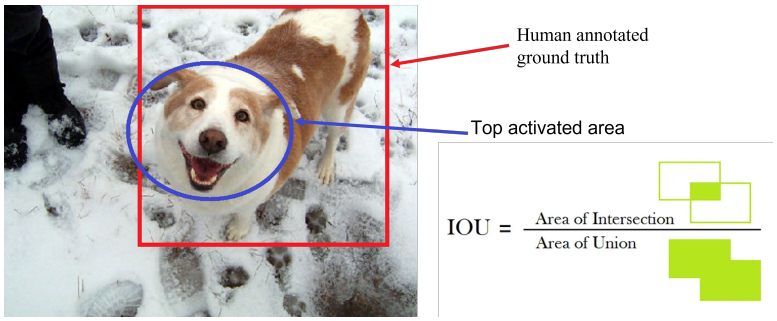


Figure 2.3: To compute the Intersection over Union (IoU), the human ground truth annotation is compared with the top activated area

In general, feature visualizations offer valuable insight into the internal workings of neural networks. Given the complexity and opacity of networks, this method represents a crucial step toward analyzing and interpreting their behavior. Last but not least the paper [67] introduced feature visualization in terms of text and tabular data

2.3.3 Saliency Maps

Methods that fall under this concept are a specific form of feature attribution (also referred to as pixel attribution when working with image data). Essentially, this approach interprets the model's predictions by assigning a weight to each input variable based on its impact on the prediction (positive or negative). These variables may take the form of pixels (in the case of images), words, or tabular data. There are various

feature attribution techniques available, but they can be broadly categorized into two types [22]:

- Occlusion or perturbation-based methods such as LIME [68], modify specific parts of an image to generate explanations. This is a model-agnostic approach.
- Gradient-based methods compute the gradient of the prediction concerning the input data.

As an illustration, a model outputs a prediction in the form of a vector of length C. The output from the Network for the input image I is $S(I) = [S_1(I),...,S_C(I)]$. All methods require an input $x \in \mathbb{R}^p$ with p features and then output a contribution score for each p input feature as an explanation. When dealing with imagery, both types of approaches have in common that they produce an output explanation that is the same size as the input image, assigning a value to each pixel. This value is indicative of the pixel's significance in the prediction or classification of the image.

Image-Specific Class Saliency

The Image-Specific Class Saliency method (or Vanilla Gradient) [69] is among the earliest feature attribution techniques based on gradient computation. This approach is relatively straightforward, assuming familiarity with the gradient backpropagation method. To begin, we compute the gradient of the model's loss function for the class of interest, given the input data (image). Once it's done, the output is a map reflecting the size of the input features, with both positive and negative values.
The algorithm looks like this[69]:

Algorithm 1 Image-Specific Class Saliency

Run the image of interest through a forward pass.

Compute the gradient of a class score of interest with respect to the input data:
$E_{grad}(I_0) = \frac{\partial S_c}{\partial I} \mid_{I=I_0}$

Crucial step is to assign 0 to all the other classes.

Visualize the gradients by displaying their absolute values or by color-coding the positive and negative contributions separately.

This controlled backpropagation technique is effective at visualizing the intricate details within an image, with each unit serving as a specific feature detector. Usually,

it's not positive/negative values that matter, but rather the magnitude of the gradient. Pixels with small gradient magnitudes are typically considered unimportant for predictions. By contrast, if the magnitudes of the gradients are large, it signifies that the pixel is important. However, the gradient's sign remains significant as it provides additional information. The negative value corresponds to a kind of inverse detection of a feature during the processing of an image. On the contrary positive values corresponds to positive detection of a certain feature. To sum it up, it should be noted that the interpretation methods of saliency maps are not always reliable. This approach has a saturation problem described in [46] Data preprocessing, including normalization, may produce undesirable changes in the output maps [70]. Furthermore, these maps can be vulnerable to adversarial attacks, as demonstrated in [71].

DeconvNet

The Deconvnet method is almost identical to the Vanilla Gradient method. Its purpose is to reverse the neural network backward. The original paper [72] proposes operations that are reversals of filtering, pooling, and activation layers. This approach provides insight into the input image's components that influence the consequent neuron activation. So, the DeconvNet technique is grounded in the principles of convolution neural networks themselves. Modifies the current CNN layers to allow operations of the same layer in a backward fashion, creating a return path to the input image. The process starts with an input image prediction, then obtaining activation maps for the layer containing the monitored unit. Subsequently, all the map values, except for the monitored neuron's value, are set to zero. These maps serve as input to the DeconvNet method. The proposed work assumes the use of ReLU as the activation function. It is crucial to note that identical activation functions are essential for the full reconstruction of unit contribution. Nonetheless, the same approach may be applied to ELU or LeakyReLU activation functions. The method resembles vanilla gradient calculation, but DeconvNet selects a different approach for backpropagating the gradient through ReLU. Additionally, the monitored unit's value is considered during the gradient calculation.[72]

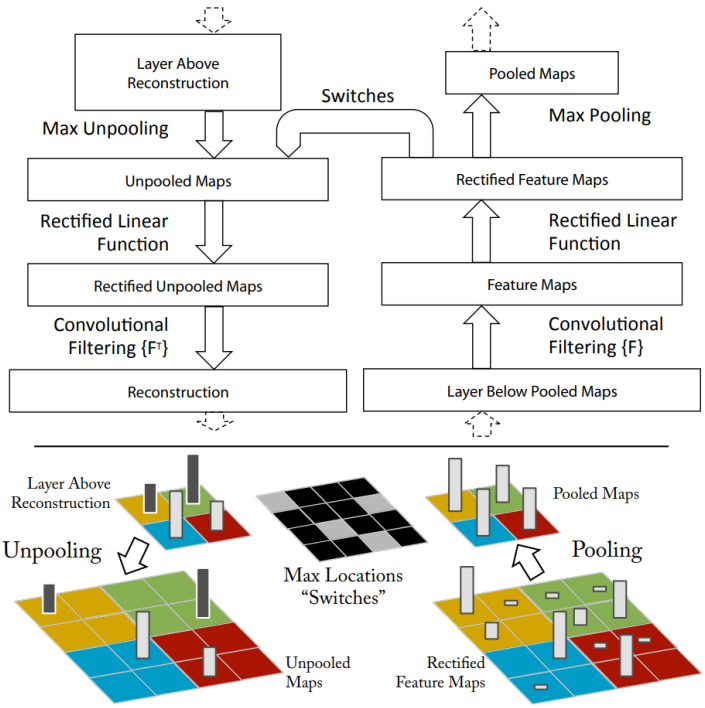


Figure 2.4: At the top of the figure, we can see a DeconvNet layer (left) connected to a ConvNet layer (right). The role of the DeconvNet layer is to reconstruct an approximate version of the ConvNet features from the layer beneath. The bottom of the figure illustrates the unpooling operation in the DeconvNet. This process utilizes switches that record the location of the local max in each pooling region (indicated by colored zones) during the pooling process in the ConvNet. [72]

Guided Backpropagation

This method is another approach that visualizes the activation of the neuron at the input image level and involves the gradient calculation process, as implied by its name. The authors refer to this approach as a modification of the DeconvNet method, which yields significantly better results. The modification in this method involves adjusting the gradient calculation at the ReLU activation function (again) by combining both the DeconvNet and Saliency Maps ('Vanilla Gradient') approaches. With this, the negative gradient from the higher layer is zeroed out, and the prediction process's information is taken into account at the same time, leaving only the modified gradient at positive positions. This method operates on the assumption that a negative gradient reduces the activation value of the observed neuron (the inverse detection of feature), thus setting it to zero [73].

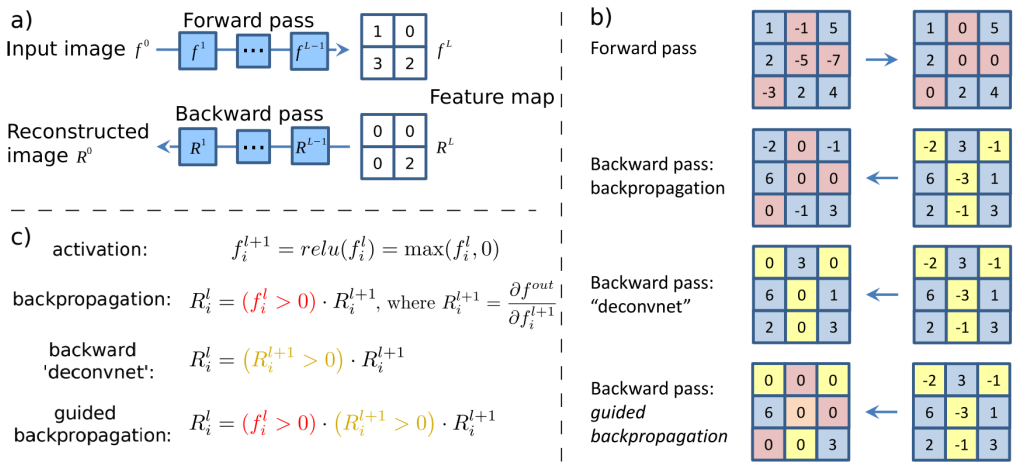


Figure 2.5: Showcase for comparison of gradient computations by various visualization methods [73]

a) It begins by providing an input image and conducting a forward pass to the desired layer of interest. The next step is to set all activations to zero, except for one, and backpropagate to the image in order to generate a reconstruction.

b) The image showcases various techniques for backpropagating through a ReLU nonlinearity.

c) Depicts the precise definition of various techniques used for backpropagating an output activation through a ReLU in specific layer l. The Guided Backpropagation and 'Deconvnet' approach do not calculate a true gradient, but an imputed version. [73]

Grad CAM

Grad-CAM (Gradient-weighted Class Activation Map) [74] is another method that can provide visual interpretations for the decisions made by Convolutional Neural Networks (CNNs). Unlike other methods, Grad-CAM does not backpropagate the gradient to the input data. Instead, it typically computes the gradient with respect to the last convolutional layer, which results in a coarse localization map. This map highlights the areas in the image that have the most influence on the model. Grad-CAM assigns a contribution to each unit of the network. This can be the model output prediction, or it can be just another layer in the network. Grad-CAM's objective is to determine the areas of the image that the convolutional layer focuses on for a particular classification. It accomplishes this by analyzing the activated areas in the feature map of the previous convolution layer. The information passes through the

convolution layers in the form of feature maps, with the features gradually becoming more complex. Therefore, Grad-CAM's analysis identifies the areas of the feature map that are activated in the previous convolutional layer.[74] When analyzing a Convolutional Neural Network (CNN) with k-feature maps $A_1, A_2,...A_k$, and a specific class of interest c, simply averaging the feature maps and displaying them in an image is insufficient for understanding how the CNN made its decisions. Grad-CAM is a method that determines the importance of each feature map in classifying a given class c. This involves weighing each pixel of the feature maps with a gradient before averaging the maps, which results in highlighted areas that contributed positively or negatively to the prediction of class c. After the gradient-weighing process, the resulting heat map passes through the Rectified Linear Unit (ReLU) function to remove negative values, which correspond to inverse detections. This step is important because we are only interested in the areas that contributed positively to the classification of the given class. In this scenario, the term "pixel" can be misleading in this context since the final heat map is smaller than the original image due to pooling units, but at the end of the process is scaled back to the original input size. The Grad-CAM map is then normalized to the interval [0,1] for visualization purposes and overlaid onto the original image. Examining the Grad-CAM approach, the main aim is to determine the localization map, which is defined as [22]:

$$L^c_{Grad-CAM} \in \mathbb{R}^{x \times y} = ReLU\left(\sum_k \alpha^c_k A^k\right) \tag{2.7}$$

the variables x and y represent the width and height of the explanation, respectively, and c represents the class of interest. This method provides class-selective visualizations that only localize relevant areas of the image, without emphasizing the importance of individual pixels, as the guided backpropagation method does.

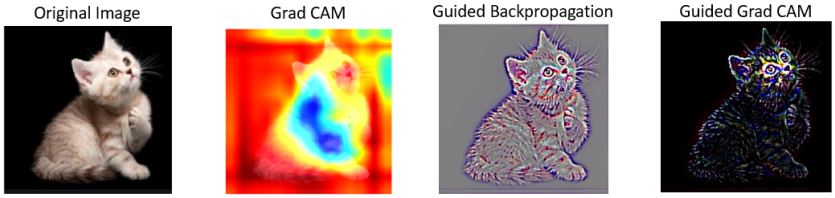


Figure 2.6: Different methods for interpretation of network predictions [74]

Guided Grad-CAM

From the explanation of Grad-CAM, it can be inferred that the localization is less precise due to the lower resolution of the last convolutional feature maps compared to the input image. In contrast, other attribution techniques, which backpropagate all the way to the input pixels, offer more detailed information and can pinpoint individual edges or spots that were most influential in a prediction. A combination of both types of methods is known as Guided Grad-CAM, which is a simple technique. The Grad-CAM and another attribution method, such as 'Vanilla Gradient', are computed for an image, then the Grad-CAM output is upsampled using bilinear interpolation. The two maps are then multiplied (element-wise), with the Grad-CAM acting as a lens that focuses on specific areas of the pixel-wise attribution map.[22]

SmoothGrad

The SmoothGrad paper [75] proposes a method to reduce noise in gradient-based explanations by adding noise to the image and averaging the resultant artificially noisy gradients. Note that SmoothGrad is not an independent explanation technique, but rather a supplement to any gradient-based explanation method.
The following algorithm describes how SmoothGrad operates [75]:

Algorithm 2 SmoothGrade Approach

Generate multiple versions of the image by introducing noise.

Generate pixel attribution maps for all images.

Average the pixel attribution maps.

$R_{sg}(x) = \frac{1}{N} \sum_{i=1}^{n} R(x + g_i) \qquad g_i \sim N(0,\sigma^2)$

The rationale behind this idea is that the derivative shows significant variations when observed at a small scale. In the training phase, neural networks prioritize the delivery of the most accurate prediction for a given task over maintaining smooth gradients. By averaging over multiple maps, the fluctuations in the gradient are "smoothed out" [22]. There are also some extensions of the Grad-CAM original method presented in papers [76, 77]

Layer-Wise Relevance Propagation

Another method in this domain measures is Layer-Wise Relevance Propagation (LRP) [78]. It is used to interpret the decisions made by the network by assigning relevance scores to the input features. The LRP technique propagates the relevance scores all the way back through the network, utilizing a predetermined set of propagation rules. The rules used in this process ensure that the relevance activations are preserved through the whole process of propagation, maintaining the main condition that the total relevance at a layer's input is equal to the total relevance at its output.

The Layer-Wise Relevance Propagation (LRP) method employs several standard propagation rules [78], including:

- The Z^+ *rule*: This rule divides the relevance at the output of a neuron among its input neurons in proportion to their positive activation values.
- The Z^- *rule*: This rule divides the relevance at the output of a neuron among its input neurons in proportion to their negative activation values.
- The ϵ *rule*: This rule adds a small positive constant (ϵ) to all activation values before dividing the relevance. This prevents division by zero.
- The $\alpha - \beta$ *rule*: This rule combines the Z^+ and Z^- rules by dividing the relevance among input neurons based on a combination of their positive and negative activation values.

Mathematically, relevance scores R can be defined as:

$$R_i^l = \sum_{k=1}^{B_{l+1}} R_{i\leftarrow k}^{l,l+1} = \sum_{k=1}^{B_l+1} \frac{a_i^l w_{i,k}^{l,l+1}}{\epsilon + \sum_{j=1}^{B_l} a_i^l w_{j,k}^{l,l+1}} R_k^{l+1} \tag{2.8}$$

where $i \in \{1,...,B\}, l \in \{1,...,L-1\}$, then B is the number of units(neurons) in the specific $l-th$ layer and L is the total number of layers. By using this technique, the end-user can comprehend the input features that played a crucial role in deciding the model prediction for a given sample.

2.4 Anomaly Detection

2.4.1 Understanding the domain of Anomaly Detection

Anomaly detection is a process that involves identifying patterns in data that deviate from expected behavior. These deviant patterns can also be referred to as anomalies, outliers, discordant observations, exceptions, aberrations, etc. The references usually depend on the specific application domain, but anomalies and outliers are the two most commonly used terms in this context and are often used interchangeably. The importance of anomaly detection lies in its ability to identify and flag unusual patterns that may indicate a potential threat or problem, allowing for timely intervention and prevention of negative consequences.[79] So, the anomalous data can provide critical and actionable information, and examples of such anomalies include cyber-attacks in network security [80], credit card fraud in the financial industry [81], sensor anomalies in industrial settings, or anomalies in medical images such as optical coherence tomography (OCT) [82].

The study of detecting outliers or anomalies in data dates back to the 19th century, as early as the work of Edgeworth [83] in statistics. Since then, various anomaly detection techniques have been developed, some of them tailored to specific application domains while others are more generic.

What is an anomaly?

To begin with, it is imperative to establish a clear understanding of anomalies. Typically, data that conform to a specific distribution pattern are considered normal. In light of this, anomalies can be defined as follows:

An outlier is an observation that deviates so substantially from other observations that it prompts suspicion, that the given observation originated from a completely different mechanism altogether.

– Hawkins,[84].

In other words, anomalies represent patterns in data that do not adhere to a clearly defined concept of normal behavior. They frequently indicate serious situations, unusual events, or failures across a range of domains. Improper handling of anomalies can

lead to inaccurate conclusions, faulty decision-making, or even to wrong explanations, underscoring the significance of detecting and addressing them appropriately [79] [85].

Challenges within the anomaly detection domain

At a theoretical level, an anomaly can be described as a pattern that deviates from the expected normal behavior. An uncomplicated method for detecting anomalies is to establish a range that represents normal behavior and identify any observation in the data that falls outside of this range as an anomaly. This seemingly simple approach to anomaly detection is actually quite challenging due to several factors which were described in [86]:

- Defining a normal region is difficult because the boundary between normal and anomalous behavior is not always precise.
- Adversarial samples can make anomalous instances appear normal, thus the definition of what represents normal behavior gets complicated.
- Normal behavior can change over time, which makes defining normal behavior challenging.
- The definition of an anomaly varies depending on the domain in which it is being applied.
- Labeled data for training/validation of models used by anomaly detection techniques are usually limited.
- Noise in the data can be similar to actual anomalies, making it difficult to distinguish and remove.

Given the information, it is evident that anomaly detection in general is a complex problem that is not easily solvable. As a result, many existing anomaly detection techniques are designed to address specific formulations of the problem. Definitions like this are influenced by several factors such as the origin and characteristics of data or the types of anomalies to be detected. These factors are typically determined by the specific application domain in which the anomalies are expected to occur.

The ability to identify atypical behavior has significant benefits in a variety of applications. Anomaly detection methodologies can be categorized based on the data necessary for model training, such as supervised, unsupervised, or semi-supervised approaches. Given the rarity of anomalous instances in training datasets, unsupervised and semi-supervised settings are the most commonly used for anomaly detection [87].

Supervised deep anomaly detection

This approach entails the training of a deep supervised binary or multi-class classifier using labels of normal and anomalous data instances. For example, supervised deep anomaly detection (DAD) models formulated as multi-class classifiers aid in detecting rare brands, prohibited drug name mentions, and fraudulent healthcare transactions [88].

Unsupervised deep anomaly detection

In unsupervised anomaly detection, the training set typically consists of unlabeled data with infrequent anomalies, which aligns with the inherent nature of data distributions in real-world scenarios. Since anomalies are rare in actual situations, the one-class setting is a specific case of the unsupervised setting where all unlabeled training data are assumed to be normal samples.[86] [87]

Semi-supervised deep anomaly detection

Conversely, in the semi-supervised anomaly detection setting, there is an assumption of limited labeled normal and abnormal samples, along with a substantial number of unlabeled samples in the training dataset. Recent research has highlighted the effectiveness of utilizing a small number of anomalies during training, which can significantly enhance anomaly detection performance. Collecting anomalies can be difficult, and thus, a specialized semi-supervised setting called positive-unlabeled learning has emerged, which enables anomaly detection based on labeled normal and unlabeled samples. [86] [87]

In this thesis our main focus is on supervised pattern recognition being explainable so, we also aim for a supervised approach to anomaly detection.

Types of output for detection techniques

Another crucial factor in anomaly detection methods is the way how the outliers are highlighted. Typically, such methods produce outputs in the form of anomaly scores or binary labels. [87]

Labels - Certain techniques may assign a categorical label, designating each data instance as normal or anomalous. In unsupervised anomaly detection using autoencoders, the magnitude of the residual vector, or reconstruction error, is measured to obtain anomaly scores. Subsequently, domain experts rank or threshold the reconstruction errors to label data instances. [89]

Anomaly scores - indicate the degree of outlierness for each data point. The ranking of data instances can be determined based on their anomaly scores. A subject matter expert selects a domain-specific threshold, commonly known as the decision score, to identify anomalies. Decision scores generally provide more information than binary labels. For example, in the Deep SVDD approach [90], the decision score represents the distance between a data point and the center of the sphere. Data points that are farther away from the center are considered anomalous [91].

Furthermore, Anomaly detection models can be classified into shallow and deep models, depending on whether complex deep neural networks (DNNs) are utilized. While anomaly detection has been extensively researched for decades, numerous traditional machine learning models, i.e., shallow models, have been utilized for detecting anomalies, including support vector data description (SVDD)[90], and principal component analysis (PCA)[24] [92] or One-Class Support Vector Machines (OCSVMs)[93], etc... These models are generally efficient in identifying anomalies in tabular data, but they struggle with complex data types such as text, image, or graph due to their inability to capture non-linear relationships among features. In recent years, numerous deep learning models have been proposed for anomaly detection due to their ability to automatically learn representations or patterns from various types of data.

Deep and shallow anomaly detection approaches can be broadly classified into four categories[85][94]:

- **Density estimation and probabilistic models** identify anomalies by estimating the distribution of normal data. Both shallow and deep generative models, such

as Gaussian mixture models (GMMs)[95][96], variational autoencoder (VAE)[97], and generative adversarial networks (GAN)[98], have been employed for anomaly detection. Generative models approximate the data distribution based on a dataset, and deep generative models can model complex and high-dimensional data more effectively.

- **One-class classification models** are discriminative approaches to anomaly detection, which seek to learn a decision boundary based on normal samples. A classical one-class classification model, such as a one-class support vector machine (OC-SVM)[93], is commonly used for anomaly detection. Similarly, recently developed deep one-class classifiers, such as Deep SVDD[90], are employed to identify anomalies in complex data.

  In this thesis, we are mainly focused on a multi-class approach because with the boundaries of normal samples for each class is possible to achieve more precise detection in contrast to the one-class approach.

- **Reconstruction models** are trained to minimize reconstruction errors on standard samples, enabling the detection of anomalies with large reconstruction errors. Anomaly detection models based on reconstruction include traditional PCA or Deep Autoencoder models [99], which are parameterized by a deep neural network for encoding and decoding data.

- **Miscellaneous techniques**, have also been developed for anomaly detection. For instance, methods like Isolation Forest [100]. This approach constructs trees from unlabeled samples and identifies anomalies in samples located near the trees' roots. Local Outlier Factor (LOF)[101] is another unsupervised anomaly detection method. This method calculates the local density deviation of a given data point with respect to its neighbors and labels points with lower density than their neighbors as anomalies [94]

There is also a large number of other miscellaneous DAD techniques and their main key concepts are shown to be effective and promising.

Clustering-based anomaly detection

In the realm of anomaly detection, clustering has emerged as a promising technique. Numerous algorithms for detecting anomalies based on clustering have been proposed in

the literature [102]. By grouping similar patterns based on extracted features, clustering can effectively detect new anomalies. However, the time and space complexity of clustering grows linearly with the number of classes to be clustered [103]. As a result, clustering-based anomaly detection may not be practical for real-time applications. To address this limitation, feature extraction within the hidden layers of deep neural networks can reduce the dimensionality of input data and ensure scalability for complex and high-dimensional datasets. In the deep learning-enabled clustering approach to anomaly detection, techniques such as word2vec [104] models are used to obtain the semantical representations of normal data and anomalies, which are then clustered to detect outliers [105] [87].

Ensemble-based anomaly detection

Deep neural networks have a significant drawback in that they are vulnerable to noise in input data and typically require a substantial amount of training data to ensure reliable performance, as observed in paper [106]. To overcome this limitation and enhance robustness in noisy data, a novel strategy involves introducing random variations in the connectivity architecture of autoencoders, which has been demonstrated to yield significantly improved results. Authors behind the paper [107] conducted experiments using autoencoder ensembles composed of diverse randomly connected autoencoders and obtained encouraging outcomes on various benchmark datasets. [88]

Deep Reinforcement Learning based anomaly detection

The capacity of Deep Reinforcement Learning (DRL) techniques to acquire complex behaviors in high-dimensional data space has garnered considerable attention. To this end, the papers [108] and [109] have proposed approaches for anomaly detection utilizing deep reinforcement learning. Notably, the DRL-based anomaly detector operates without relying on preconceived notions regarding the nature of the anomaly. Rather, it continually enhances its knowledge by leveraging reward signals accumulated over time, enabling the detector to identify novel anomalies consistently.

To gain a more comprehensive understanding of the anomaly detection domain, these sources [110] - [114] address the issue of outliers in more depth.

2.4.2 Anomaly detection using Deep Generative Modeling

While several anomaly detection methods perform well on low-dimensional problems, there is a significant dearth of effective techniques for high-dimensional spaces like images. However, in the domain of deep generative modeling, this is not the case. Generative models are designed to learn the exact data distribution, enabling the creation of new data points with some degree of variation. Two of the most widely used and efficient generative approaches are Variational Autoencoders (VAE) [97] and Generative Adversarial Networks (GAN) [98]. There is also a more customized version of generative models known as Adversarial autoencoders (AAE) [115]. The ability of AAEs to learn input distributions has been leveraged in several Generative Adversarial Networks-based Anomaly Detection (GAN-AD) frameworks [116] - [120]. Nevertheless, in cases with fewer anomalies, conventional techniques such as K-nearest neighbors (KNN) have been demonstrated to outperform deep generative models [121].

As previously stated, the interpretability of anomaly detection models is highly significant. Nevertheless, it's not the only factor as the comprehensibility of deep learning as a whole relies on this matter. Because of this, the work aims to introduce a new approach to anomaly detection by utilizing deep generative modeling. The fundamental principle of outlier detection using these models involves training a model to recognize normal behavior and identify anomalous activity if there is any deviation from normality (same concept across the field). [87] However, in Deep Generative Modeling (DGM) the majority of models designed for detecting anomalous activity belong to the encoding-decoding architecture group. The encoder learns to generate a latent representation of the input data, while the decoder reconstructs the original input data based on this representation. Although techniques may vary across different architectures, the primary advantage of these systems is their ability to learn the underlying distribution of original data samples and highlight anomalous samples based on the acquired knowledge.[88]

Autoencoder as anomaly detector

Autoencoders are trained to minimize their reconstruction error, which is calculated by measuring the difference between the original input data and the reconstructed output sample[94].

$$L_r(G_m,\tilde{G}_m) = \| G_m - \tilde{G}_m \|^2 \tag{2.9}$$

In practice, autoencoders are often used as a dimensionality reduction technique (like the non-linear PCA). Though pseudo-generative models, like Autoencoders (AEs), do not produce genuinely generative outputs, they still belong to the category of generative models. This is attributed to their capability to generate output that bears resemblance to the input data, although the process involves reconstruction rather than true generation. They can also be used in the extraction of features from unlabeled data and image-denoising tasks. It is crucial to understand that the mapping function of the autoencoder is tailored to the distribution of the training data. As a result, autoencoders may not be able to reconstruct samples that differ significantly from the original data with which they were trained (the reconstruction is very poor). This capability of learning a mapping function that is specific to a particular distribution is highly advantageous for anomaly detection tasks.

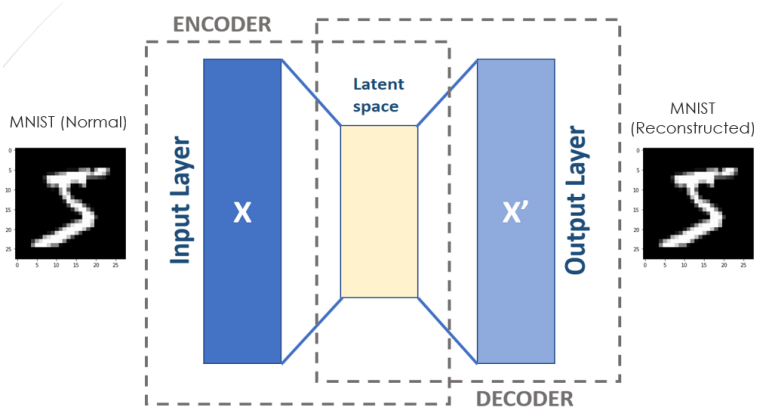


Figure 2.7: Reconstruction of the MNIST images by AE

When applying an AE for anomaly detection, the general approach is to first model normal behavior and then generate an anomalous evaluation for new data samples. In order to actually model normal behavior, an unsupervised or semi-supervised learning approach is used to train the AE on the original data. This enables the model to learn a mapping function that can successfully reconstruct the training data with a very small reconstruction error. During the detection-testing phase, the reconstruction error is used as an anomalous score for different data samples. To identify anomalies, the reconstruction error is compared to a certain threshold. If the error is greater than the set threshold, an anomalous sample is detected. The process of identifying an outlier using an autoencoder is depicted in figure 2.8. If the autoencoder attempts to reconstruct the outlier, it fails to do so because the data was either not present in the

training dataset at all (or was represented poorly). As a result, the reconstruction error surpasses a pre-defined threshold, designating the given data sample as an anomaly.[89]

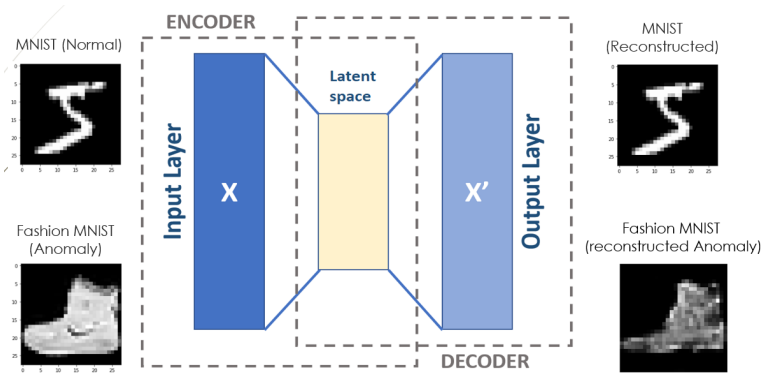


Figure 2.8: Anomaly detection according to anomaly score(Reconstruction Error)

Different types of autoencoder architecture exist as depicted in Figure 2.9 designed for anomaly detection applications. The selection of an appropriate architecture depends on the nature of the data being analyzed, where convolutional networks are typically employed for image datasets, while models based on long short-term memory (LSTM) [122] tend to yield satisfactory outcomes for sequential data. Recent studies have shown that combining convolution and LSTM layers, with a convolutional neural network (CNN)[59][58] serving as the encoder and a multilayer LSTM network as the decoder, can effectively detect anomalies within data by reconstructing input images. The use of hybrid models, such as Gated Recurrent Unit Autoencoders (AE-GRU) [123], Convolutional Neural Networks Autoencoders (CNN-AE) [124], and Long Short-Term Memory Autoencoders (LSTM-AE)[125], obviates the need for manual feature engineering, facilitating the use of raw data with minimal preprocessing in anomaly detection tasks [87].

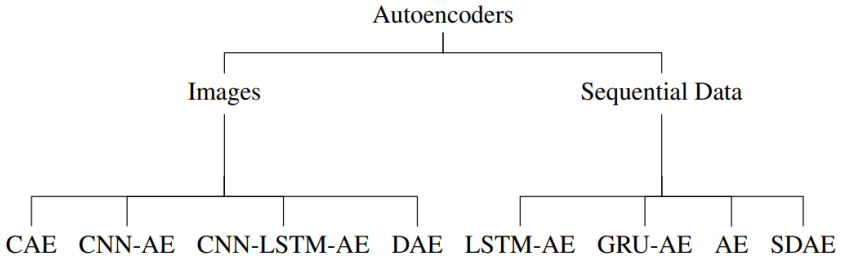


Figure 2.9: AE architectures for anomaly detection cause [87]

Variational Autoencoder as anomaly detector

The Variational Autoencoder (VAE) [97] is an unsupervised deep learning generative model that can effectively model the distribution of the training data. This model is derived from Bayesian inference and comprises three main components: an Encoder, a latent distribution, and a Decoder. The underlying principle is based on a simple distribution, such as the Gaussian distribution, with known parameters and superimposable characteristics that can be combined with neural networks to theoretically fit any distribution. Both the Autoencoder and the Variational Autoencoder can capture the essential features of the data through dimensionality reduction. However, the VAE is unique in that it is a stochastic generative model that can provide calibrated probabilities, unlike AE. In contrast, the AE is a deterministic discriminative model that lacks a probabilistic foundation. Its forward propagation process involves the input sample X passing through the Encoder to obtain the parameters of the latent space distribution. Subsequently, the latent variable z is generated by sampling from the current distribution, and z is utilized to reconstruct a sample via the Decoder[97]. The loss function of VAE can be expressed as follows [79]:

$$L_{vae}(E,D) =\| X - X_{recon} \|^2 + \frac{1}{2} \sum_{i=1}^{z_{dim}} [(\mu_i^2 + \sigma_i^2) - 1 - \log(\sigma_i^2)] \tag{2.10}$$

During the forward propagation stage of a Variational Autoencoder, sampling from the distribution is a crucial step. However, the act of sampling is non-differentiable, making it impractical to incorporate into the VAE algorithm. As such, VAE uses a method known as "re-parameterization" to facilitate the sampling process. By utilizing the properties of the Gaussian distribution, the resultant samples are equivalent to those obtained directly from the distribution corresponding to the specific distribution parameters. [126] In the case of the entire VAE model, the sample derived from the standard normal distribution can be considered a constant. This constant can be calculated through a differentiable process, enabling the entire model to perform standard backpropagation. [79]

$$\epsilon \sim N(0,1) \quad z = \mu + \epsilon * \sigma \rightarrow \quad z =\sim N(\mu,\sigma) \tag{2.11}$$

The implementation of a Variational Autoencoder (VAE) as an anomaly detector involves training the model on a dataset of normal data and subsequently employing it

to reconstruct new data points. This approach relies on the assumption that the VAE will perform well in reconstructing normal data, but may encounter difficulties when processing anomalous data points. By measuring the degree of reconstruction error, it is possible to evaluate the degree of anomalousness of a given data point.

Adversarial Autoencoders as anomaly detectors

Adversarial Autoencoders (AAE) architectures [115], overcome the main limitation of VAEs, which is an absence of a closed-form analytical solution for the integral of the Kullback-Leibler divergence (KLD) term[127]. It is done by utilizing adversarial learning to learn a wider range of distributions as priors for the latent code.[126]

In this architecture, an adversarial autoencoder comprising of an encoder f_ϕ and a decoder g_ψ is utilized for the training process. To begin, the encoder (generator) network $f_\phi(z|G_m)$ creates a latent representation z which is then used by the decoder to reconstruct the input $\hat{G}_m$. The weights of both the encoder f_ϕ and the decoder g_ψ are updated by backpropagating the reconstruction loss between $\hat{G}_m$ and G_m. Following this, the discriminator is presented with two inputs: z, which is distributed as $f_\phi(z|G_m)$, and z', which is sampled from the true prior $P(z)$. The discriminator then calculates a score for each input ($D(z)$ and $D(z')$) and minimizes the loss incurred by backpropagating through the discriminator to update its weights. The loss function for the autoencoder L_E consists of both the reconstruction error and the loss incurred by the discriminator L_D. [115]

$$L_E = \frac{1}{M'} \sum_{m=1}^{M'} \log D(z_m) \quad \textit{and} \quad L_D = -\frac{1}{M'} \sum_{m=1}^{M'} [\, \log D(z'_m) + \log(1 - D(z_m)) \,] \tag{2.12}$$

where z represents latent representation created by Encoder, z' is sample from the true prior $P(z)$ and M' is the batchsize.

Variational Encoder for Anomaly Detection with Stability Control

The above-mentioned techniques are built upon the generative model concept, which assesses abnormalities through the comparison of data errors between the original samples and reconstruction samples. One of the well-known architectures is the VAE. However, it suffers from the problem of over-generalization. To address this issue, the Variational Encoder for Anomaly Detection with Stability Control (VESC) [79] model has been introduced as an unsupervised deep learning anomaly detection method that

proposes the recursive reconstruction strategy. VESC model adopts the concept of data compression based on the original model. The recursive reconstruction strategy improves the accuracy of the model by increasing the count and normality of training samples. This strategy can be applied to most unsupervised learning methods.
The model comprises four components [79] [97]:

- The encoder and decoder of the original VAE, are primarily employed for dimensionality reduction coding of the original sample to produce a latent vector and subsequently decode the reconstructed sample.
- The Spatial Constrained Network (SCN), builds upon the knowledge that the original VAE model has learned the normal data distribution effectively. SCN extracts the feature representation vector of the latent space through clustering and then leverages it to reform and re-represent the latent vector.
- The Reformer structure: After the SCN acquires the feature vector, the Reformer structure maps abnormal samples to normal reconstruction samples. As a result, the reconstruction error increases, and the abnormality becomes more prominent.
- The Re-Encoder: It has better modeling capabilities for normal data. This complex architecture can optimize the reconstruction error in both the original and latent spaces to model the normal samples accurately. Furthermore, it can calculate the anomaly score in both feature spaces (i.e., the original space and latent space), resulting in higher accuracy than considering only the original space.

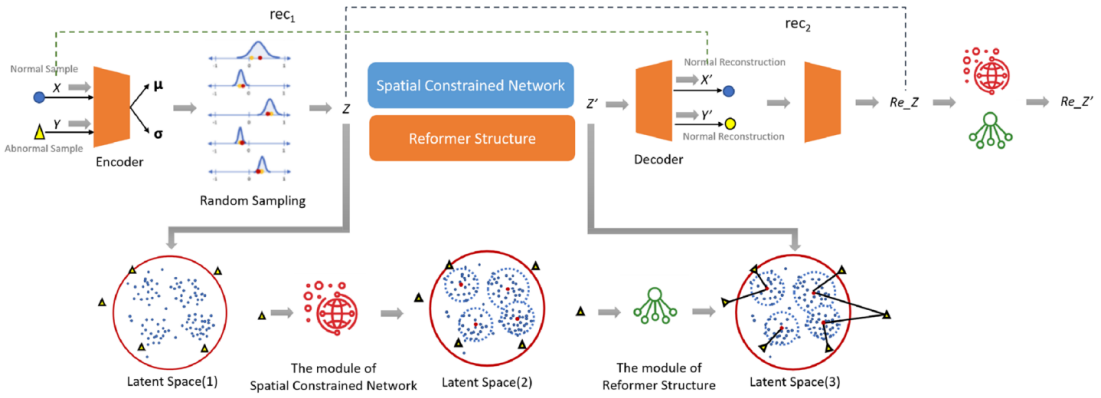


Figure 2.10: The concept of VESC model[79]

The initial stage of the training process is identical to that of VAE training. The model is trained using regular samples, allowing it to grasp the distribution of normal data in the latent space. The Spatial Constrained Network is then employed to learn the distribution of normal data in the latent space and derive a set of feature vectors C_i. Subsequently, vector Z is transformed and re-represented as a new vector Z' through a sparse mechanism in the Reformer structure. This transforms all data into "normal data," which is then decoded by the Decoder into a reconstructed sample X' in the original data space. The model is trained by minimizing the reconstruction error and updating the loss iteratively.

The cost function is composed of three distinct parts (different loss functions). The first part is the original VAE loss function, which has been previously described (refer to Figure 2.10). The second part of the loss function pertains to the Re-Encoder. Building upon the principles established by the original VAE loss function [97], it has been devised a loss function specific to the Re-Encoder network, which mirrors the second term of the original VAE loss function.

$$L_{KL\_2} = \frac{1}{2} \sum_{i=1}^{z'\_dim} [\ (\mu'^2_i + \sigma'^2_i) - 1 - \log(\sigma'^2_i)\ ] \tag{2.13}$$

The final part is the error associated with the latent vector. To calculate this error, the Re-Encoder remaps the generated data to the new latent space and then determines the L2 distance between the resulting latent vectors [79]:

$$L_{lat} = \| \ Z' - Re\_Z' \ \|_2 \tag{2.14}$$

In summary, the overall loss function for the entire model can be described as follows:

$$L_{VESC} = L_{vae} + L_{KL\_2} + L_{lat} \tag{2.15}$$

Anomaly Detection with Generative Adversarial Network (ADGAN)

Generative Adversarial Networks (GANs) [98] or their slight modification Conditional GANs [128] are neural networks designed to train a generative model based on the distribution of input data. In the classical formulation of GAN networks, the generator learns to model the underlying data distribution of input data x by transforming random noise z into the distribution of x. For anomaly detection purposes, it is possible to use again, the intrinsic ability of the Discriminator model. The Discriminator might behave

as a critic of samples. During training, he learns the data distribution of the training set as well as the generator does (In fact, the generator uses the feedback from the discriminator to optimize its output). [129] During the training process, the generator is instructed to map low-dimensional random samples to a high-dimensional space, which emulates the target dataset. If the generator has successfully acquired a precise approximation of the training data's distribution, it is reasonable to assume that there exists a corresponding point in the GAN's latent space for any sample drawn from the data distribution. Subsequently, by passing this point through the generator network, a sample closely resembling the original can be generated. If the input sample deviates from the learned data distribution (could be an authentic fake from the generator or a completely different sample), the discriminator as a critic highlights the given sample as an outlier. [89] [130]

It is crucial to comprehend the fact, there is no straightforward way to use this knowledge for controlled inference similar to the autoencoder process (such as generating a sample similar to a specific input data sample). Although we can search the latent space to recover the most representative latent noise vector for a particular sample, this process is computationally intensive and slow.

Bidirectional Generative Adversarial Network as anomaly detector

A slight modification to the GAN architecture that involves adding an additional neural network - the encoder - allows for efficient management of adversarial inference, meaning another approach how to detect outliers. Such a network is called BiGAN and is proposed in the following paper [131]. During the training phase of a BiGAN network, the added encoder learns the inverse mapping of the generator - that is, it learns to generate a fixed vector $E(x)$ from the input data x. This modification also affects the discriminator network, which takes pairs of data containing both generated and real data, as well as their respective latent representations, $G(z)$,z and x,$E(x)$. During training, the encoder is trained alongside the generator. The generator learns the true distribution of the input data x by generating $G(z)$ samples from the latent noise vector z, while the encoder learns the true distribution for the latent representations $E(x)$ of the input data x. [130]

The transformations learned by models during the training process are always specific to the training data. For instance, if we train a generator on a data set of

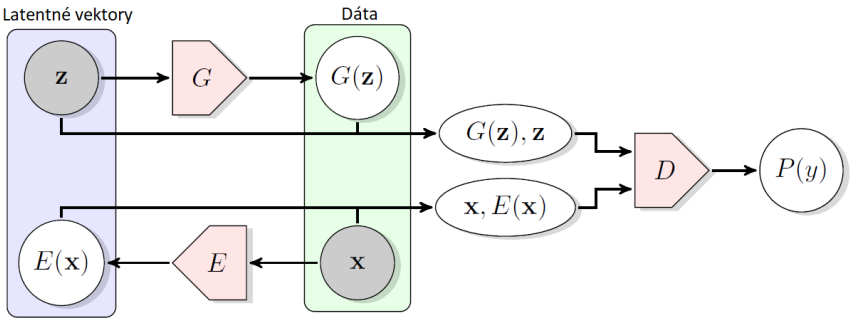


Figure 2.11: Architecture of BiGAN network[131]

cars, the generator will always create an image that looks like a car, regardless of the input noise vector (latent variables) that we provide to the model. When modeling normal behavior, we train the network on a training data set. After the learning process is complete, we have an encoder that can map input samples to a latent vector, a discriminator that can distinguish real data from generated data, and a generator that can transform a latent code (noise vector of fixed length) into an authentic sample. To evaluate data for anomalies, we need to establish a metric for detecting them. To detect outliers using BiGAN, the encoder initially generates a latent representation $E(x)$ of the input sample x. The given latent code then passes to the generator, which produces an image $G(E(x))$ based on the latent representation. The outlier can be identified by comparing the reconstruction errors between the input sample x and the generated sample $G(E(x))$. Additionally, we can consider the discriminator error, which is the cross-entropy between the data sample x and $G(E(x))$.

In the preceding sections, we have discussed various deep anomaly detection techniques, each with its own advantages and limitations. To effectively address an anomaly detection problem, it is crucial to discern which technique is most suitable for the specific context. However, it is worth noting that DAD is an area of active research, and therefore, it is impractical to provide a definitive understanding of the most appropriate anomaly detection technique for every scenario.

Chapter 3

Developed Explainable Approach: Proposal

3.1 Background and Motivation

Explainability is a fundamental requirement for a scientific approach to problem-solving. The epistemological triangle [132] expresses its basic principle (Figure 3.1). Vertices of the triangle may have different interpretations (see also the term "semiotic triangle" [133]). Other variants of the "world - knowledge - model" chain mentioned here are "thing - logos - states of mind" (Aristotle), "world – cognitive processes – description" (Foerster), or "object - feature - interpreter" (Pierce).

In simple terms, we extract knowledge from the environment in the form of features and then use logical reasoning to construct a model of reality. Based on this model, we make decisions. The accuracy of these decisions depends on the accuracy of the model. With the advent of machine learning, this process has become more direct, allowing us to identify model parameters directly from data. However, if the transformation is non-linear, as in DNNs, the user may not have visibility into how the model makes its decisions. Replication of human thinking and aligning the neural network's decision-making process with the epistemological triangle is imperative. This has been achieved implicitly in the advancement of DNNs.

These networks designed for pattern recognition consist of two components:

- Feature extraction.
- Decision generation (e.g. classification)[134].

This division requires the explainability of both, extracted features and classification. The features may be straightforwardly explicable by domain experts, and in certain cases, they are obtained by pre-processing the data and utilized as inputs to the neural network.

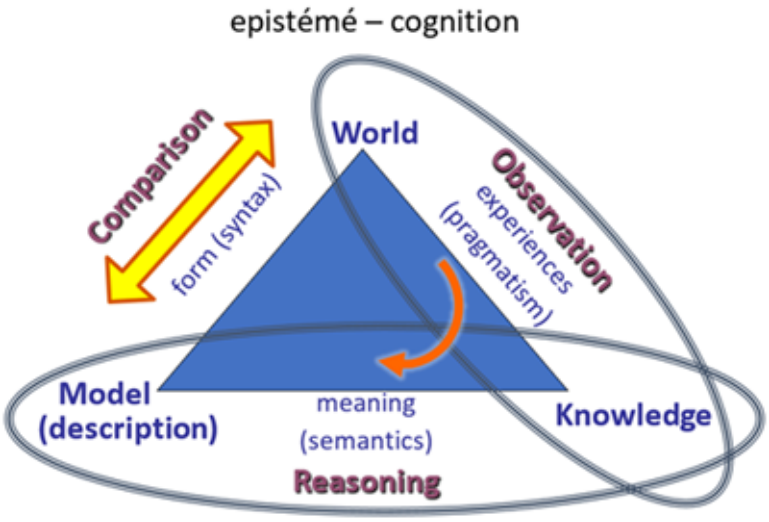


Figure 3.1: Epistemological Triangle as the scientific approach

The transformation of knowledge into features is often referred to as feature engineering. An illustration of this could be sound processing, where it is known that due to the shape of the basilar membrane in the inner ear, a person perceives sound in the spectral domain. As a result, a spectrogram of the sound is used directly as input to the neural network. Another possibility is to approximate the features of objects (e.g., the age of the person in a face image - face aging [134]). Then, the neural network can be trained to locate features in a particular part of the latent feature space. If these features are used for image reconstruction [135], the position of the features influences the reconstructed image [136]. See [137] for facial aging.

It is not even required to use the features extracted from the input data to generate patterns; instead, a generative adversarial network (GAN) can be trained to produce patterns from random coordinates of a feature vector within a defined subspace of the latent space [138] (e.g., refer to [139] for face aging).

Practical tools that enable the extraction of features in a manner recognizable to the end user imply their explainability. However, we aim to leverage the advantages of non-linear pattern recognition systems, which require acknowledging the extracted features are unfamiliar to the users and thus not immediately interpretable. The main idea is that the explainability of the features lies in the knowledge acquired from the extracted features, a process akin to extracting knowledge from data. Therefore it follows a similar scientific methodology. The feature extractor obtains features from the application viewpoint, meaning it can identify features that would otherwise remain unnoticed due to human biases shaped by past experiences. This process refers to the search for model parameters through machine learning techniques. However, the design of the model's architecture and the selection of its hyperparameters still fall under human control.

Additionally, there is experience in interpreting the more complex features produced by linear extractors, such as those obtained through Principal Component Analysis. Although the features are derived from linear combinations of the input data, resulting in uncorrelated features. For stationary random processes, the eigenvectors of the transformation matrix consist of harmonic functions. Hence, the features represent the power spectrum of the random process. As a result, these spectral features have become a widely used tool for time series analysis, not only in the case of sound but also in mechanical and economic systems.

In this context, explainability is not perceived as a fully automated process that provides explanations to even an uninterested audience (non-experts). Rather, it is regarded as the process of gaining knowledge from the features generated by a specific pattern recognition system (as depicted in Figure 3.1). This approach may result in a narrow specialization, consistent with the trend of specialized scientific disciplines. For example, AlphaGo's analysis of a move in the game of Go [140], which was previously unknown to players but crucial in its victory against Lee Sedol, serves as a demonstration of this concept. The transformation of features into knowledge allows the user to learn from the computer. Additionally, by examining the features, we can conclude that while a feature may have contributed to the recognition of an image, it may not necessarily have a direct connection to the object being recognized (known as the Clever Hans effect) [21], [141].

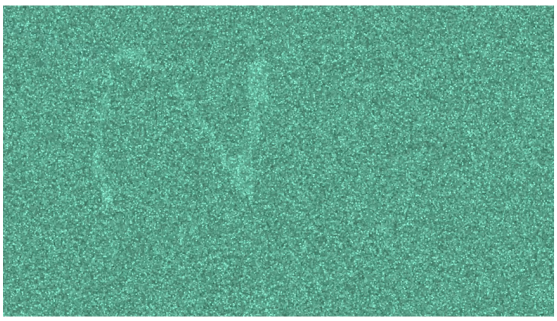
Thus, the goal for the user is not to identify which features were extracted from the input data but rather to understand the reasoning behind the classifier's decision based on these extracted features. For this reason, we advocate for the replacement or supplementation of commonly used non-linear classifiers with linear systems or logical rules. The proposed approach to resolving the explainability problem in this work involves the utilization of post-hoc support.

Figure 3.2: The concept of explainable machine learning created according to the principles of the scientific approach

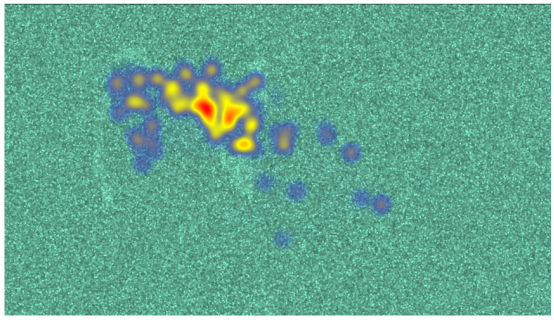
3.2 Introduction: How to Learn from Computers

In the field of non-linear decision-making systems, it is crucial to separate the system into two components for the purpose of explainability: feature extraction and prediction inference. The proposed method aims to examine the explainability of these components separately, rather than trying to explain the decision system as a whole. The literature on recognition systems' explainability highlights two main approaches. The first approach focuses on preserving explainability by using self-explaining systems, also known as explainability by design[142] [144]. This approach involves replacing a part of the network, such as a classifier, with an explainable system, such as a linear system or a logic function. The second approach, known as post-hoc, involves building an explainable system in parallel to the existing system, allowing for a possible explanation of the results achieved, at least locally [145], [146]. The proposed method is based on a post-hoc approach that focuses on explaining the feature extractor and the inference system separately. This approach allows us to extract features that are not known to current science. The subsequent understanding of the extracted features is like knowledge extraction from data, which must occur in the user's mind. The explanation of decisions made from these features can be based on Aristotelian (or fuzzy) logic or linear systems. Paper [147] investigates the implementation of a classifier using fuzzy logic. The designed method has a classifier based on fuzzy logic, in parallel to the neural network classifier, and restricts the description of the solutions to explainability in forwarding DNNs for pattern recognition since the proposed solution is also of this type. When explaining the decisions made by a fuzzy logic function, we take into account not only the value of the features but also the truth values (significance) of these features for the decision.

In the field of Deep Learning, the overall explainability of the entire system is a common topic in the literature. However, some papers have also explored the importance of the input data, such as pixels in an image [78], [148], as well as the internal variables of a DNN, such as features [149]. Although the methodology and terminology used in these studies are diverse, including saliency maps [69], [150], attribution maps [151], attention maps [152], [153], sensitivity maps [154], and Shapley additive explanation [155], they are fundamentally similar to sensitivity analysis of parametric systems. The results of pixel-attribution maps can also be replicated in humans using eye-tracking systems.



(a)



(b)

Figure 3.3:
(a) Handwritten digit 2 with a left rotation and heavy Gaussian noise,
(b) The eye-tracking record during the digit recognition process.

As an example, figure 3.3 presents the results of the thesis supervisor's recognition of the handwritten digit 2. The recognition process is challenging due to the random positioning, rotation, and heavy Gaussian noise applied to the digits. This results in a recognition process that requires effort and concentration. For comparison, the results obtained through the Layer-wise Relevance Propagation (LRP) method are presented in this source [156] (the parameters used are: LRP Epsilon formula, Long ReLu model, and Beta value of 1). Figure 3.4a) shows a digit 2 selected from the MNIST database. In figure3.4b), the shape of digit 2 was replicated, similar to the eye-tracking experiment shown in figure 3.3. However, since MNIST does not contain rotated digits, the recognition system misidentified the pattern as digit 4. This serves as a cautionary example highlighting the importance of including an anomaly detector to filter out patterns that are distinct from those in the training set before attempting to explain any incorrect decisions. The explainable pattern recognition system should also be capable of detecting misrecognition. As stated in [157], the explanation can be achieved through either an approximation or an example-based approach. In the

approximation approach, several patterns with similar features to the target pattern are selected, and a common explainable decision subsystem is derived for them using linear or logical systems. This approach is often employed during the training phase to ensure that similar patterns have the same explanation.

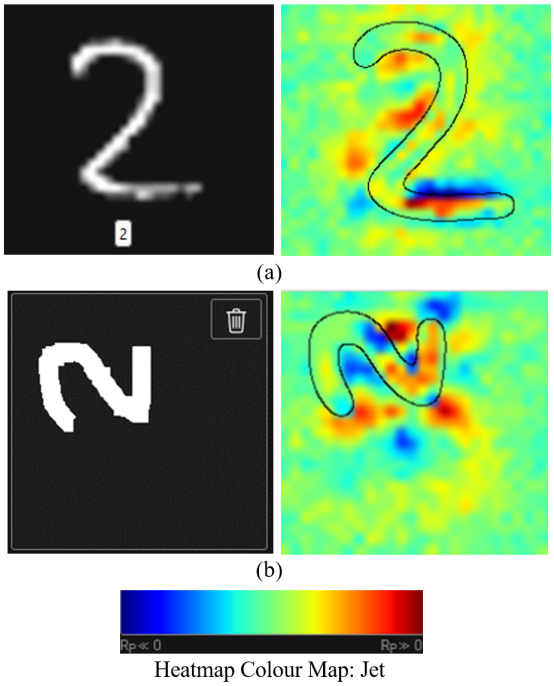


Figure 3.4: Heatmaps achieved by demo in [156].

3.3 The concept of the proposed method: Overview

This work introduces a tool designed to enhance the user's understanding of patterns and their impact on the recognition of classes. The tool is an explainable classifier in the form of a fuzzy logic function that provides consistent explanations for a subset of patterns belonging to a single class. In the example-based approach, we examine samples with similar features either from the same class or from a different class (counterfactual) compared to the pattern being explained. This approach is used during the recognition and explanation phase of a new pattern. When explaining features, the user can experiment with variations in feature values and observe the impact on both the decision and the corresponding pattern. This provides a visual representation of the logic function used for classification explanation, which produces the same result as a non-linear classifier.

As an illustration, we have chosen to examine the explainability of handwritten digit recognition from the MNIST database [158]. The block diagram in figure 3.5a) displays the structure of the handwritten digit recognition system, which consists of a feature extractor, a feature explanation interface, and a fuzzy logic function serving as an explainable component for the non-linear classifier. Figure 3.5b) presents the user interface.

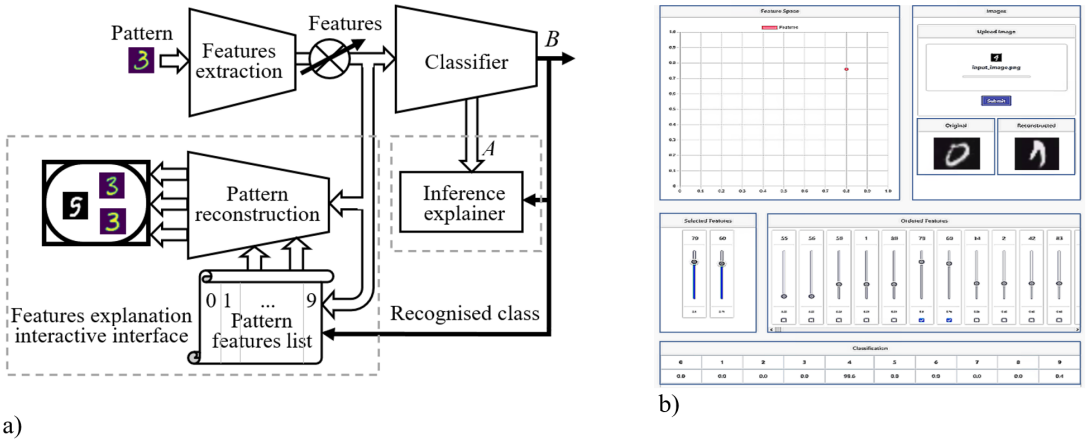


Figure 3.5: Concept for proposed explainability approach.
a) The block diagram.
b) The inference explanation interface.

The enhanced LeNet-5 neural network performs both feature extraction and classification [159]. The gradient of the classifier's output is backpropagated to the feature layer, enabling us to calculate the feature importance value, represented as $\mathbf{y} \in R^M$, which serves as a measure of the features' relevance in the decision-making process [78].

As previously discussed, we contend that it is imperative to maintain the non-linearity of the feature extractor, even if the features are not immediately explicable and the explanation process is analogous to discovery through observation of events. Our perspective aligns with Galileo Galilei's experimental approach to acquiring knowledge, as opposed to Plato's idea of passive observation. For this post-hoc approach, we have developed a tool, the block diagram of which is depicted in Figure 3.5a) (modification of [160]). The pattern recognizer, consisting of the feature extractor and the classifier, is tested. The experiment involves variations in the features and observation of their impact on the reconstructed image, the classification result, and the relevance of the features. The decoder model, which reconstructs the image using the extracted

features, is trained while the compressor (feature extractor part of the classifier) remains fixed. The graphical interface presents a 2D subspace of the feature space, displaying the original image and the image with altered features. The classifier's explanation is facilitated by shading the sliders that control the relevant features entering the codeword (refer to equations 3.8, 3.10).

3.4 How to measure the significance of features

In this work, we posit that human knowledge in the cognitive process corresponds to the extracted features in the machine learning process (see Figure 3.2). In the preceding subsection, we introduced a method of teaching the user to explain these extracted features. The following subsections describe two ways to use the features to explain pattern classification. We investigate two approaches for utilizing the features $f_i, i \in \{1,...,M\}$ in determining the classified class through the inference explainer:

1. Using them directly, as done by the non-linear classifier, where $y_i = f_i$.
2. Utilizing the features' relevance measure, where $y_i = \rho(f_i) = R_i$.

Regarding the **2nd point** - Utilizing the feature relevance measure we already mentioned a considerable amount of literature that assesses the significance of variables in neural networks, which we apply to features. In order to test the proposed method, we have used two exemplary techniques to calculate feature relevance: a gradient-based backpropagation approach and a feature-relevance approach [161],[162].

As a representative of the feature-relevance approach, we use Layer-Wise Relevance Propagation [78]. In this method, the target output value represents its relevance R. We assume that the classification neural network has L layers (L - number greater than 2) and consists of neurons $n_i^l, l \in \{1,\ldots,L\}, i \in \{1,\ldots,B_l\}$, where B_l is the number of neurons in the $l-th$ layer. Now, let's assume $w_{i,k}^{l,l+1}, l \in \{1,\ldots,L-1\}, i \in \{1,\ldots B_l\}, k \in \{1,\ldots B_{l+1}\}$ the weights between neurons n_i^l and n_k^{l+1}, next $a_i^l, l \in \{1,\ldots,L\}, i \in \{1,\ldots B_l\}$, outputs from neurons for the tested pattern ($a_i^1 = y_i, B_1 = M$ are the features), then $R_i^l, l \in \{1,\ldots,L\}, i \in \{1,\ldots B_l\}$ is the relevance of outputs from neurons for the tested pattern, so $R_{i\leftarrow k}^{l,l+1}, l \in \{1,\ldots,L-1\}, i \in \{1,\ldots B_l\}, k \in \{1,\ldots B_{l+1}\}$ is relevance transfer from neuron n_k^{l+1} to neuron n_i^l. At the output layer, we consider

only the winning neuron, which determines the confidence of the winning class (i.e., $B_l = 1, R_l = a_1^l$). Backward computing of relevance is defined by:

$$R_i^l = \sum_{k=1}^{B_{l+1}} R_{i\leftarrow k}^{l,l+1} = \sum_{k=1}^{B_l+1} \frac{a_i^l w_{i,k}^{l,l+1}}{\epsilon + \sum_{j=1}^{B_l} a_i^l w_{j,k}^{l,l+1}} R_k^{l+1} \tag{3.1}$$
$$i \in \{1, \dots B_l\}, l \in \{1, \dots, L-1\}$$

At last, the result is feature relevance $R_i^1, i \in \{1, \dots, M\}$. As a representative of the gradient-based approach, we choose the Vanilla Gradient method to display Saliency Maps [69] because it shows very nicely the general recipe that other methods follow. Let $\mathbf{f} = (f_1, \dots, f_M)$, is the feature vector, $f_i = a_i^0, i \in \{1, \dots, M\}$ is the input value to the classifier for the given pattern and $a_i^L, i \in \{1, \dots, N\}$ is the output value of the classifier for the class i (probability that the pattern belongs to the class i). The feature class score (the saliency) S_i^j of the feature f_i for the specific class j is given as partial derivation:

$$S_i^j = \frac{\delta a_j^L}{\delta f_i}\Big|_{f_i = a_i^0}, i \in \{1, \dots, M\}, j \in \{1, \dots, N\} \tag{3.2}$$

In both cases, the raw feature or its computed relevance needs normalizing to a unit interval to interpret its value as the truth value of the fuzzy logic statement [147]. For the values, we normalize $y_i \in \mathbb{R}, i \in \{1,...,M\}$, and use min-max linear normalization:

$$\begin{aligned} \tilde{y_i} &= \frac{y_i - y_{min}}{y_{max} - y_{min}}, \\ y_{min} &= min\{y_1, \dots, y_M\}, \\ y_{max} &= max\{y_1, \dots, y_M\} \end{aligned} \tag{3.3}$$

There is also an option to normalize features with the use of the Sigmoid function:

$$\tilde{y_i} = S(y_i) = \frac{1}{1 + e^{-y_i}} \tag{3.4}$$

The sigmoid function applies a nonlinear transformation to the features, mapping them to a range between 0 and 1 based on a curve that asymptotically approaches these limits. While this method can be effective for some applications it is not the best approach for fuzzy logic, since it may distort the relationships between the features, making it difficult to accurately compare the degree of truthfulness of the resulting output. Although, Min-max normalization rescales the features linearly to the range between 0 and 1 based on the minimum and maximum values of the input set. This

method preserves the order and distance between the values of the input set, which is important for the implemented fuzzy Zadeh function, which applies max, min functions for disjunction, conjunction (function of truthfulness)[163] [164]. Thus, the min-max normalization allows for a more flexible and nuanced representation of information, which can be useful in decision-making.

In the context of fuzzy logic, we employ one important term - **the degree of truthfulness**. This term refers to the extent to which a statement is true or false. In fuzzy logic, a fuzzy proposition or rule is defined by a function of truthfulness, which assigns a degree of truthfulness to the proposition. The degree of truthfulness ranges from 0 to 1, where 0 means the proposition is completely false, and 1 means the proposition is completely true.

3.5 Fuzzy Explanator as a Classifier

In the previous section, we describe how to measure the significance of features and their subsequent normalization for the Zadeh fuzzy function. Such pre-processing of data is the first step in our proposed architecture as we can see in Figure 3.6 - Normalization module.

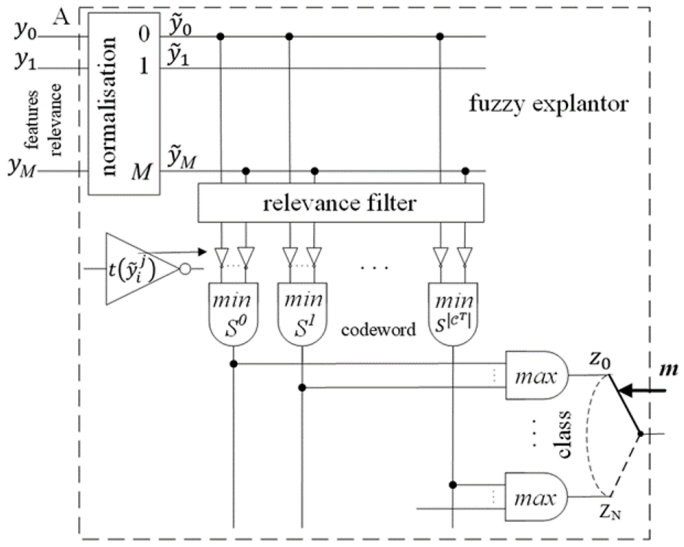


Figure 3.6: Architecture of Logical Fuzzy Classifier

The next phase involves the relevance filter module, which transforms the features into corresponding relevance states. Positive relevance refers to the degree to which a

particular input feature or set of features is important for making accurate predictions in a deep learning model. These features are considered to be positively correlated with the model's predictions and are essential for the model's performance. Conversely, negative relevance refers to the degree to which a particular input feature or set of features is detrimental to the accuracy of predictions. These features are considered to be negatively correlated with the model's predictions and can cause the model's performance to degrade. Overall, both positive and negative relevance is important for interpreting and explaining the performance of deep learning models. By understanding which input features are most relevant (both positively and negatively), we can gain insights into the inner workings of the model and improve its performance.

So, in the Relevance filter module, we interpret the feature $c_i, i \in \{1,\ldots,M\}$ as positively relevant $c_i = 1$, if its value or weight of relevance $\tilde{y_i} \in [0,1]$ is higher than a given threshold $\tilde{y_i} > \frac{1}{2} + \Delta$, and as negatively relevant $c_i = 0$ if its value is lower than a specific threshold $\tilde{y_i} < \frac{1}{2} - \Delta$.

The third scenario is irrelevant feature $c_i = X$ if $\frac{1}{2} - \Delta \leq \tilde{y_i} \leq \frac{1}{2} + \Delta$:

$$c_i = \begin{cases} 1, & \tilde{y_i} > \frac{1}{2} + \Delta \\ X, & \frac{1}{2} - \Delta \leq \tilde{y_i} \leq \frac{1}{2} + \Delta \\ 0, & \tilde{y_i} < \frac{1}{2} - \Delta \end{cases} \tag{3.5}$$

Thus, we distinguish between the relevance $c_i \in \{0,X,1\}$ and value or degree of relevance $\tilde{y_i} \in [0,1], i \in [1,\ldots,M]$, which we obtain by normalizing eq.(3.3). Note that Δ needs to be positive and $\tilde{y_i} = \frac{1}{2} \Rightarrow c_i = X$.

Next, we derive the truth value (degree of truthfulness) for the relevant features:

$$t(\tilde{y_i} = c_i) = \begin{cases} 1 - \tilde{y_i}, & c_i = 0 \\ \tilde{y_i}, & c_i = 1 \end{cases}, c_i \neq X, i \in \{1,\ldots,M\}. \tag{3.6}$$

It is true that if the feature is relevant, then:

$$t\left((c_i = 1) \vee (c_i = 0)/c_i \neq X\right) \in \left[\frac{1}{2} + \Delta, 1\right], i \in \{1,\ldots,M\}. \tag{3.7}$$

Based on eq.(3.3) and eq.(3.5), it can be inferred that there are a minimum of two relevant features with a unit truth value. At least one of which is positively relevant $(t(c_i = 1) = 1)$, and at least one is negatively relevant $(t(c_i = 0) = 1)$. We exclude the

rare scenario where all truth values are identical, and we associate them with the degree of relevance of the other features. Conversely, multiple features may have a relevance measure near unity compared to a probability distribution. Thus, the normalization method determines the relevance coding. Probabilistic normalization (softmax) results in a one-hot relevance code, while $min - max$ linear normalization produces general binary relevance codewords (in which at least one value is 1 and one value is 0).

So, we use Zadeh's fuzzy logic to determine the truth values of more complex statements[163], [164] for $t(A), t(B) \in$ [0,1]:

$$t(\neg A) = 1 - t(A),$$
$$\bar{t}(A \wedge B) = min(t(A), t(B)),$$
$$t(A \vee B) = max(t(A),t(B))$$

The goal of the fuzzy explanator is to assign to the input pattern $\mathbf{x}$ with the vector of relevance measures $\tilde{\mathbf{y}}$ the relevance codeword $\tilde{\mathbf{c}}$ (explanation) with the highest degree of truthfulness. The truth value of the statement that the evaluated codeword $\tilde{\mathbf{c}}$ is equal to the codeword $\mathbf{c^j}$ is:

$$t(\tilde{\mathbf{c}} = \mathbf{c^j}) = \min_{\substack{i=1,\ldots,M \\ c_i^j \neq X}} t(\tilde{c}_i = c_i^j), j = 1, \ldots, 3^M \tag{3.8}$$

while irrelevant features are not considered. Although the potential set of codewords $\left\{\mathbf{c^j} = \left(c_1^j, \ldots, c_M^j\right) \in \{0,X,1\}^M\right\}, j = 1, \ldots, 3^M$ contains $\mid \mathbb{C}_{full} \mid = 3^M$ codewords, due to linear $min - max$ normalization, a codeword must have at least one 1and one 0, so the set of generated codewords $\mathbb{C}$ will have cardinality of:

$$\mid \mathbb{C} \mid = M(M-1)\binom{M-2}{3} = \frac{M!}{6(M-5)!}, M \geq 2 \tag{3.9}$$

Then, the fuzzy model assigns a code word $\mathbf{c}^{\tilde{m}}$ with the maximum truth value to the recognized pattern (classification done by fuzzy model):

$$\tilde{m} = \underset{c^j \in \mathbb{C}}{\operatorname{argmax}}\, t(\tilde{\mathbf{c}} = \mathbf{c}^j) \tag{3.10}$$

We select the truest codeword from the entire set of admissible codewords $\mathbb{C}$. Rounding the relevance measure of the relevant features can also give the same result. The following theorem discusses this property. It is an extension of Theorem 1 in [147] to the tertiary codespace.

Theorem 1.
Let $\left\{\mathbf{c}^j = (c_1^j, \ldots, c_M^j) \in \{0,X,1\}^M\right\} j = 1, \ldots, 3^M$ be the tertiary codespace,
$\tilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_M) \in [0,1]^M$ a normalized output of the neural network,
$t(\tilde{y}_i = c_i^j)$ truth-value according to eq.(3.6),
and the winning codeword $\tilde{\mathbf{c}}$ be taken according to eq.(3.10).
Then:

- the truth-value $\tilde{\mathbf{c}}$ if the winning codeword is higher than the upper limit $\frac{1}{2} + \Delta$,
- there is only one codeword with a truth-value higher than the upper limit,
- rounding relevant bits according to eq.(3.5) gives the winning codeword.

The proof of the theorem could be found in the paper [147] for the binary case.

3.5.1 Training Phase of Fuzzy Classifier

The main task of training a fuzzy classifier is to find the relevance (code)words $\mathbf{c} = \{(c_1, \ldots, c_i, \ldots, c_M)\} c_i \in \{0,X,1\}, i \in \{1, \ldots, M\}$ that maximize the probability that the fuzzy explainer classifies the codeword into the same class as the non-linear classifier. When aiming to interpret a non-linear classifier, it is reasonable to expect that the proposed fuzzy model should behave as closely as possible to the classifier.

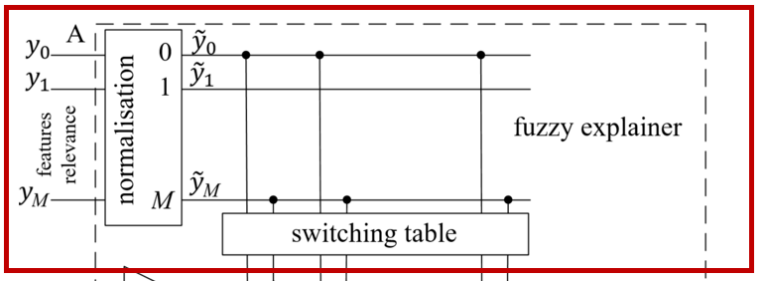


Figure 3.7: Training Phase (creation of codewords) with proposed Fuzzy model

To train the model, two steps are involved in using a training set of data:

1. In the initial stage, we allocate each pattern in the training set to a corresponding codeword based on the formula (3.5). We keep track of created codewords and their respective classes by gathering this information in the statistic table of assigned classes to a codeword 3.1 During this 'training', we increment the count of the

output codeword for a specific class recognized by the fuzzy model. Information is then gathered in the statistics table for classes assigned to a specific codeword (as illustrated in Table 3.1).

Codeword $\mathbf{c}^1$			...	Codeword $\mathbf{c}^{\|\mathbb{C}\|}$		
Class1	...	Class N	...	Class1	...	Class N
n_1^1	...	n_N^1		$n_1^{\|\mathbb{C}\|}$	...	$n_N^{\|\mathbb{C}\|}$

Table 3.1: Patterns mapping into codewords

$\mathbb{C}$ is the set of possible tertiary codewords and $\mid \mathbb{C} \mid$ is its cardinality. Due to the previously mentioned min-max normalization, a codeword must have at least one 1 and one 0. So, $\mid \mathbb{C} \mid < 3^M, M \geq 2$, but still it is very high number (see eq.(3.9)). For instance, if $M = 50$ features, more than 42 million codewords need to be considered. To circumvent the issue of high memory consumption for $\mid \mathbb{C} \mid$ possible codewords, one viable approach is to generate the codeword solely during the training process, when there arises a need to modify the given codeword statistics. The reduction in memory requirements comes at the cost of slower training of the fuzzy logic function. Hence, it is imperative to give due consideration to both memory requirements and training time.

2. During the second step, the appropriate codewords (see dominance condition 3.11) are chosen and assigned to their respective classes. This is done the following way. The value of n_k^j associated with the pattern of class $k \in \{1,\ldots,N\}$ assigned to codeword $\mathbf{c}^j \in \{1,\ldots,3^M\}$ denotes the ratio of accurate to inaccurate classifications made by the fuzzy model.
   In other words, the value of n_k^j indicates the number of times that the classifier correctly identified the given class $n_k^{j^+}$ in comparison to the number of incorrect classifications $n_k^{j^-}$, i.e., $n_k^j = n_k^{j^+} - n_k^{j^-}$ A codeword $\mathbf{c}^j \in \{1,\ldots,3^M\}$ is deemed appropriate for a particular class $k \in \{1,\ldots,N\}$ if the following condition is satisfied:

$$n_k^j > \alpha \sum_{i \neq k} n_i^j \tag{3.11}$$

   where $\alpha \geq 1$ gives the chosen security measure (in this work $\alpha = 1$). Should a

codeword be inappropriate for any class, it will not be used in the fuzzy classifier. Denoting the set of all codewords by $\mathbb{C}$, and the subset of codewords used by the fuzzy explainer after training as $\mathbb{C}^T \subset \mathbb{C}$, the output of the training process is a set of suitable codewords $\mathbb{C}^T$, each of which is assigned to an appropriate class $m = \psi(\mathbf{c}), m \in \{1, \ldots, N\}, \mathbf{c} \in \mathbb{C}^T$.

3.5.2 Testing Phase of Fuzzy Classifier

In order to get a model prediction of the target pattern $\mathbf{x}$ belonging to class $m \in \{1, \ldots, N\}$, it is fed as input to the black-box recognizer (DNN classifier), whereupon the feature extractor identifies the salient $\mathbf{f}$ for the given pattern.

Subsequently, the classification part utilizes the extracted features $\mathbf{f}$ to estimate the class of the pattern. If the estimation is of class $z \in \{1, \ldots, N\}$, these classes may be the same $z = m$, or it may be the wrong class $z \neq m$. From the decision of the black box classifier, we derive the relevance of the features $\mathbf{y}$ and feed them to the input of the explainable classifier. Thus, during the testing phase, we obtain the corresponding codewords from the inference of test samples. According to the created Codeword Table (3.1) we find the nearest codeword (discrete vector) that was not discarded during the training phase.

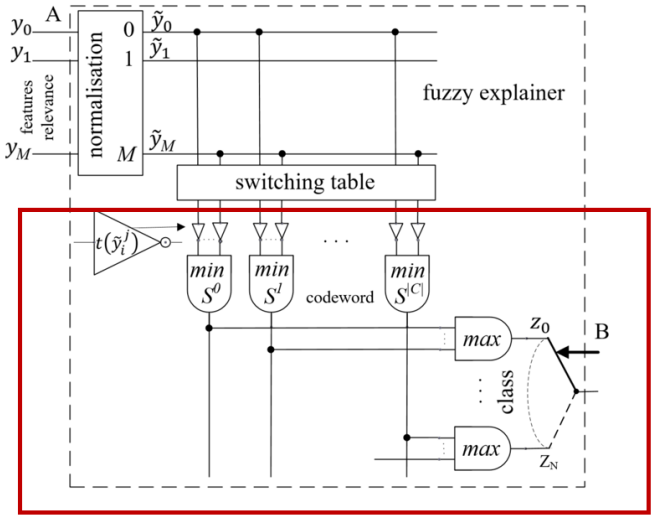


Figure 3.8: Testing Phase with proposed Fuzzy model

3.5.3 Evaluation Metrics

Unlike the training phase, during testing, the selection of codewords is limited to the subset $\mathbb{C}^T \subset \mathbb{C}_{full}$ generated during training as opposed to the entire set of codewords $\mathbb{C}_{full}$. Since the main goal is to explain the classification of the target pattern $\tilde{m}$ determined by the black box classifier, only the codewords assigned to the same class $\mathbb{C}^T_{\tilde{m}}$ as predicted by the black-box classifier are selected, irrespective of whether the prediction itself was accurate or not.

$$explanation \;\; \tilde{\mathbf{c}} = \underset{j:\mathbf{c}^j \in C^T_{\tilde{\tilde{m}}}}{\operatorname{argmax}} \;\; t(\tilde{\mathbf{c}} = \mathbf{c}^{\mathbf{j}}) \tag{3.12}$$

where $t(\tilde{\mathbf{c}} = \mathbf{c}^{\mathbf{j}})$ is defined in 3.8.

The accuracy of prediction on the black-box model is computed as:

$$acc_{BBox} \;\; = \frac{\sum_{i=1}^{n} \delta(\tilde{\tilde{m}}_i = m_i)}{n} \tag{3.13}$$

where $\tilde{\tilde{m}}_i$ is black-box prediction, m_i is true label, and

$$\delta(\tilde{\tilde{m}}_i = m_i) = \begin{cases} 1, & \tilde{\tilde{m}}_i = m_i \\ 0, & \tilde{\tilde{m}}_i \neq m_i \end{cases}$$

Then, the accuracy of an explainable classifier can be evaluated in a manner, similar to the common accuracy metric for a black box classifier 3.13.

$$acc_{fuzzy} \;\; \rho = \frac{\sum_{i=1}^{n} \delta(\tilde{m}_i = m_i)}{n} \tag{3.14}$$

where $\tilde{m}_i$ is the class of test sample $\mathbf{x}_i$ estimated by the fuzzy classifier with respect to true labels m_i.

Eq.(3.14) basically represents the regular accuracy metric - the proportion of correct predictions made by the model over the total number of predictions made.

The fidelity metric is just a slight alteration from eq.(3.14) and (3.13).

$$fidelity \;\; \rho_r = \frac{\sum_{i=1}^{n} \delta(\tilde{m}_i = \tilde{\tilde{m}}_i)}{n} \tag{3.15}$$

It is still the proportion of correct predictions made by the model over the total number of predictions but with respect to a black-box prediction $\tilde{m}_i$. Because the explainable classifier explains the result obtained by the black box classifier, this metric - fidelity of

explanation eq.(3.15) is the most important.

Another important metric to compute during the testing phase is the stability of explanations. It refers to the degree to which an explanation remains consistent and valid over time **within the class**, despite changes in the surrounding circumstances or additional information. In order to compute this metric we employ the Hamming distance [165].

$$stability \quad H(C_t,C_m) = \sum_{i=1}^{n} \delta(b_i^t \neq b_i^m) \tag{3.16}$$

where b_i^t are respective bits in codewords C_t assigned to the pattern during the testing phase for a specific class.

3.5.4 The contribution of fuzzy logic to the explainability within the classification domain

- The post-hoc classifier takes the form of a fuzzy logical expression, with a set of rules (IF, THEN) forming the basis of logic (modus ponens: $(P \rightarrow Q) \wedge P \rightarrow Q$) applied to indefinite data that has been extended to fuzzy logic. In fuzzy logic(measures the degree of truthfulness of a statement on a unit interval.).

- Within the classification task, we obtain the degree of truthfulness for a specific classification. Additionally, it is possible to eliminate irrelevant statements and determine the degree of truth for crucial statements.

- Since the proposed method is in the role of a post-hoc model, we can determine how well the explanation approximates the prediction of the black box model(fidelity).

- As a degree of truthfulness of features, we can use the values of the features themselves or their respective relevance. Then we can compare the effectiveness of various relevance methods by comparing the fidelity, or the accuracy of the classification for the explainable classifier.

- The resulting Fuzzy Logic Function (FLF) is expressed in the format of a Disjunctive Normal Form (DNF). Such a function can be converted into a tertiary decision tree. On top of that, the number of logical operations can be reduced using the same approach applied in minimizing boolean logic functions.

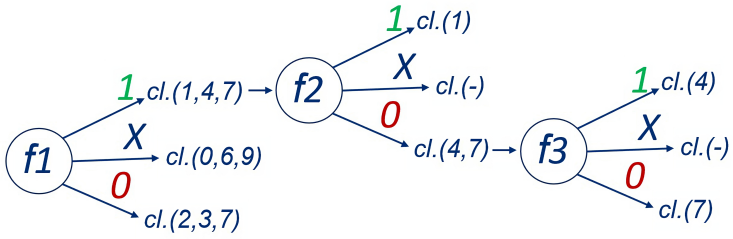


Figure 3.9: Example of FLF in tertiary decision tree

- We can search for the custom optimal transformation of the features based on their degree of truthfulness $t = \mu(f)$. This way we get rid of the replacement of the raw feature by their computed relevances - this is an objective for future research.

3.6 Interface for Interpretation of Features

The inherent issue of explainability, which persists throughout the entire research presented in this work, is apparent. The non-linear classifier serves as a black-box model, where an unknown pattern is introduced at the input, and a classification for a given pattern is produced at the output. Nevertheless, the challenge lies in the fact that the computer model is not accountable for the accuracy of the recognition; rather, it is the responsibility of the end-user. The role of the recognizer is that of aid, and it is imperative that the end-user not only be informed of the system's recommendation but also be provided with a comprehensible explanation as to why. To achieve this, the first step is to clearly distinguish between the feature extraction and classification subsystems, as proposed in Section 3.1. If we aim to include non-natural features, the user must learn to comprehend the extracted features for optimal classification, which they may not have previously encountered. The crux of the proposed explainable method is the development of a User-Interface that not only presents the features and classification outcome for a given input image but also allows the user to modify the features (model them) and observe the impact of these changes on the pattern presented and the classification decision.

The User Interface (UI) proposed in this study is just one of several possible implementations. Its primary objective is to ensure that the end user can easily comprehend the information provided. However, selecting the most suitable implementation requires careful consideration of numerous factors, including the problem's domain, the specific task, the system architecture, data complexity, and data type, among others.

So, to provide a comprehensive explanation for the specific image sample, it is crucial to first describe the Explainable User-Interface illustrated in Figure 3.5. Accordingly, the explanation of user inference is divided into multiple components:

1. The graph located in the upper left corner depicts a 2D feature space that highlights the relationship between two features chosen by the user. As the values of these features are altered, the user is presented with graphical feedback on the updated location of the features within the feature space.

2. The area next to the feature space is designated for image manipulation (explanation objects). The user can choose an image/file to analyze and upload it from

their local storage to the server. Upon upload, all the features are extracted, and the original image is presented. If the user modifies any features, the reconstructed image will be displayed. For the purpose of image reconstruction, we trained the decoder model as a subsequent part to the already used classifier. So, extracted features from the classifier are then used not only as an input to the fuzzy model but also as a latent representation needed for the reconstruction of the image. In this specific architecture, the classifier and decoder model represents a fully-convolutional autoencoder. Also, the decoder part is reacting to changes in features done by the end-user and modifies the reconstructed pattern accordingly.

3. The subsequent section presents a summary of the features extracted from the uploaded image data. Every feature is assigned an identifier (header) and a value within the normalized range of $[0-1]$. In this section, all features remain inactive and are immune to updates in value. However, the user can select any feature to experiment with and modify its value accordingly. The explainable classifier also offers the user the option to select feature relevance. The proposed post-hoc fuzzy model offers three states to convey this information: 1 - denotes positive relevance, X - signifies irrelevance, and 0 - indicates negative relevance. Hence, the user can concentrate on elements that make positive or negative contributions and disregard irrelevant elements.

4. To the left of the feature overview is the section where the user can modify the two selected features based on their relevance and interest. By limiting the number of chosen features, the user can experiment by manipulating their values and gaining a better understanding of their relevance. Upon altering the values, the pattern reconstruction subsystem generates the new corresponding image, and the black-box classifier performs a new classification prediction for a generated image. Moreover, the updated feature relevance is exhibited. This enables the user to understand the significance of the features by observing how they affect the image.

5. The final section located at the bottom of the interface portrays the outcome of the black box model's prediction (classification). The top row indicates the available classes of the image data used, while the second row displays the assigned

probabilities for each class by the non-linear model (SoftMax normalization).

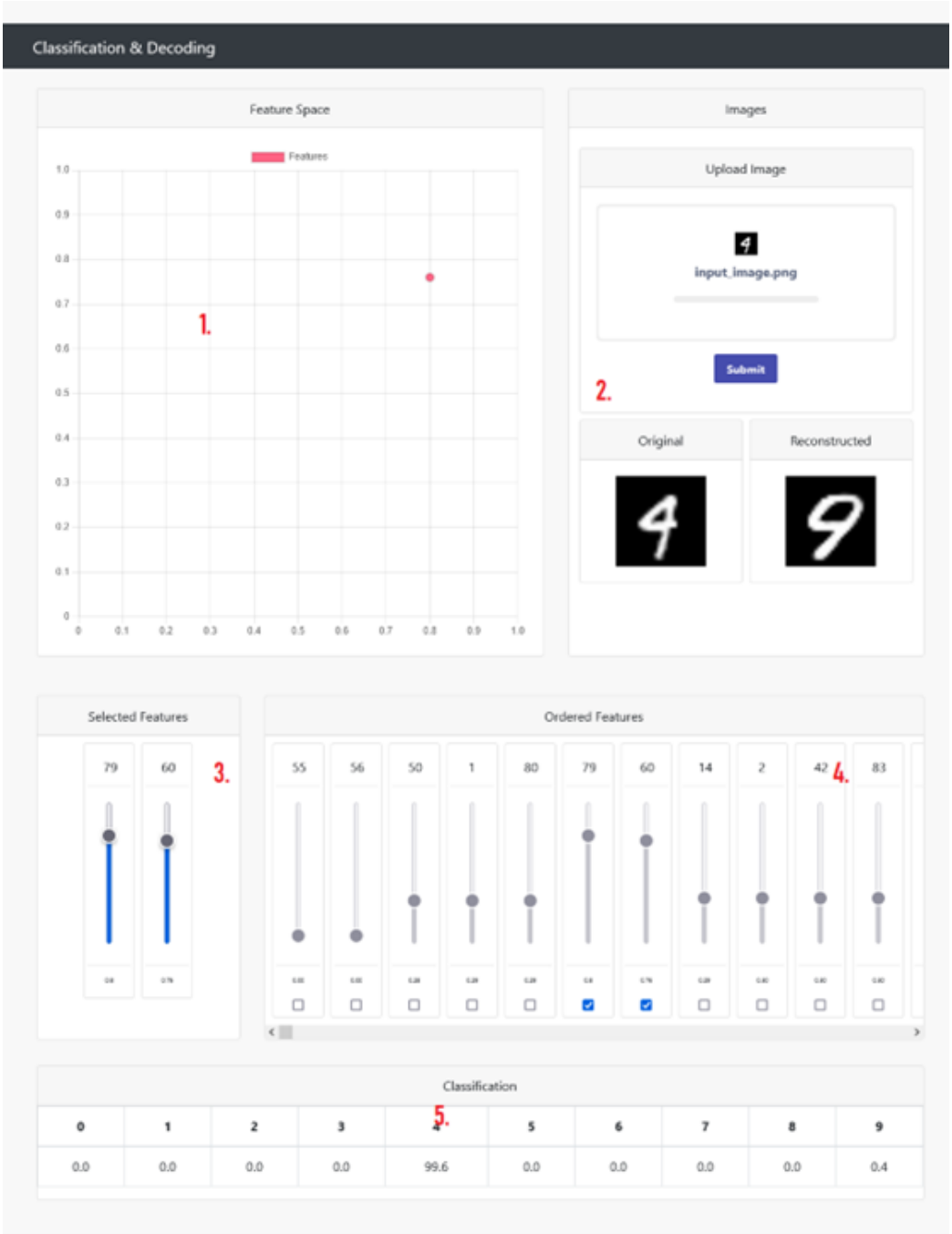


Figure 3.10: The illustration of Explanation User-Interface

Figure 3.10 shows a concept of the interface between the user and the feature interpreter. In contrast to other feature interpretation methods, the feature UI provides a unique advantage in teaching end-users to understand the individual symptoms used by the classifier. The learning process centers around user experimentation with the pattern recognition process. The interface presents the input pattern, features, and

predicted class to the user, but also allows active modification of features. Users can observe how the classification result changes and what input image leads to such changes. The system provides the ability to determine the most relevant features for a change in classification and compare differences in the input images leading to different classifications. This approach will enable experienced users to provide a qualified recommendation for improving specific non-linear classifiers.

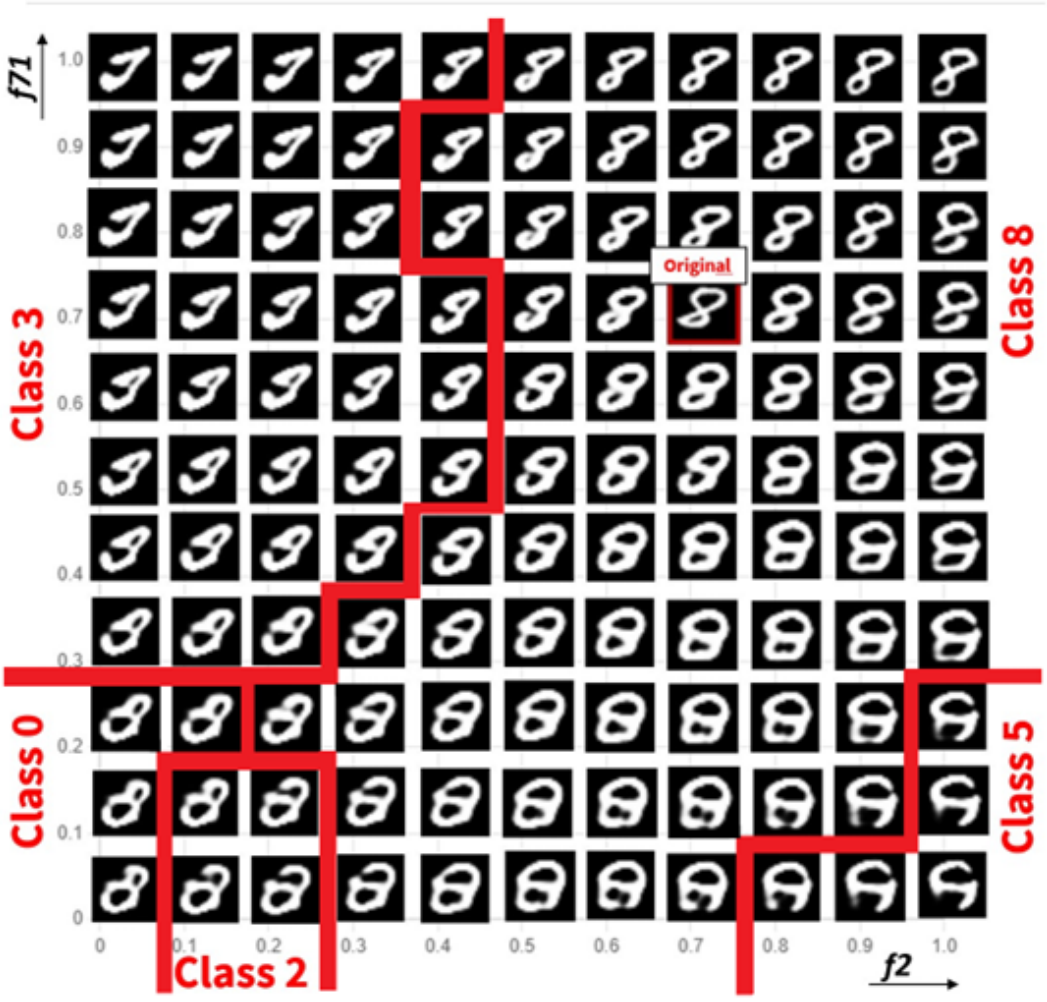


Figure 3.11: Feature interpretation example: an impact of the two positive relevant features on the shape of the reconstructed image - built with explainable UI

In Figure 3.11 we choose features No.71 and No.2 as these were provided by the Fuzzy model as the two most positive relevant features in the explanation $\tilde{c}$.

Chapter 4

Developed Explainable Approach: Experimental Evaluations

To demonstrate the proposed concept, we begin with less complex datasets such as MNIST [158], and Fashion-MNIST[166], but also include very complex and large the benchmark dataset ImageNet[167]. Although these datasets have reduced complexity (in the case of MNIST and FMNIST - $HelloWorld$ of computer vision), they serve as reliable references in our research. It is important to note that using less complex datasets is still informative and can demonstrate the effectiveness of the fuzzy classifier and our proposed concept of explainability. Also, it is worth mentioning that experiments with more complex data are also of great interest, but they require significantly more computational resources and a higher level of customizability, which must be taken into account. In addition to the aforementioned reasons, using datasets of lesser complexity like MNIST also offers an advantage in terms of achieving high accuracy in classifying images, with most state-of-the-art neural network models achieving an accuracy of over 99% on such datasets. This is in contrast to the more complex ImageNet scenario, where high accuracy may be more challenging to attain.

It is important to note that while complex heavy data may not have a direct impact on the proposed post-hoc model, it is still crucial to achieving high accuracy on the black box classifier for two reasons:

- First, although accurate (true) predictions are not necessary for explainability cause, they are necessary to ensure that the explanation itself is not worthless and does not reveal incorrect classifications.
- Second, highly accurate models can extract better or more essential features, which are highly effective as input for the post-hoc explanation.

The effectiveness of the post-hoc interpretation is dependent on the complexity of the task at hand, although it can be applied universally. Using datasets that are less complex, such as MNIST, can help conserve computational resources while also providing the most accurate classification and thus the best explanation as a conceptual showcase demonstration.

4.1 Model Evaluation with MNIST and Fashion-MNIST

4.1.1 General Description

The original LeNet-5 model architecture [159] was utilized for MNIST and Fashion MNIST. the antiquated architecture of the Lenet5 has no detrimental effect in terms of explainability aim. However, we had to make several modifications to achieve state-of-the-art (SOTA) accuracy on these datasets. The deeper and denser network enabled the model to learn more complex features. As was already mentioned, extracting the most meaningful features from the data is crucial for the proposed explainability method, which uses these features as inputs to the post-hoc fuzzy classifier in their raw form or modified with interpretation methods for deep learning (e.g., gradients, relevance). The convolutional part of the architecture was doubled with the same hyperparameters, and the number of filters in the convolutional layers was significantly increased. Consequently, the number of dense layers (classification head) was also increased to accommodate the more extensive input resulting from the increase of the previous layers. We performed

various optimizations for both MNIST and FMNIST datasets. These optimizations included regularisation techniques such as batch normalization[168], dropout[169], L2 regularisation [170] and variable learning rate. These techniques helped us achieve an accuracy of over 99% on MNIST and 92% on FMNIST within just 30 epochs of training. It is important to note that the features used as input data for the post-hoc classifier were extracted immediately after the final convolutional layer. When using interpretation methods, the relevance is computed in reverse to the same layer and then extracted as a feature vector. This approach is appropriate for focusing on obtaining the most compelling features. At the end of the convolutional section of the network, we assume that all essential spatial information is preserved and compressed.

4.1.2 Experimental Results

To attain the best precise results on both datasets, we performed ten training iterations on the employed model using the abovementioned datasets. We obtained accuracy metrics from each training and computed the mean and standard deviation to determine the most precise performance of the model for a given dataset.

Relevance Method	Fidelity ρ_r (%)	Accuracy Fuzzy ρ (%)	Accuracy B_{Box}(%)
Vanilla Grad	99.47 [0.21]	99.10 [0.22]	99.61 [0.05]
Raw normalized features	99.20 [0.17]	99.06 [0.17]	99.61 [0.05]
LRP relevance	98.35 [3.21]	97.99 [3.20]	99.61 [0.05]
Guided Backprop	99.79 [0.21]	99.42 [0.22]	99.61 [0.05]
DeconvNet	**100 [0]**	**99.61 [0.05]**	**99.61 [0.05]**

Table 4.1: MNIST results on Fuzzy-logical Model with respect to enhanced LeNet5 classifier

In order to obtain a more comprehensive understanding of the overall performance of the post-hoc model, we have included two additional gradient-based methods in our comparison: Guided Back-propagation [73] and DeconvNet [72]. As can be seen from Tables 4.1 and 4.2, the post-hoc explainer has the potential to be just as accurate as the black box classifier itself. This is yet another compelling reason to strive for the highest possible accuracy in the black box model. Furthermore, our results confirm the direct

Relevance Method	Fidelity ρ_r (%)	Accuracy Fuzzy ρ (%)	Accuracy B_{Box}(%)
Vanilla Grad	98.33 [0.75]	91.28 [0.61]	92.46 [0.25]
Raw normalized features	91.83 [0.51]	88.62 [0.42]	92.46 [0.25]
LRP relevance	93.38 [2.10]	86.96 [2.05]	92.46 [0.25]
Guided Backprop	99.59 [0.66]	92.25 [0.54]	92.46 [0.25]
DeconvNet	**100 [0]**	**92.46 [0.25]**	**92.46 [0.25]**

Table 4.2: FMNIST results on Fuzzy-logical Model with respect to enhanced LeNet5 classifier

correlation between the accuracy of the post-hoc explanation and the relevance of the input data and extracted features. This implies that utilizing more relevant features results in an improved ability of the post-hoc model to approximate the non-linear classifier. Neural networks are trained to converge by calculating gradients with respect to the loss function and network parameters. The gradient-based method utilizes a similar approach to extract relevant feature information. It is important to note that the features perform quite well in this method, as demonstrated in the tables. In particular, the DeconvNet technique filters out negative gradients, which has an even more desirable effect in this specific instance. Typically, it is not the positive or negative values themselves that are important, but rather the magnitude of the gradient. A pixel is likely unimportant for making a prediction if its associated gradient has a small magnitude. On the other hand, if the gradient magnitudes are significant, it indicates that the pixel is essential. Also, it is possible to interpret negative values as a form of inverse feature detection during the convolutional operation, while positive values correspond to the positive detection of a specific feature.

Thus, the results presented in Tables 4.1 and 4.2 suggest that the proposed post-hoc explanation method can be applied universally to computer vision tasks that are easy to solve by neural networks with exceptionally high prediction accuracy. Additionally, the choice of feature extraction method has a relatively minor impact on performance, resulting in only marginal reductions in accuracy across various methods. This approach is also valid from an explainability perspective. An effective neural network demonstrates accurate predictions, which are directly associated with the accuracy of the post-hoc

method and, ultimately, lead to a clear explanation.

4.1.3 Feature reduction for better explainability

The post-hoc model takes a vector with a length of 84 extracted features as input in the case of LeNet-5 architecture. However, a significant number of these features in the output explanation are irrelevant, and they do not contribute to the final prediction. As we scale the method for more complex datasets (ImageNet scenario), the number of features in the vector increases significantly depending on the architecture and choice of layer for feature extraction. In such scenarios, the resulting explanations for the prediction could become noisy and challenging to interpret. Hence, it is imperative that we perform feature reduction in order to effectively handle complex data. One way to achieve this is by adjusting the lower and upper thresholds of the Relevance Filter - part of the post-hoc model design, which can help to expand the irrelevant range (see equation 3.5).

Table 4.3 and 4.9 present the outcomes of the feature reduction process, wherein we have widened the irrelevant zone and imposed more stringent conditions for both positive and negative features. During this reduction process the average count of positive $N_{positive}$ and negative $N_{negative}$ features is diminishing. Additionally, it is crucial to note the inevitable trade-off between the performance of the post-hoc model and the feature reduction process if the initial thresholds of the filter are set too strictly. Imposing such excessively strict thresholds for positive and negative features may erroneously designate certain critical features as irrelevant, thereby impacting the model's performance and the interpretation of its predictions. It is worth highlighting that the lower and upper limits of the relevance filter determine the overall bandwidth of the irrelevance range. It is symmetrically centered around the center of the unit interval, starting at one-third:

$$LowerLimit = \frac{1}{3} - \Delta, \quad UpperLimit = \frac{2}{3} + \Delta \tag{4.1}$$

LeNet5/MNIST	Δ	$N_{POSITIVE}$	$N_{NEGATIVE}$	Fidelity $\rho_r(\%)$
Vanilla Grad	0	7.88	9.5	99.24
	0.1	3.62	2.73	99.33
	0.2	1.39	1.27	99.5
	0.3	**1.01**	**1.01**	**99.63**
Raw normalized features	0	**15.17**	**22.55**	**99.06**
	0.1	5.8	8.09	96.08
	0.2	1.53	1.64	85.54
	0.3	1.02	1.02	89.8
LRP	0	**2.18**	**40.4**	**99.8**
	0.1	1.39	4.52	99.12
	0.2	1.04	1.15	98.63
	0.3	1	1	99.06
Guided Backprop	0	4.7	66.27	99.96
	0.1	2.83	61.94	99.96
	0.2	**1.84**	**57.84**	**99.99**
	0.3	1.11	53.91	99.97
DeconvNet	0	6	64.37	100
	0.1	4.34	58.49	100
	0.2	2.39	52.33	100
	0.3	**1.39**	**45.11**	**100**

Table 4.3: Responsiveness of the Feature Reduction Method used on LeNet-5/MNIST

The results were also published in the paper [171].

4.2 Model Evaluation with ImageNet

4.2.1 General Description

In the following section, we included the ImageNet dataset [167] to evaluate the post-hoc model's scalability on more complex data. We selected four different neural network models for this very purpose: VGG16 [150], ResNet-50 [172], DenseNet121 [173], and InceptionV3 [174]. This time, we did not use any further enhancements or re-training multiple versions of these architectures. The networks had already been trained with the highest possible accuracy. Table 4.4 illustrates the overall performance of these models.

Architectures	Accuracy (%)
VGG 16	64.44
ResNet-50	67.46
DenseNet121	71.20
Inception V3	**75.84**

Table 4.4: Employed Architectures and their respective Accuracy score on ImageNet

4.2.2 Experimental Results

The results presented in Table 4.5 demonstrate a clear decline in the performance of the proposed post-hoc model on the ImageNet dataset. However, the overall accuracy of black box architectures is significantly lower than the 90% threshold (see Table4.4). In the case of the DenseNet121 and VGG16 models, we were able to achieve a comparable level of accuracy with the post-hoc explainable model. However, the available methods for feature extractions now indicate more significant performance gaps.

	ResNet50			**VGG 16**		
Relevance Method	**Fidelity p_r**	**Accuracy Fuzzy p**	**Accuracy B_Box**	**Fidelity p_r**	**Accuracy Fuzzy p**	**Accuracy B_Box**
Vanilla Grad	5.09	4.58	67.46	47.89	26.82	64.44
Raw normalised features	**34.09**	**31.04**	**67.46**	41.01	36.84	64.44
LRP relevance	-	-	67.46	21.74	17.8	64.44
Guided Backprop	5.09	4.58	67.46	15.66	12.66	64.44
DeconvNet	5.09	4.58	67.46	**100.0**	**64.44**	**64.44**

	InceptionV3			**DenseNet121**		
Relevance Method	**Fidelity p_r**	**Accuracy Fuzzy p**	**Accuracy B_Box**	**Fidelity p_r**	**Accuracy Fuzzy p**	**Accuracy B_Box**
Vanilla Grad	11.35	10.42	75.84	100	71.2	71.2
Raw normalised features	**65.87**	**60.49**	**75.84**	40.24	36.99	71.2
LRP relevance	-	-	75.84	-	-	71.2
Guided Backprop	11.35	10.42	75.84	**100**	**71.2**	**71.2**
DeconvNet	11.35	10.42	75.84	**100**	**71.2**	**71.2**

Table 4.5: ImageNet results on proposed Fuzzy-logical Model with respect to specific classification Architectures

Once again, the results of the experiments reinforce the high dependence on black box architecture in use. The more efficient the architecture is, the more relevant features are extracted as input for the post-hoc method. If the black box model fails to extract the necessary features, computing the interpretation methods (rather than just raw extraction) might improve the overall effectiveness of feature extraction. From Table 4.5 is possible to infer the overall results on ImageNet.

Architectures	Total Score (%)
ResNet-50	23.0
VGG 16	64.44
Inception V3	49.9
DenseNet121	**71.2**

Table 4.6: Overview of best performance in combination Black-box/Fuzzy model

Table 4.6 evaluates the best performance from all employed architectures in terms of accurate explanations. We consider the explanation to be correct when both Black-box and Fuzzy models provide an accurate prediction. If we assume independence, it can be defined as:

$$Total\ \ Score = Accuracy\ \ B_{Box} \times Fidelity\ \ \rho_r \tag{4.2}$$

Another, better solution to this issue is to investigate the development of custom feature extraction methods (Custom Transformation) that are either universal or tailored to the specific architecture. These methods could help maximize the performance of the post-hoc model by enhancing the extraction effectiveness if the black box model is insufficient in extracting features.

The LRP method [78] was only calculated for the VGG16 model as is shown in Table 4.5. The VGG16 architecture is designed more traditionally, with a proper classification head that allows us to compute the LRP signal backward through multiple fully-connected layers and to the point of relevance signal extraction (the final convolution layer). However, the remaining models have convolutional layers throughout their whole architecture, and just one linear output layer with SoftMax normalization (prediction layer) immediately follows the final convolution layer. In this case, the LRP is not meaningful due to its design, as the calculated signal is too shallow.

The results were also published in the paper [171] (see List of Publications at the end of work).

We also utilized the stability metric (see 3.16) to assess the similarity between explanations of instances within each class for a particular architecture. It is crucial to have stable explanations since a dramatic change in explanation with a small variation in input can result in mistrust or ambiguity regarding the model's behavior. A stable explanation assures that the model is making consistent decisions based on the input provided.

Architecture / Codeword Length	Relevance Method	Bit Match	Stability (%)
VGG16/4096 bits	Raw	2971.28	72.54
	Vanilla Grad	3789.96	92.53
	Guided Backprop	3937.23	96.12
	DeconvNet	**4096**	**100**
ResNet50/2048 bits	Raw	2004.03	72.54
	Vanilla Grad	**2046.44**	**99.92**
	Guided Backprop	**2046.44**	**99.92**
	DeconvNet	**2046.44**	**99.92**

Table 4.7: The average stability of explanations within classes for architecture VGG16 and ResNet50

Architecture / Codeword Length	Relevance Method	Bit Match	Stability (%)
DenseNet121/1024 bits	Raw	984.12	96.12
	Vanilla Grad	**1024**	**100**
	Guided Backprop	1022.51	99.85
	DeconvNet	**1024**	**100**
InceptionV3/2048 bits	Raw	1937.14	94.58
	Vanilla Grad	**2046.67**	**99.93**
	Guided Backprop	**2046.67**	**99.93**
	DeconvNet	**2046.67**	**99.93**

Table 4.8: The average stability of explanations within classes for architecture Inception V3 and DenseNet121

Based on the Stability evaluation presented in Tables 4.7 and 4.8, it is evident that

the explanations provided by the Fuzzy Logical model exhibit stability. This parameter holds significant importance as it enables end-users to comprehend the rationale behind a particular decision, thereby enhancing overall comprehensibility. Moreover, these explanations can assist in facilitating a smooth transition to feature interpretability by offering insights into the input features that are pivotal in driving the model's decisions.

4.2.3 Feature reduction when dealing with ImageNet

When we deal with more complex data like ImageNet the feature reduction method (chapter 4.1.3) yields even greater benefits. In this particular experiment, we have opted to employ the DenseNet121 architecture. Table 4.9 displays the average count of positive and negative features alongside the overall performance of the model under various threshold levels. Based on the results, we can conclude that the feature reduction (total amount of $N_{POSITIVE}$ and $N_{NEGATIVE}$) was successful, with an explanation fidelity rate of over 99% maintained.

We have again the lower and upper limits symmetrically centered around the center of the unit interval, starting at one-third:

$$LowerLimit = \frac{1}{3} - \Delta, \quad UpperLimit = \frac{2}{3} + \Delta \tag{4.3}$$

DenseNet121/ImageNet	Δ	0	0.1	0.2	0.3
VanillaGrad/ Guided-Backprop/DeconvNet	$N_{POSITIVE}$	10.59	5.19	2.5	2.04
	$N_{NEGATIVE}$	774.12	275.15	44.28	21.75
	Fidelity $\rho_r(\%)$	100	100	99.78	99.12
Raw normalized features	$N_{POSITIVE}$	4.12	2.44	1.5	1.05
	$N_{NEGATIVE}$	989.03	949.35	856.96	627.17
	Fidelity $\rho_r(\%)$	40.24	40.24	38.61	27.57

Table 4.9: Responsiveness of the Feature Reduction Method used on DenseNet-121/ImageNet

Also, please not that Table 4.9 indicates that all gradient methods have identical values. This is once again attributable to the specific structure of the DenseNet121 architecture, which was also observed in the LRP scenario depicted in Table 4.5. The convolutional

section of the network spans the entire architecture, causing the calculated gradients to be derived immediately prior to the output layer. As a result of this shallow gradient calculation, all gradient methods returned the same values.

4.2.4 Comparison with conventional method - Decision Tree Algorithm

To facilitate a comparison between the suggested fuzzy classifier and conventional approaches, we utilized a decision tree for analysis. It is a highly adaptable machine learning algorithm that can perform classification, regression, and multi-output tasks proficiently. Moreover, it is a highly intuitive concept that generates easily explainable decisions, often known as "white box" models. Given their robustness and capability to fit complex datasets, decision trees prove to be an ideal choice for conducting a valuable comparison. Furthermore, decision trees demand minimal data preparation, such as feature scaling or centering, to operate at an optimal level. The decision trees employed in this study were generated via the CART[175] algorithm, which creates binary trees, where split nodes invariably have precisely two children[176][177].

How to choose the optimal depth of the Decision Tree

In a scenario like this determining the optimal depth of a decision tree represents a complex process as it depends on several factors, including the distribution of the features, the complexity of the dataset, and the desired evaluation metrics. As a general rule, it is advisable to set the maximum depth of a decision tree at a level that achieves a balance between the bias-variance tradeoff. A shallow tree with a low maximum depth could result in high bias, which oversimplifies the underlying patterns and relationships within the data. Conversely, a deep tree with a high maximum depth may exhibit high variance, which overfits the data by capturing noise and variability that are not necessarily representative of the underlying trends[178].

There are some possibilities and general guidelines that can help determine an appropriate maximum depth for a decision tree.

- Our intention is to utilize the decision tree as a post-hoc approach to the pre-existing black box classifier, aiming to compare its performance with the proposed Fuzzy model. Given that both models are logical, we can set the maximum depth

of the decision tree equal to the maximum logical decision made by the Fuzzy classifier to provide an explanation. One way how to achieve this is to derive the estimate by calculating the average $N_{POSITIVE}$ and $N_{NEGATIVE}$ features from the resulting explanation vectors of the Fuzzy model.
There exist alternative techniques to calculate the maximum number of logical decisions made by the Fuzzy model. However, further investigation is required in this domain and therefore it is left for further research.

- Another approach for identifying the appropriate maximum depth for a decision tree is to leverage cross-validation to assess its performance across varying depths. This method involves splitting the data into training and validation data folds, training the model on the training set, and then evaluating its performance on the validation set. This process is repeated for various maximum depth values, and the optimal depth is selected based on the desired performance metric, such as accuracy or precision [178][179].

- Last but not least its possible to apply regularization techniques, such as pruning, to restrain the complexity of the decision tree and avoid overfitting. Pruning consists of eliminating branches from the tree that do not enhance the model's performance on the validation set [180].

In summary, the optimal depth of a decision tree depends on various factors and can be determined through experimentation and fine-tuning.

Training and Inference

After the experimental analysis and fine-tuning of decision tree depth, we decide to proceed with two models. The first tree was trained with a depth node of 786 while the second implementation is 512 nodes deep. The first parameter was chosen based on the maximum logical decision made by the Fuzzy classifier, while the second attribute is the result of cross-validation tuning. Both of the decision trees were trained as post-hoc models for DenseNet121 architecture. It's important to note that the optimal maximum depth may vary depending on the specific dataset and the actual architecture of the black box since we dealing with a post-hoc concept. Thus, it's always a good idea to perform experiments to determine the best hyperparameters for specific use cases.

The results presented in Table 4.10 indicate a lower performance compared to

Decision Tree Depth	Relevance Method	Fidelity ρ_r	Accuracy DTree ρ	Accuracy B_{Box}
786	Deconvnet	74.54	53.07	71.2
	Raw features	63.42	45.15	71.2
512	Deconvnet	10.92	9.59	71.2
	Raw features	51.13	39.96	71.2

Table 4.10: Performance of post-hoc Decision Tree as a substitute for Fuzzy logical model used with DenseNet-121/ImageNet

the scenario utilizing a post-hoc fuzzy classifier (see Tables 4.5 and 4.6), implying the superiority of the Fuzzy model over the deployed Decision trees. However, future research could explore the possibility of employing decision trees as an absolute substitute for the Fuzzy model. Similar to the research on Custom Transformation, which replaces the conventional relevance/saliency method, replacing the Fuzzy model with a Decision tree could be a viable alternative. These findings suggest that by conducting a thorough investigation on fine-tuning decision trees in specific application domains, better results could be achieved with shallower models.

4.3 Conclusion

Experimental results show that the explainable fuzzy classifier can match its classification almost identically with the black box classifiers. Performance parity is only achieved when the classifiers strictly separate the feature extraction process from classification, and when they can successfully extract meaningful and effective features from input data (architecture dependent).

The primary step is to use the relevance of the features as input values for the fuzzy classifier. The best results were obtained by methods that determine feature relevance using gradient-based methods with back-propagation, and the overall winner among the tested methods was the DeconvNet. However, it is crucial to think through the entire process, including available data and architecture, when considering the implementation of this post-hoc explanation. Although the proposed general concept remains immutable, the actual implementation must be customized for the specific task at hand.

Also, it's worth mentioning that is significantly easier for the user to explain the classification process with fewer relevant features (due to the fact that modern models are vast in terms of parameters). Therefore, it is also necessary to experiment with the irrelevance bandwidth on the unit interval of the truth values of the features. The test results show that aggressive widening of the irrelevance band negatively affects the explanation if we use the normalized features as the truth values. When is used feature relevance, the number of relevant features can be reduced without degrading the actual relative accuracy between post-hoc and black-box models. Notably, during the process of reducing the relevant features, a greater number of negatively relevant features (i.e., those that should not be present) persist compared to positively relevant ones (i.e., those that should be present).

Chapter 5

Challenges Posed by Outliers: An Investigation

The definition of outliers provided in Chapter 2.4 encapsulates the primary approach to anomaly detection in the current work. Also, in Chapter 2.4, the outlined methods are characterized by their reliance on comparing the similarity of the tested pattern with other patterns present in the original dataset. The quotation in question asserts that the discrepancy may be significant enough to warrant the conclusion that the patterns were not produced by the same mechanism. Therefore, it is not imperative for us to have knowledge of the precise creation mechanism of the given pattern, but rather it suffices to understand the mechanism responsible for generating its image.

According to this, we expect the normal samples in the same class to be created by the same mechanism. Unfortunately, this mechanism is not known for the training/original data. However, we do have a mechanism how to generate the samples which can not be distinguished from the training data during the turning test performed by the trained neural network - Discriminator. If the generated data are similar to the training data then we expect the same generation mechanism for both, because we successfully capture the original data distribution.

While compression algorithms, whether linear or non-linear, facilitate the extraction of crucial information from images and enable the reconstruction of image patterns based on such information, unique pattern features are still necessary for object construction. In Chapter 2.4.2, GAN networks (but also DGM in general) are presented as a solution to comprehend the fundamental workings (creation mechanism) that generate synthetic

objects (such as images) that closely resemble real data.

Assuming that the GAN network's generator can convert instances of pseudo-random noise into image patterns that resemble those in the original data, we can infer that the formation mechanism of the training set's patterns is analogous to that which generates the artificially produced patterns. However, while we may understand the process of converting noise into a synthetic pattern, there still remains uncertainty about the specific noise source used to generate a given synthetic image. Therefore, the search for an inverse transformation (i.e., the Disassembler model) is an essential part of the proposed approaches in this work. The objective of the Disassembler model is to identify the specific noise source that corresponds to a given pattern. This allows the generator in a GAN to utilize the identified noise input in order to create a pattern that is highly similar to the previous one. The Generator-Disassembler pair functions similarly to the Encoder-Decoder design in Autoencoders, with the key distinction being that the former transforms input noise into an image before disassembling it back into noise, whereas the latter compresses an image into information bottleneck and then performs the reconstruction.

The initial stage of developing the anomaly detector in the proposed methodologies entails training a standard GAN model to obtain an adversarial generator model (Figure 5.1). This step aligns with the conventional GAN network training procedure. However, to enhance the generator's learning, we opted for the use of conditional GAN (cGAN). This type of network affords greater control over image generation for all available classes and promotes improved convergence during model optimization

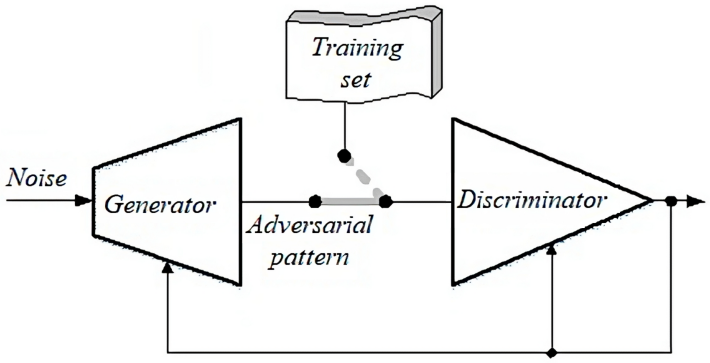


Figure 5.1: cGAN - training phase (simplified)

The second step involves training the Disassembler model that can convert the

generated adversarial image back into noise that is comparable to the original source of the adversarial image.

The generator model takes a one-hot encoded vector c (using one-hot coding) that specifies the image's class, along with a vector n containing pseudo-random numbers from a designated probability distribution (detailed in chapter 5.1), as its input. The output of the trained generator is an adversarial image $\tilde{x}$ that bears a resemblance to the patterns in the original set x. We then fetch the generated synthetic fake to the input of the Disassembler model, which ideally produces a vector comprising an estimation of the class $\tilde{c}$ and an estimation of the pseudo-random number vector $\tilde{n}$ (it depends on the final cost function). With the one-hot coding for classification, we can utilize the softmax function to train this segment. For the second part, we can use the Mean Squared Error (MSE) as the loss function to train the pseudo-random numbers. Once the Disassembler model has been trained, and we achieve a low $\|n - \tilde{n}\|$ value while also ensuring that the components of the n vector originate from the specified probability distribution, the components of the $\tilde{n}$ vector will also belong to the same probability distribution. In this regard, applying an adversarial pattern to the Disassembler's input would produce pseudo-random numbers that originate from the original probability distribution.

Nowadays, the study of pseudo-random number generators is essential in cryptography, as the use of a poor-quality generator can result in easy code-breaking. Pseudo-random number generators with a uniform distribution form the basis, and well-established tests exist for their uniformity and independence. For the proposed method, a generator of pseudo-random numbers with a uniform distribution is also considered (see Chapter 5.1 for more details). While the reconstruction of noise belonging to the uniform distribution demonstrates the quality of the Disassembler's training, we cannot determine the cost function's gradient. Therefore, for training the Disassembler, we are limited to adversarial patterns only, for which we know the original and reconstructed realization of the noise, and the MSE can serve as the primary loss function (Figure 5.2).

The fundamental assumption of the proposed method is that any anomalous patterns fed into the Disassembler will be converted into non-uniform noise (Figure 5.3).

The next question to explore is whether the Disassembler, which is trained to convert

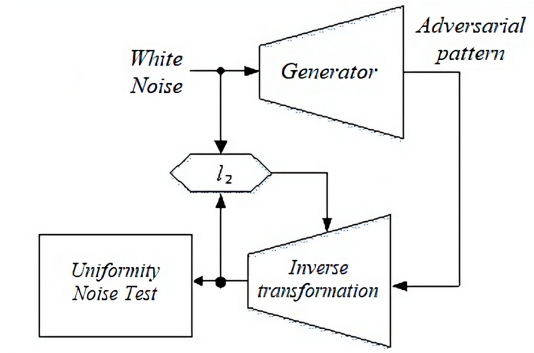


Figure 5.2: Inverse Transformation - training phase with adversarial images

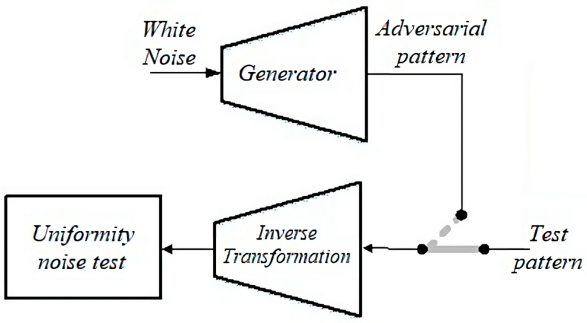


Figure 5.3: Inverse Transformation - inference for outliers

adversarial patterns into uniform noise, can also transform the patterns of the training set (true images) into white noise. We speculate that it may not, and it will be necessary to train the Disassembler using samples from the training set (Figure 5.4).

However, in the case of original images, the source noise of their creation is unknown. Therefore, we cannot utilize the mean squared error between the source and reconstructed noise as a loss function for the Disassembler. One potential solution is to input the reconstructed noise to the trained generator, which should output an image resembling the image at the input of the Disassembler. This idea can be applied not only to

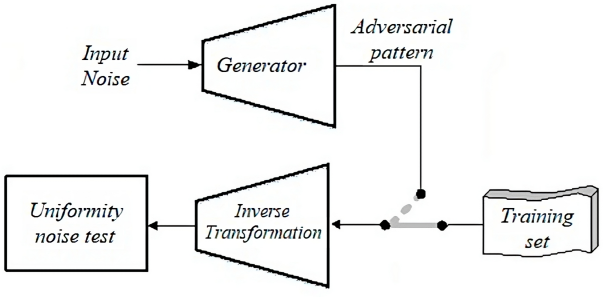


Figure 5.4: Inverse Transformation - inference for original images

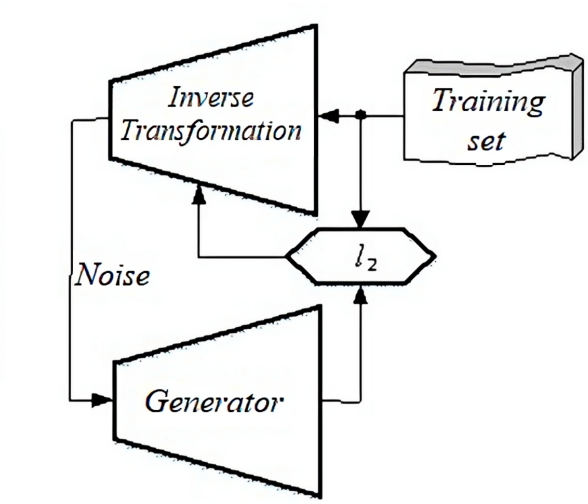


Figure 5.5: Inverse Transformation - training phase with original images

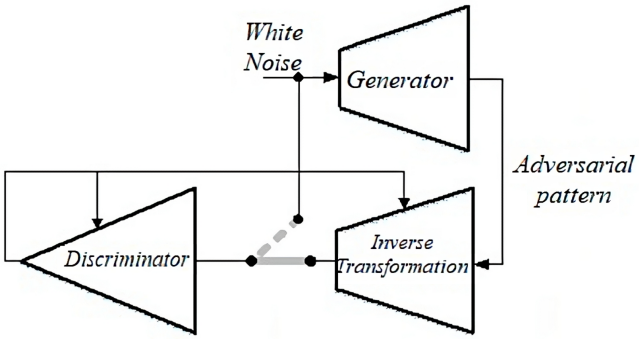


Figure 5.6: Using Discriminator to test noise uniformity

generate adversarial images but also to images from the training set. Thus, the third step of the Disassembler's training phase could potentially follow this approach, Figure 5.5).

An alternative approach to evaluating the properties of the reconstructed noise, rather than relying on MSE between the original and reconstructed noise, involves the training of a GAN-Discriminator specifically, for this very task. By doing so, the discriminator model serves as a form of noise critic (Figure 5.6).

When the Discriminator is trained to distinguish between the initial noise at the Generator's input and the noise after being processed by the Disassembler, and simultaneously the Disassembler is trained to make sure that the Discriminator cannot perceive any difference, the result will be a Disassembler capable of converting adversarial patterns into uniform noise and a Discriminator able to distinguish the reconstructed noise from known adversarial patterns and other images.

5.1 Generator utilizing uniform noise

GANs utilize pseudo-random values drawn from a standardized normal probability distribution to generate adversarial patterns. As previously discussed, the proposed anomaly detection approach operates on the principle that the inverse transformation of the test image generates patterns from a uniform distribution. Hence, the Generator network must be explicitly designed to produce adversarial patterns from a uniform distribution. In Chapter 6.3, the initial attempt to use a uniform distribution on the interval $[0, 1]$ did not yield satisfactory results. The generated adversarial patterns exhibited low variability not only for different realizations of the vector of pseudo-random numbers but also for different classes. It was concluded that the variability of the generated patterns is linked to the variability of the used pseudo-random numbers. The variability of the generated images can be determined by the entropy of the distribution from which the patterns are generated. To provide a point of reference, we will use a random variable R with a normal distribution having a mean value of $E(R) = a$ and variance of $D(R) = \sigma^2$, with a corresponding probability distribution density:

$$f_R(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}} \tag{5.1}$$

While the entropy of a continuous random variable increases to infinity, certain distributions have a bounded definite integral, which can be directly interpreted as entropy. The normal distribution is an example of such a distribution [181].

$$H(R) = -\int_{-\infty}^{\infty} f_R(x) \ln f_R(x) dx = -\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}} \ln \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}} dx = \ln \sigma\sqrt{2\pi e} \tag{5.2}$$

For the sake of simplicity, we utilize the natural logarithm to express entropy in $[nat]$ units. It is worth noting that entropy is not dependent on the mean value, but rather solely on the variance (or standard deviation - σ), as illustrated in figure 5.7.

It is worth noting that, unlike the entropy of discrete random variables, the entropy defined like this may not be non-negative $\sigma < \frac{1}{\sqrt{2\pi e}}$. Additionally, we can determine the entropy of a random variable Q with a uniform distribution having a mean value of m and an interval width of h, i.e., with the probability distribution density:

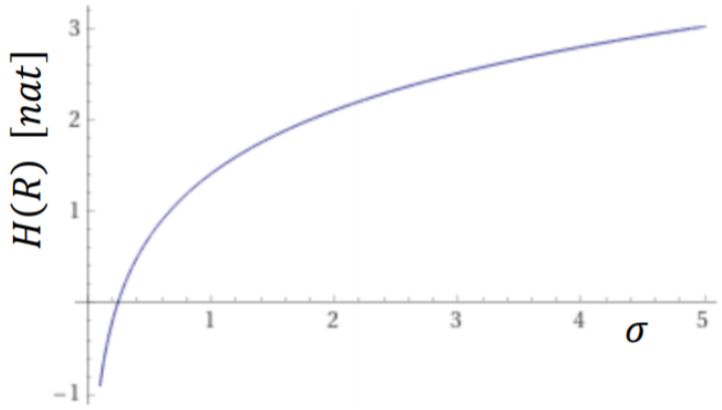


Figure 5.7: The entropy of a random variable following a normal distribution, with the standard deviation σ.

$$f_Q(X) = \begin{cases} 1/h, & x \in [m - h/2, m + h/2] \\ 0, & x \notin \lceil m - h/2, m + h/2 \rceil \end{cases} \tag{5.3}$$

$$H(Q) = -\int_{-\infty}^{\infty} f_Q(x) \ln f_Q(x) dx = -\int_{m-h/2}^{m+h/2} \frac{1}{h} \ln \frac{1}{h} dx = \ln h \tag{5.4}$$

It should be noted that the value of the integral outside the interval $x \in [m - h/2, m + h/2]$ is zero. Furthermore, as with the previous example, the entropy value for $h < 1$ may also be negative (Figure 5.8).

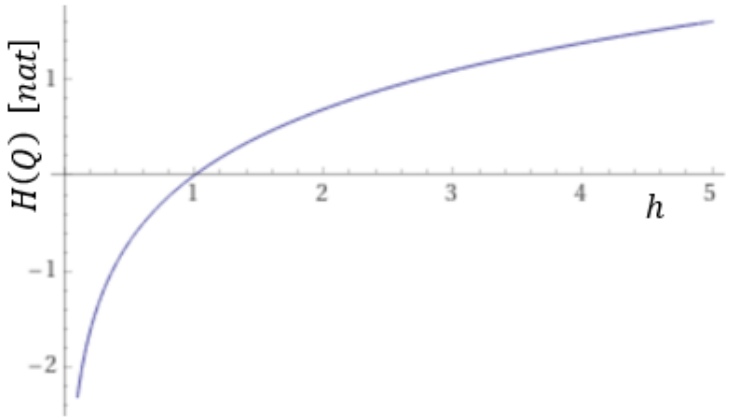


Figure 5.8: The entropy of a random variable with a uniform distribution, having an interval of length h.

The difference of random variables can be expressed by calculating the difference between their respective entropies:

$$H(Q) - H(R) = \ln h - \ln \sigma\sqrt{2\pi e} = \ln \frac{h}{\sigma\sqrt{2\pi e}} \tag{5.5}$$

The determining factor is the ratio $x = h/\sigma$, as depicted in Figure 5.9.

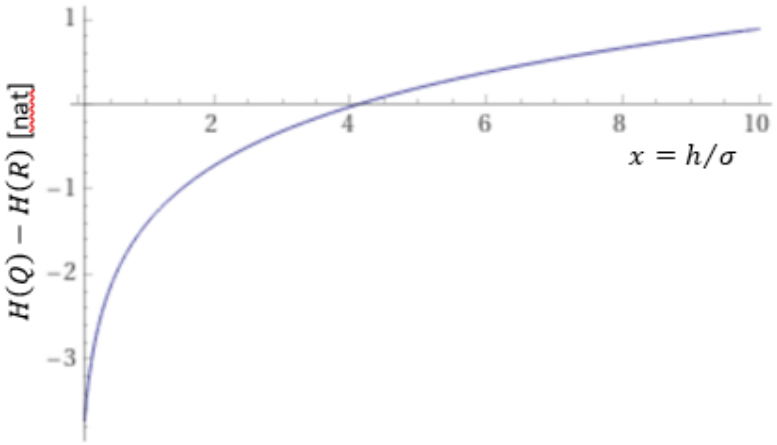


Figure 5.9: The difference of random variables with uniform and normal distribution expressed by the difference of Shannon entropies

One possible approach to selecting a uniform distribution is to ensure that both distributions have the same mean ($m = a$) and entropy ($H(Q) = H(R)$):

$$\ln h = \ln \sigma\sqrt{2\pi e} \Rightarrow h = \sigma\sqrt{2\pi e} \tag{5.6}$$

For $\sigma = 1$, the interval's width is $h = \sqrt{2\pi e} \cong 4{,}13273$. It is important to note that expressing the difference between random variables in terms of the difference between Shannon entropies does not impact the difference in their respective mean values.

It is a reasonable expectation that the degree of dissimilarity in variability between generated numbers should not be negative. This requirement is grounded in the notion that any measure of dissimilarity should be a non-negative quantity. This requirement can be fulfilled by Rényi information [182], which is also referred to as relative entropy (differential entropy, Kullback-Leibler divergence, often used in Variational AE [127]).

$$\begin{gathered} H(Q \parallel R) = \int_{-\infty}^{\infty} f_Q(x) \ln \frac{f_Q(x)}{f_R(x)} dx \\ f_R(x) = \varnothing \approx h_{Q\|R}(x) = \varnothing \end{gathered} \tag{5.7}$$

The relative entropy in NATs of the assessed random variable Q, which follows a uniform distribution, with respect to the reference random variable R, can be expressed

as:

$$H(Q \parallel R) = \int_{-\infty}^{\infty} f_Q(x) \ln \frac{f_Q(x)}{f_R(x)} dx = \int_{m-h/2}^{m+h/2} \frac{1}{h} \ln \frac{\frac{1}{h}}{\frac{1}{\sigma\sqrt{2\pi e}} \frac{(x-a)^2}{2\sigma^2}} dx \tag{5.8}$$

$$H(Q \parallel R) = \frac{1}{2} \left(\frac{m-a}{\sigma} \right)^2 + \frac{1}{24} \left(\frac{h}{\sigma} \right)^2 + \ln \frac{\sigma\sqrt{2\pi}}{h} \qquad [nat] \tag{5.9}$$

Entropy in „Shannon“ (binary logarithm):

$$H(Q \parallel R) = \left(\frac{1}{2} \left(\frac{(m-a)}{\sigma} \right)^2 + \frac{1}{24} \left(\frac{h}{\sigma} \right)^2 + \ln \frac{\sigma\sqrt{2\pi}}{h} \right) \log_2 e \qquad [Sh] \tag{5.10}$$

- Note 1:

$$\lim_{\frac{h}{\sigma} \to 0} H(Q \parallel R) = \lim_{\frac{h}{\sigma} \to \infty} H(Q \parallel R) = \lim_{\frac{m-a}{\sigma} \to \infty} H(Q \parallel R) = \infty$$

- Note 2: relative entropy is not symmetrical, i.e.

$$f_Q(x) \neq f_R(x) \Rightarrow H(Q \parallel R) \neq H(R \parallel Q)$$

Up until now, a considerable amount of experience has been gained in working with GAN networks that employ a normal distribution. However, for the proposed scenario, it is important to utilize a uniform distribution. This brings up an interesting question about when the relative entropy of a random variable that follows a uniform distribution will be minimal compared to a reference random variable that follows a normal distribution. The answer is given by the following statement:

The Theorem:

If a random variable Q follows a uniform distribution on an interval of length h and has a mean of $E(Q) = m$, with respect to a reference random variable R that follows a normal distribution with mean $E(R) = a$ and variance $E(R) = \sigma^2$, then the relative entropy between them is minimal when both distributions have identical mean and variance:

$$m = a, h = \sigma 2\sqrt{3}$$

Proof. : Calculate the minimum relative entropy:

$$\frac{\partial H(Q \parallel R)}{\partial m} = 0 \rightarrow m = a$$
$$\frac{\partial H(Q \parallel R)}{\partial h} = \frac{\partial}{\partial h} \left(\frac{1}{24} \left(\frac{h}{\sigma} \right)^2 + \ln \frac{\sigma\sqrt{2\pi}}{h} \right) = 0 \rightarrow \frac{h}{\sigma} = 2\sqrt{3}$$

The variance of a random variable with a uniform distribution on an interval of length h is:

$$D(Q) = \int_{m-h/2}^{m+h/2} \frac{1}{h}(x-m)^2 dx = \int_{-h/2}^{h/2} \frac{1}{h}x^2 dx = \frac{h^2}{12}$$

The uniform distribution with the lowest relative entropy with respect to the normal distribution with variance $D(R) = \sigma^2$ has an interval length of $h = \sigma 2\sqrt{3}$ and a variance of $D(Q) = \frac{(\sigma 2\sqrt{3})^2}{12} = \sigma^2 = D(R)$ □

The minimum relative entropy magnitude is:

$$H(Q \parallel R)_{min} = \frac{1}{2}\left(1 + \ln\frac{\pi}{6}\right) \cong 0{,}1764852 \qquad [nat]$$
$$H(Q \parallel R)_{min} = \frac{1}{2}\left(1 + \ln\frac{\pi}{6}\right)\log_2 e \cong 0{,}2546143348 \qquad [Sh]$$

The following is the relative entropy for the unit interval of the uniform distribution $x \in [0,1]$ with respect to the standardized normal distribution:

$$m = \frac{1}{2}, h = 1 \rightarrow H = \ln\sqrt{2\pi} + \frac{1}{4} + \frac{1}{24} \cong 1{,}2106[nat]$$

The equation 5.9 shows that uniform distributions with two different interval lengths have the same relative entropy with respect to the normal distribution. For instance, centered uniform distributions with two different interval lengths $h1 \cong 0{,}95808; h2 \cong 6{,}966506$. have relative entropy $H = 1[nat]$ with respect to the normalized normal distribution. The following figures (Fig. 5.10, Fig. 5.11) depict the magnitude of relative entropy using substitution:

$$x = \frac{m-a}{\sigma}, y = \frac{h}{\sigma} \tag{5.11}$$

In the case of relative entropy with respect to the normalized normal distribution ($a = 0$,$\sigma = 1$), the substitution variables directly indicate the parameters of the uniform distribution $x = m, y = h$.

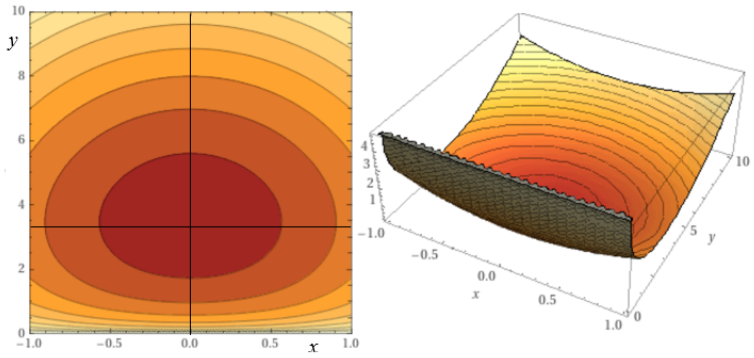


Figure 5.10: Relative entropy of a random variable with a uniform distribution with respect to a normal distribution

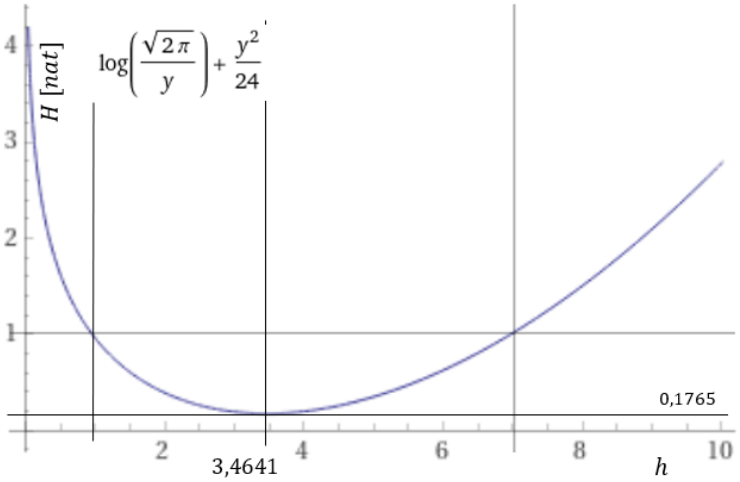


Figure 5.11: Relative entropy of a random variable with a uniform centered distribution with respect to the normalized normal distribution

In order to comprehend the relationship between the variability of the created adversarial patterns and the relative entropy of the reference and test noise, we assess the relative entropy for the following combinations:

- **1**. reference noise R has a normalized normal distribution, test noise Q has a general uniform distribution.

$$H(Q \parallel R) = \int_{-\infty}^{\infty} f_Q(x) \ln \frac{f_Q(x)}{f_R(x)} dx = \int_{m-h/2}^{m+h/2} \frac{1}{h} \ln \frac{\frac{1}{h}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}} dx, \tag{5.12}$$

$$H(Q \parallel R) = \frac{m^2}{2} + \frac{h^2}{24} + \ln \frac{\sqrt{2\pi}}{h}, [nat]. \tag{5.13}$$

- **2**. reference noise R has a normalized normal distribution, test noise Q has a

normal distribution

$$H(Q \parallel R) = \int_{-\infty}^{\infty} f_Q(x) \ln \frac{f_Q(x)}{f_R(x)} dx = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}} \ln \frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}} dx, \tag{5.14}$$

$$H(Q \parallel R) = \frac{1}{2}(a^2 + \sigma^2 - 1) + 2\left(\frac{a}{\sigma}\right)^2 + \ln \frac{1}{\sigma}, [nat]. \tag{5.15}$$

- **3**. reference noise R has a uniform distribution with zero mean and standard deviation equal to one, test noise Q has a uniform distribution

$$H(Q \parallel R) = \int_{-\infty}^{\infty} f_Q(x) \ln \frac{f_Q(x)}{f_R(x)} dx = \int_{r^-}^{r^+} \frac{1}{h} \ln \frac{\frac{1}{h}}{\frac{1}{2\sqrt{3}}} dx \tag{5.16}$$

$$r^+ = min(m + h/2, \sqrt{3}), r^- = max(m - h/2, \sqrt{-3}),$$

$$H(Q \parallel R) = \begin{cases} \frac{\Delta}{h} \ln \frac{2\sqrt{3}}{h}, \Delta > 0 \\ 0, \qquad \Delta \leq 0 \end{cases}, [nat], \Delta = r^+ - r^- \tag{5.17}$$

For uniform distributions, the relative entropy is zero when the distributions are identical, but also when the intervals with non-zero values do not intersect. This general rule applies to the probability distribution's densities, the test or reference that generates the random variables, which acquire non-zero values at finite intervals or unification of finite sub-intervals.

Tables 5.1, 5.2, and 5.3 show calculated relative entropy between different data distributions. They provide examples of how we can control the variability of the produced adversarial patterns since changing the input data distribution to any other than training has a direct impact on image fidelity and diversity. It is done by modifying the mean value $E(Q)$ and standard deviation σ. Sampling the input noise vectors from two different distributions that have a relative entropy close to zero produces images of similar quality (see Figure 6.6)

E(Q)|σ	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.8**	**2.0**
10	5 051,13	1 300,50	605,75	362,54	250	188,93	152,18	128,43	112,26	100,81
5	1 263,63	325,50	151,58	90,67	62,5	47,26	38,15	32,34	28,46	25,81
0	1,129	0,496	0,191	0,043	0	0,038	0,144	0,310	0,532	0,807
-5	1 263,63	325,50	151,58	90,67	62,5	47,26	38,15	32,34	28,46	25,81
-10	5 051,13	1 300,50	605,75	362,54	250	188,93	152,18	128,43	112,26	100,81

Table 5.1: Relative entropy between the generalized normal distribution and the normalized normal distribution

E(Q)|σ	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.8**	**2.0**
10	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	-0,020
0	1,609	0,916	0,511	0,223	0	-0,152	-0,240	-0,294	-0,327	-0,347
-5	0	0	0	0	0	0	0	0	0	-0,020
-10	0	0	0	0	0	0	0	0	0	0

Table 5.2: Relative entropy between the uniform distribution and the test uniform distribution

E(Q)|σ	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.8**	**2.0**
10	51,31	50,67	50,37	50,22	50,18	50,21	50,32	50,49	50,71	50,98
5	13,81	13,17	12,87	12,72	12,68	12,71	12,82	12,99	13,21	13,48
0	1,31	0,67	0,37	0,22	0,18	0,21	0,32	0,49	0,71	0,98
-5	13,81	13,17	12,87	12,72	12,68	12,71	12,82	12,99	13,21	13,48
-10	51,31	50,67	50,37	50,22	50,18	50,21	50,32	50,49	50,71	50,98

Table 5.3: Relative entropy between the normalized normal distribution and the test uniform distribution

Table 5.1 presents the relative entropy values between the generalized normal distribution and the normalized normal distribution. Meanwhile, Table 5.2 illustrates the symmetry of the distributions in relation to the mean value $E(Q)$. We emphasize that the same variability does not necessarily translate to identical patterns in the generated images. Table 5.3 highlights the variable correctness by indicating zero entropy for identical distributions.

5.2 Through controllable generation to anomaly detection

As previously mentioned, the generator model has a different objective from the discriminator, which functions as a classifier or critic. The primary goal of the generator is to represent various classes in general, rather than just distinguish between them. In terms of probabilities, the generator aims to determine $p(x|y)$, which is the likelihood that a generated sample ($y = class$) results in the corresponding image x. However, the output space for each possible class is extensive, making the generator model particularly challenging. This task can be much more challenging than discrimination, particularly given the increasing complexity of training data. It is straightforward to differentiate between two potential classes, but comprehending all the characteristics of every potential variation of the class is an entirely different task. However, if a generator model can achieve its objective after training, it is feasible to start producing data samples of the possible classes included in the training dataset.

5.2.1 Truncation trick in GANs as inspiration

The noise vector z plays a crucial role in ensuring that the generated images from the same class y are diverse, akin to a random seed. This vector is randomly generated, typically by sampling random numbers from a uniform distribution between 0 and 1 or from a normal distribution, denoted as $z \sim N(0,1)$. Here, the value 0 represents the mean of the normal distribution, while 1 denotes its variance. In practice, the vector z is usually larger than a single value to allow for more combinations, and there is no predetermined optimal size. However, a common practice is to set its dimension to at least 100. Some implementations may use powers of 2, such as 128 or 512, but the

choice is arbitrary and primarily depends on the requirement for a large number of possibilities.

During our experiments with different noise vectors and distributions, we propose a new method for training inverse transformations based on a well-known concept used to fine-tune the output of a generator network. This concept is called the truncation trick, and it's essential to understand it as a way to balance the quality and diversity of image samples. The truncation trick works by controlling the distribution of the noise vector randomly sampled by the generator network. Instead of using a standard normal distribution, we use a truncated normal distribution. The truncation threshold parameter determines the range of values accepted in the distribution. A lower truncation threshold results in more diverse but lower-quality generated samples, whereas a higher truncation threshold results in higher-quality but less diverse samples. It's important to note that during training, the model becomes more familiar with noise vectors within a standard deviation from the mean of the normal distribution. Therefore, it's likely to model these areas more accurately, resulting in realistic but less diverse image samples. This is the trade-off between fidelity and diversity. In other words, by adjusting the truncation threshold, we can control the balance between the quality and variety of the generated image samples.

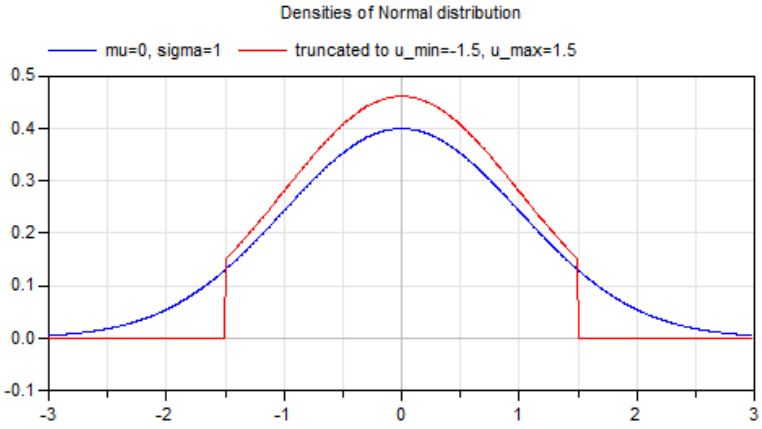


Figure 5.12: Example of truncated normal distribution [183]

Based on the so far theory of GANs and according to the truncation trick, modeling the input distribution has a direct impact on the output prediction (generated image). In order to generate both altered adversarial images and realistic ones, it is possible to broaden the concept of the truncation trick as a truncation and extension throughout

the data distribution. Thus, this approach allows generating of synthetic anomalies or authentic fakes from original data based on input data. More specifically, with the aid of the cGAN model (used in this work), it is feasible to purposefully create a highly varied set of altered anomalous images or to produce realistic samples from the training data distribution space.

As the input noise vector z is generated from a normal distribution, the model is more likely to encounter z values within one standard deviation from the mean compared to those at the tails of the distribution. This phenomenon occurs during the network's training process. It is important to note that the model does not encounter any samples of z outside the distribution. Consequently, during training, the model becomes familiar with certain noise vectors and tends to generate more realistic results in those areas. However, as the model approaches the outskirts of the distribution, the realism of the images begins to decline, and the diversity of images increases.
In the event that an end-user inputs a noise vector sampled from a data distribution that is considerably distant from the training distribution, the network will generate adversarial images that differ greatly from what the user intended. This occurs because the model was not trained on this type of distribution. When the input noise is sampled significantly far from the training distribution, the resulting images appear as skewed artifacts without any discernible pattern. As the samples move closer to the training distribution, the output images take on better-defined shapes. At the tails of the training distribution, exists a threshold event whereby realistic images degrade to anomalies that lack clear interpretation and vice versa. However, this is not a straightforward boundary per se. There are different approaches to how is possible to set these boundaries within the data. The experimental approach based on the classification results from humans and Neural nets is proposed in this paper [184]. Another approach is to treat the training data distribution as a unit hypersphere. The Gaussian distribution is also called a spherical normal and denoted as $z \sim N(0, I)$ where the I represents the identity matrix and it means the variance is 1 in all dimensions.

The idea of detecting anomalies by placing the original data inside a unit hypersphere while anomalies are outside this hypersphere is based on the concept of distance-based anomaly detection. The main argument behind this approach is that in many cases, normal data points are expected to be tightly clustered around a central point or region,

while anomalies are expected to be far away from this central region. By placing the normal data points inside a unit hypersphere, we can define a boundary beyond which any data point is considered an anomaly.

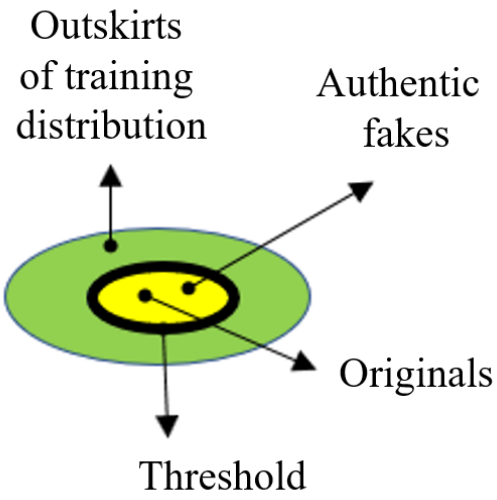


Figure 5.13: Illustration of data distribution space

The main argument behind this approach is that in many cases, normal data points are expected to be tightly clustered around a central point or region, while anomalies are expected to be far away from this central region. By placing the normal data points inside a unit hypersphere, we can define a boundary beyond which any data point is considered an anomaly.

This approach has several advantages, including:

- The method can be applied to a wide range of data types and can be easily adapted to different dimensionalities.
- The boundary of the unit hypersphere provides a clear and intuitive interpretation of what is considered normal or anomalous.

However, there are also some limitations to this approach. For example, the hypersphere method may not work well with data that does not exhibit a clear central region, or in cases where the anomalies are not well separated from the normal data. Additionally, the method assumes that the normal data points are distributed uniformly within the hypersphere, which may not always be the case in practice. In the specific

scenario of synthetically created data by the generator model, the last problem is the most important.

5.2.2 Probability distribution of Radius

If the distribution of points in the hypersphere is to be uniform throughout the entire volume, there must be the same probability of a point falling into each intermediate sphere in which the inner sphere has a radius r and the outer sphere $r+dr$. The volume of the intermediate sphere is

$$\Delta V_n(r) = S_n(r)dr = c_n r^{n-1}dr \tag{5.18}$$

where $S_n(r)$ is the surface of an n-dimensional hypersphere with radius r and:

$$c_n = \frac{n\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \tag{5.19}$$

Let $\varphi(r)$ be the density distribution of the probability of the appearance of a radius with the value $r \in [0,1]$. If the guide acquires a value from the interval $\rho \in [r,r+dr]$, then the point falls into an intermediate area of volume $S(r)dr$. Since these phenomena occur simultaneously (the first phenomenon implies the second), the probabilities of their occurrence are proportional, i.e. $\varphi(r)dr = kS_n(r)dr$, i.e. $\varphi(r) = kS_n(r)$, where k is the normalization constant based on the condition:

$$\varpi(r) = kS_n(r) = \frac{n}{c_n}c_n r^{n-1} = nr^{n-1}, r \in [0,1] \tag{5.20}$$

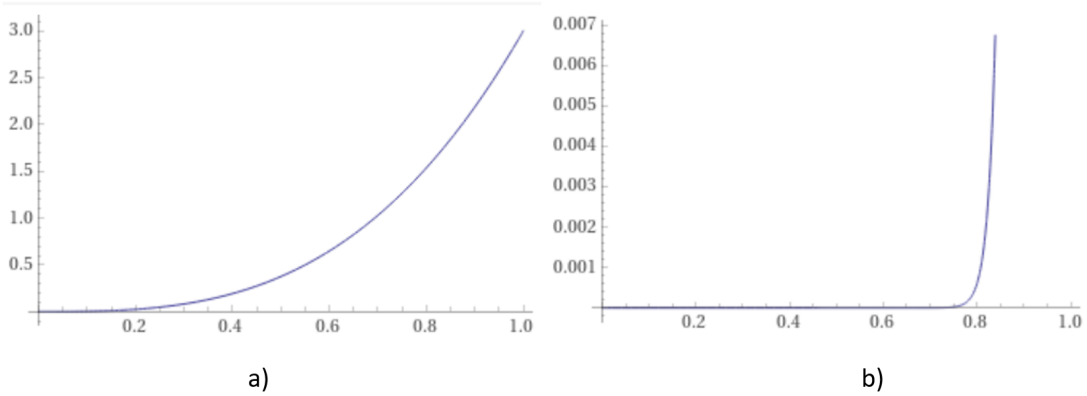


Figure 5.14: Density distribution of radius for a)m=3 and b) m=52

Figure 5.14 points out the consequence of the curse of dimensionality. To achieve uniform distribution, the vast majority of points are generated at the outer rim of the

hypersphere. Therefore, the requirement of equal distribution is unrealizable. (This domain needs further analysis and investigation in order to overcome this problem.)

However, there is an option to sample the data from the unit hypersphere by slicing from this region, not to sample from the entire space. It is possible to do this by using a uniform distribution with an interval close to the training distribution (the relative entropy between these distributions is the lowest - see Chapter 5.1). Given the ability to manipulate the generative process of anomalies and authentic fakes, it becomes feasible to train an inverse transformation - a Disassembler model, to reconstruct the noise from images based on their origin, representing either the inner or outer space of the hypersphere. (see Figure 5.15)

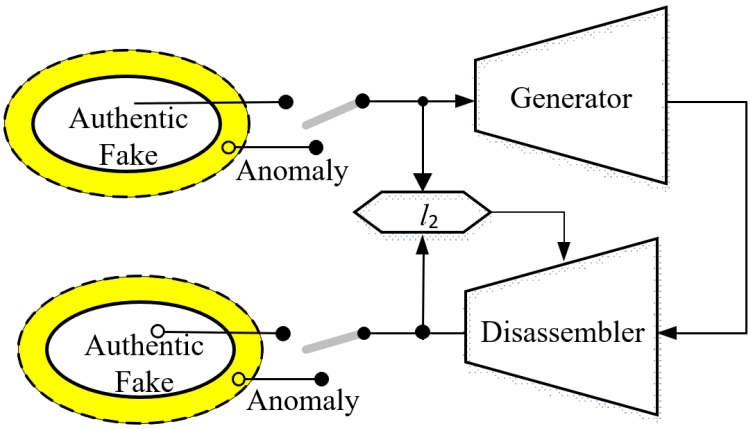


Figure 5.15: The Training phase of Inverse Transformation with the use of trained Generator model from GAN

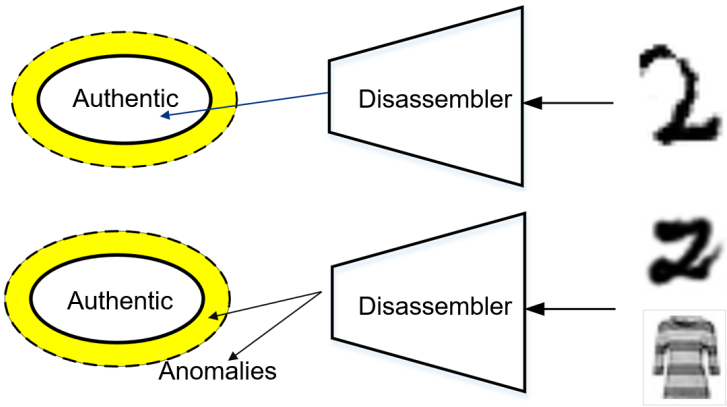


Figure 5.16: The phase of testing the inverse transformation with diverse data types.

Upon successful completion of the training, the inverse transformation must have

the capability to reconstruct the noise from the input image in order to represent its origin. The location of the reconstructed image, whether it is situated inside or outside of the hypersphere, can be used to differentiate between the anomalies and the original images. If the input image is sourced from a dissimilar dataset or is greatly distorted, the distance from the center of the hypersphere should be greater compared to an image that bears a closer resemblance to the original.

Chapter 6

Deep Generative Detection: Experimental Evaluations

In the initial stages of anomaly detection research, our focus was primarily on verifying the practical use of generative models for detecting anomalies using conventional methods. For the majority of our deep learning experiments, we used the MNIST[158] and Cifar10[185] database, which is well-suited for testing various research techniques and methods on real-world data with minimal pre-processing and formatting requirements. It is crucial to be able to assess the quality of generated data with ease when employing generative models.

6.1 Autoencoder-based anomaly detection

The initial experiments in the realm of anomaly detection utilizing Deep Generative Models (DGM) involved the application of Autoencoders (AE) to identify anomalous data samples within a test data input. AE's anomaly detection mechanism is founded on the principle of utilizing the reconstruction error generated by the AE as a metric for detection. Specifically, we trained an Autoencoder using the MNIST dataset.
Upon completing the training process, we proceeded to introduce three different data sources into the AE model. Among these sources, we designated two as anomalous and then closely monitored the resulting impact on the computed reconstruction error. The fundamental assumption was that the original database would produce a comparatively smaller reconstruction error, owing to the fact that it had been used for training

purposes. The CIFAR-10 and Fashion-MNIST databases were chosen as our sources of anomalous data, given that we anticipated a higher level of AE reconstruction error due to the fact that the model had not been previously exposed to this data. As it turned out, our assumption was indeed validated. Figure 6.1 illustrates the differences in the reconstruction error generated from the anomalous CIFAR-10 and Fashion-MNIST databases as compared to the original data. Notably, the reconstruction error for the anomalous data sources is found to be significantly greater. Furthermore, the range of these errors is noticeably larger, which enabled us to successfully distinguish anomalous samples from genuine ones. This outcome has confirmed the validity of our initial assumption.

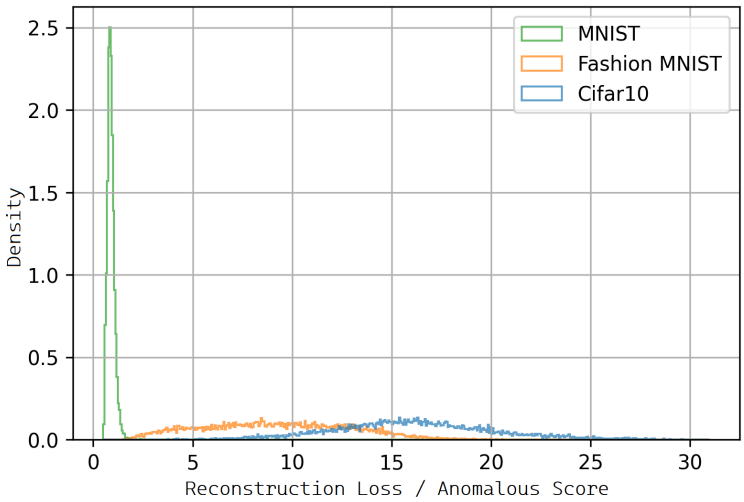


Figure 6.1: Test phase - Histogram of reconstruction errors during outlier detection / MNIST

We have conducted this experiment two more times with a slight variation, wherein we trained the AE model on the data sourced from CIFAR10 and Fashion-MNIST, both of which were previously designated as anomalous. In doing so, we were able to observe how the process of anomaly detection operates with data of varying complexity.

Based on the obtained histograms (Fig. 6.2), it is evident that the efficacy of anomaly detection is contingent on the complexity of the original image and the anomalous input. For instance, when the AE model is trained using the most complex CIFAR10 database, detecting less-complex outliers can be challenging, although we can still establish a clear boundary between the original and anomalous data. A similar outcome is observed

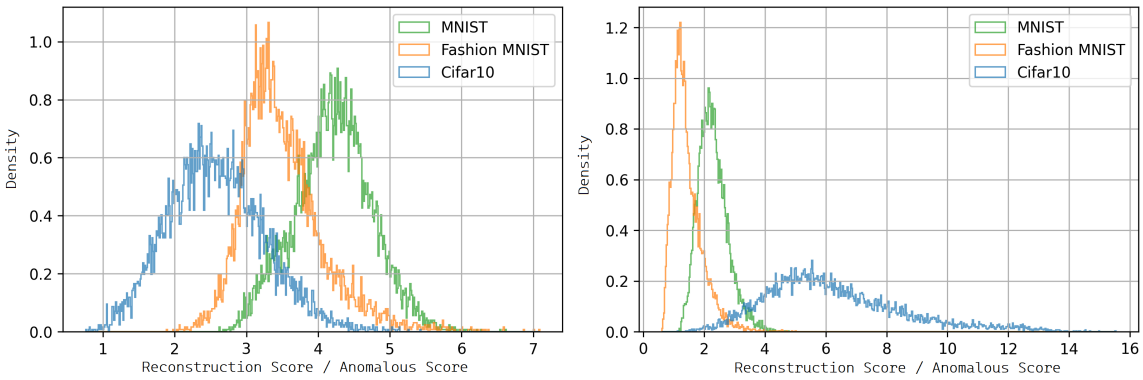


Figure 6.2: Test phase - Histograms of reconstruction errors during outlier detection / CIFAR-10(right) and F-MNIST(left)

when using the AE model trained on the F-MNIST database. In this instance, while the reconstruction error for original data is still the lowest, distinguishing between originals and anomalies is more difficult, particularly when compared to the MNIST database which we designated as the anomalous set in this instance. As the complexity of the original data exceeds that of the anomalous input, the AE model faces increasing difficulty in distinguishing between original and anomalous data. When the model is capable of reconstructing complex images with a high degree of accuracy, it is reasonable to expect that relatively simpler images can also be reconstructed with minimal errors, even if they were not seen during the training phase. Conversely, reliable anomaly detection is observed when the anomalous input is more complex than the original data.

These results were also published in the paper [89] (see List of Publications at the end of work). The article focused on the topic of anomaly detection using deep generative models.

6.2 GAN-based anomaly detection

We conducted similar experiments with Generative Adversarial Networks, using a discriminator to differentiate between genuine and anomalous data. Once again, the datasets consisted of MNIST, F-MNIST, and Cifar10, and we trained the GAN network separately for each database. This allowed us to assess the effect of image complexity on anomaly detection, just as we did with the Autoencoder.

Our hypothesis was that a well-trained GAN network generator would produce

authentic fake images, which the discriminator would classify in the same way as genuine data. Any other data that the GAN network did not encounter during training would be automatically identified by the discriminator as fake or anomalous. As shown in Figure 6.3, our assumption proved to be correct. The GAN network can distinguish between genuine and anomalous data, particularly with less complex data such as the MNIST database.

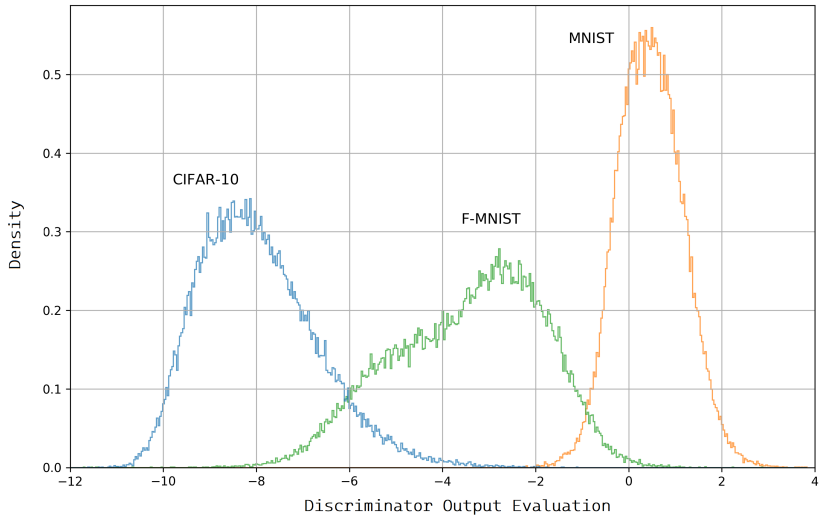


Figure 6.3: Histogram of the MNIST-trained Discriminator evaluation across various image datasets

If we replicate the experiments and use Cifar10 or F-MNIST as training sets, the results are less dependable. As seen in Figure 6.4, it becomes increasingly difficult to distinguish between a complex training set and simpler anomalous data as their relative complexity levels become more disparate. It is worth noting that various GAN types are available, and they may yield better results in outlier detection by utilizing this conventional approach (employing the trained Discriminator's intrinsic ability) compared to the base 'vanilla' GAN version.

These results were also published in the paper [89] where the article focused on the topic of anomaly detection using deep generative models.

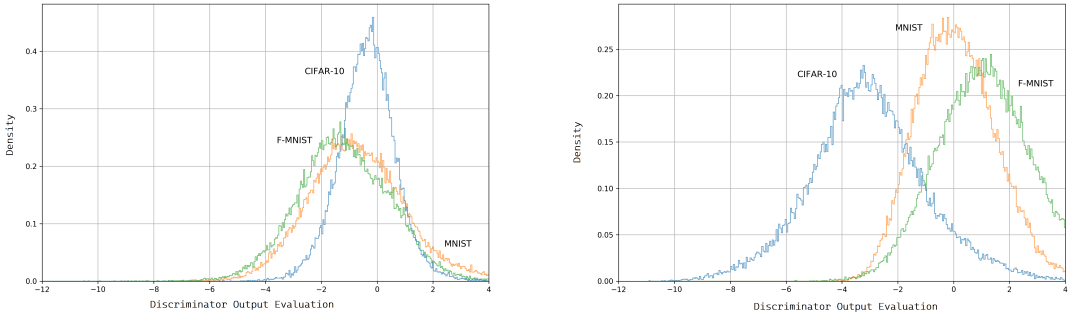


Figure 6.4: Test phase - Cifar10 trained Discriminator(right)/FMNIST trained Discriminator(left) for outlier detection

6.3 Controllable Generation: Experiments and Discussion

We conducted additional experiments in the field of anomaly detection to validate our previous assumptions and proposed a detection architecture that would not rely on data complexity for its efficiency. Our focus was on comparing individual data characteristics and utilizing significant differences between anomalous and genuine data characteristics as a metric for detecting anomalies. The proposed architecture is described in more detail in Chapter 5.

The basis of this concept lies in Hawking's definition of outliers [84]. Assuming that the counterfeit samples bear a resemblance to the original ones, and we possess knowledge of the fabrication process for the forged images, then it is impossible for the anomalies to have originated from the same process. Naturally, testing every possible input seed for the generator to confirm similarity with the test image is unfeasible. Instead, we assess the input-output pairs in reverse. To achieve this, we train the inverse function to the generator - Inverse Transformation model.

6.3.1 Generator model based on Uniformity

The training of GANs involves leveraging a normal distribution that benefits from a non-uniform distribution of the seed radius. In this case, seeds with smaller radii are employed more frequently during training, resulting in generated images that are more closely aligned with the training images. As a consequence, there may be a fuzzy

boundary between anomalous and authentic images. To address this issue, we discuss the proper training of the GAN network using uniform noise in Chapter 5.

Our goal was to achieve comparable training effectiveness with the use of a normalized normal distribution. To validate our theory, we conducted the following experiments. We trained a GAN network with uniform noise sampled from a unit interval [0,1]. Upon successful training, we observed that the generated fakes exhibited low variability, not only across different implementations of the vector of pseudorandom elements but also across different classes (Figure 6.5).

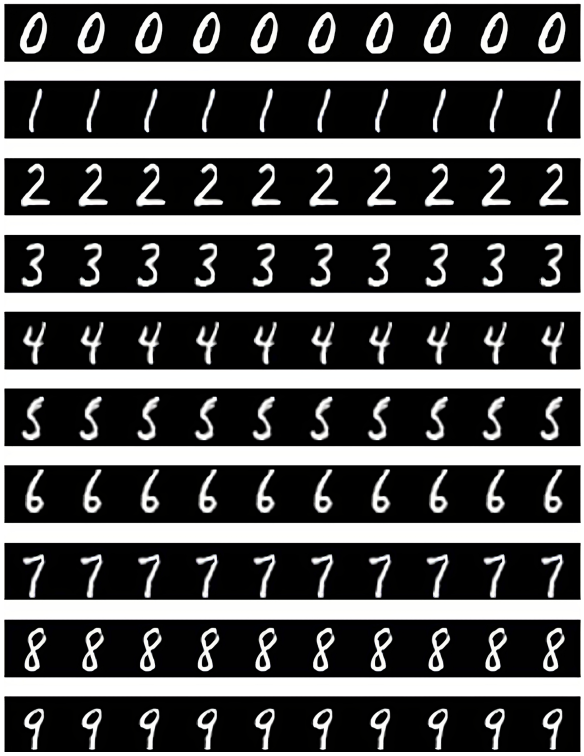


Figure 6.5: GAN model trained on Uniform distribution - Images generated from the uniform distribution with the interval $[0; 1]$

Because of this, we opted to use uniform noise with the same parameters as the normalized normal distribution - that is, with zero mean and standard deviation of one - which is commonly used in GAN training. Our experiments revealed that by manipulating the mean value and interval length of the uniform distribution, we could affect the variability of the generated patterns. Specifically, we could generate patterns that closely resembled the training set, as well as patterns that deviated significantly from the original data yet remained recognizable to human observers. By adjusting

the mean value and interval length of the uniform distribution, we could train a GAN network as effectively as we could with the normal normalized distribution, with similar results in terms of image quality. The best results were achieved when the relative entropy between a standardized normal distribution and employed training uniform distribution was close to zero (see Figure 6.6). The initial input noise from the uniform distribution on the unit interval [0,1] contained minimal information, which resulted in patterns with low variability. By changing the interval length of uniform noise we expand the information content, and we trained the GAN network to perform comparably to a model that used Gaussian noise during training.

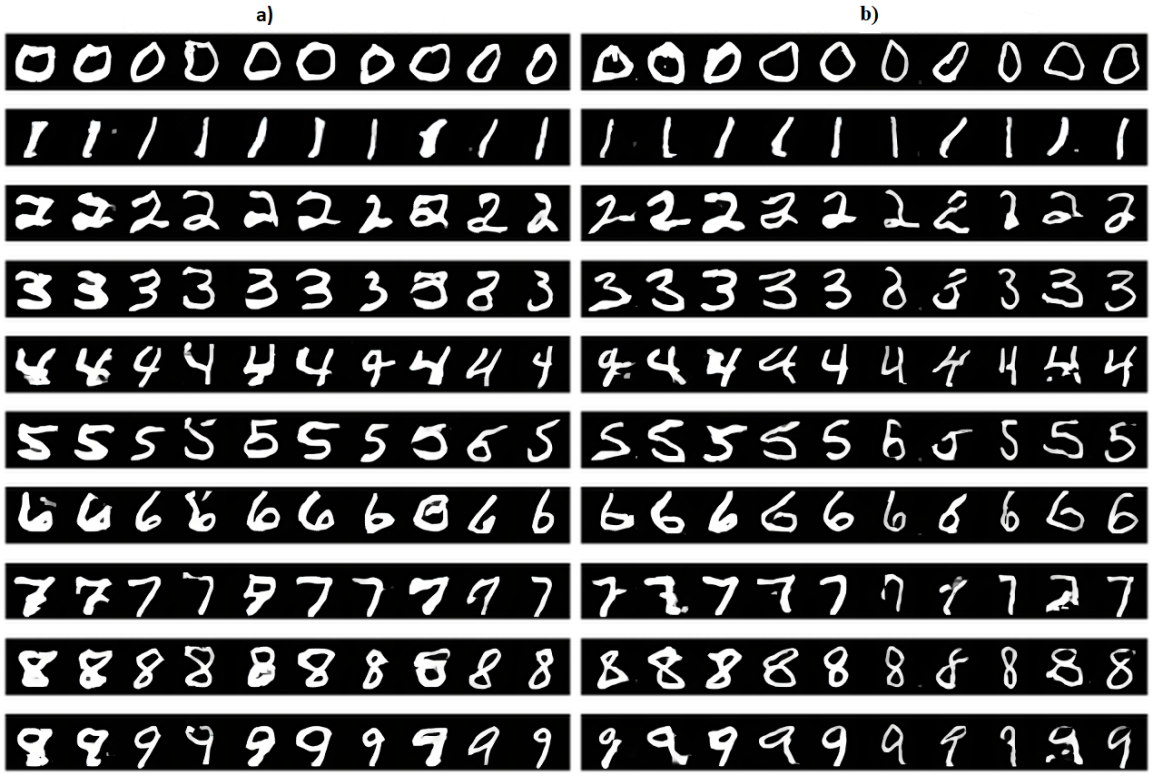


Figure 6.6: GAN model trained on Uniform distribution a) Images generated from the standard normal distribution $z \sim N(0,1)$
b) Image generated from the uniform distribution with the interval $[-\sqrt{3}; \sqrt{3}]$

Figure 6.6 displays the images generated by the GAN network that was trained using a customized uniform distribution but images were generated during inference from both types of input noise (Figure 6.6 a) and b)). Also, it provides examples of how we can control the variability of the produced adversarial patterns by changing the length of the uniform distribution interval. This ranges from patterns that resemble

the training set patterns to patterns that greatly differ from the training set patterns but are still recognizable to humans(depending on relative entropy between training and testing distributions).

Although we used the term "image similarity", its quantitative assessment is an open question with numerous approaches. The application of relative entropy (Chapter 5.1) is one of them. As an alternative, we also suggest utilizing the recognition system's confidence - Discriminator (see Chapter 6.3.2) in identifying the adversarial pattern to evaluate its similarity with the training set patterns. This means, such a system needs to be trained on original data. Overall the uniformity noise test experimentation did not deliver futile results (Chapter 6.3.2) in terms of accurate detection of outliers. Therefore we devised a different approach based on the gathered experiments and experiences with input noise manipulation. This approach is investigated in Chapter 5.2 and the main idea is about changing the way how we train the Inverse Transformation model.

6.3.2 Uniformity as anomaly indicator

The main idea behind these uniformity tests was that if the genuine fake images are created from independent uniformly distributed pseudorandom seed then genuine fake images at the input of inverse transformation must give independent uniformly distributed random numbers.

The experiment was carried out in two steps. Initially, we trained the GAN network using the MNIST training data. After successful training, we used the Generator model from GAN in the next design (see Figure 6.7) with the proposed Inverse Transformation - the Disassembler (as outlined in Chapter 5). The training of the inverse transformation was evaluated using the mean squared error (MSE) between the input noise and its reconstructed counterpart (Disassembler output).

Following training, the Disassembler model was able to accurately decompose the fake images into an output noise vector that was almost similar to the input noise for the given pattern. However, for original images, the reconstructed output noise vector cannot be compared to the input noise since we lack such information (origin of fabrication unknown). This is in line with the idea mentioned in Chapter 5. Our working hypothesis was that if we understand the mechanism of creating fake images from uniform noise, and we also comprehend the process of reconstructing uniform

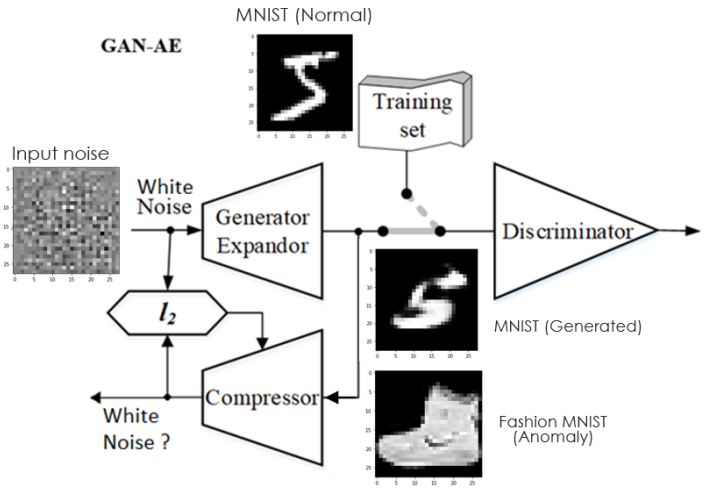


Figure 6.7: Generator-Inverse Transformation design for outlier detection

noise from fakes, then even for original images that closely resemble fakes, the same degeneration process would lead to uniform noise.

The crux of the matter is statistical tests that aim to determine whether the reconstructed noise consists of independent random variables with a uniform distribution. Hence, our primary interest was in the characteristics of the output noise, as mentioned above, and whether they were comparable to the characteristic of input noise.

We employed the Chi-squared test for testing the uniformity of noise vectors. This statistical test is specifically designed to determine whether a given sample of data is derived from a specified distribution, such as a uniform distribution. When testing for uniformity of noise vectors, the null hypothesis would posit that the noise vectors are uniformly distributed, while the alternative hypothesis would suggest otherwise. The test statistic, H, is calculated and then compared to the critical value of the chi-squared distribution. If the test statistic is greater than the critical value, the null hypothesis H_0 is rejected, indicating that the noise vectors are not uniformly distributed, indicating outliers.

$$H = \sum_{j=1}^{k} \frac{(X_j - np_j)^2}{np_j} \sim \chi^2(k-1) \tag{6.1}$$

where k is the number of classes, p_j is theoretical probability 0.1, n is the number of elements in the tested data sample, and H test statistic.

The null hypothesis under consideration can be formulated as follows:

H_0 - the reconstructed noise vectors (data) originate from a uniform distribution, thus data are not anomalous in nature.

The critical value of the chi-squared distribution χ^2 is 16.9.
In every test case, the test statistic H is compared against the critical value of the chi-squared distribution to evaluate whether the reconstructed noise vectors exhibit anomalies.

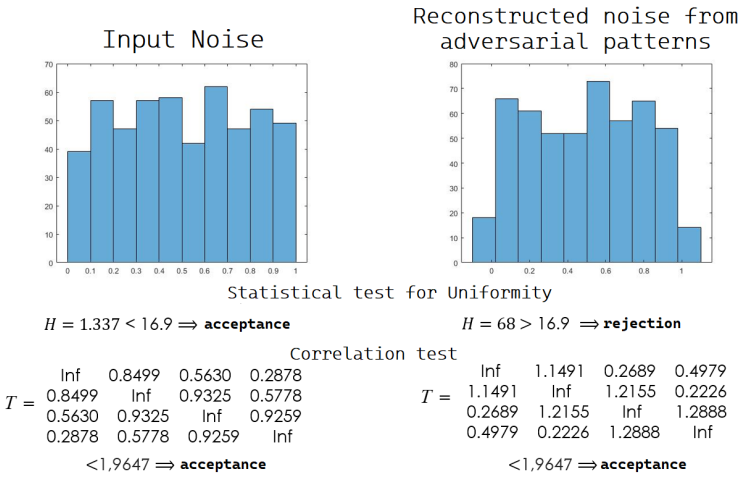


Figure 6.8: The statistical test results for uniformity: the input noise on the left and the reconstructed noise on the right.

After training the Disassembler, statistical tests were conducted, which revealed that the reconstructed noises from the generated fake patterns did not pass the uniformity test. Therefore, the Disassembler was not effectively trained to learn and preserve the characteristics of the input noise, as depicted in Figure 6.8. In addition, Figure 6.9 shows that the Disassembler trained in this way could not correctly decompose the original images into the output noise that would pass uniformity tests. These results indicate that the inverse transformation should not be trained solely on fakes created by the generator, but also on original data, as explained in Chapter 5. Moreover, the loss function MSE is not sufficient to accurately capture the data distribution during training, especially on the tails of the distribution, because the error is averaged and the training focuses more on average values than on individual ones.

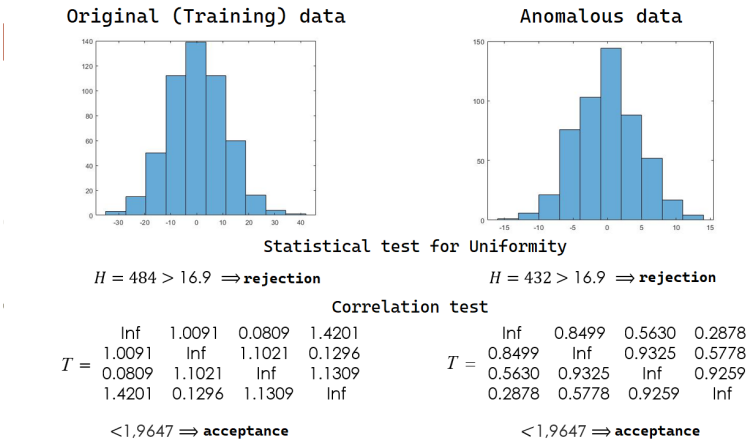


Figure 6.9: The statistical test results for decomposed original data are shown on the left, while the results for the adversarial fakes are on the right.

Scenarios	Test Statistic H	Decision
1. Input Noise	1.337	Accept null hypothesis
2. Authentic FMNIST fakes	68	Reject null hypothesis
3. Original FMNIST images	484	Reject null hypothesis
4. Anomalous MNIST images	432	Reject null hypothesis

Table 6.1: The statistical analysis for hypothesis H_0 - inference of FMNIST trained Inverse Transformation

Based on the summary presented in Table 6.1, it can be observed that the test statistics for Scenarios 2, 3, and 4 are higher than the threshold value H. This finding is significant as it indicates that the reconstructed noise vectors in these scenarios exhibit anomalies. However, only the 4th scenario was real anomalous input. At last, scenario No.1 passed the test which confirm that we used the truly uniform noise during training the inverse transformation.

Discriminator as a critic to assess the image similarity

As described in Chapter 5, training the Inverse Transformation on original data requires an alternative architecture, as the original noise vector that created the data is unknown, rendering the MSE loss unusable. To address this issue, we proposed a new architecture with a Discriminator, which is trained to distinguish between the original

noise input and the noise output from the Disassembler. The Disassembler is then trained to produce output noise that is indistinguishable by the Discriminator. The objective is to achieve a trained Disassembler that can transform both adversarial and original data into uniform noise, while the Discriminator can distinguish between the reconstructed noise from adversarial patterns and other images. Unfortunately, our hypothesis was not confirmed in this experiment. The binary cross-entropy was used as the cost function, but the implemented function takes the output values directly from the Discriminator (single output value) rather than the probability distribution. As a result, during training, the Disassembler pushed the output noises to the values of 0 and 1 with a probability of 0.5, which hindered the correct reconstruction of the noise.

Disadvantages of Uniform distribution

During our experiments, we faced challenges while training GAN networks, which is a crucial step in the proposed detection process. Training GAN networks is generally unstable and difficult because it requires solving the Nash equilibrium (stable equilibrium between the generator and discriminator). Moreover, the complexity of the task affects the level of training difficulty. In the above-mentioned case, we trained the GAN network using uniform noise, which is not commonly used for GAN training. During training, we encounter the usual issues like a mode collapse problem (more about in paper [186]) but also problems specific to using uniform noise - low diversity in the results.

To address this, we proposed an improved network architecture with two discriminators. We added an extra output with a resolution of 14x14 to the generator, which was combined with the original output with a size of 28x28. Each discriminator evaluated one output of the generator, and the training process continued as usual. Additionally, we created a secondary copy of the original training database with a resolution of 14x14 to provide auxiliary information at a lower resolution for the second Discriminator, which helped the model to train more effectively. With this modification, we achieved satisfactory results in training the GAN network. There are also other possibilities like the adoption of some of the established methods for enhancing training stability, as detailed in this paper [187].

Our experimentation focused on uniform noise due to its distinctive characteristics,

which allowed us to effectively assess the data as outliers or originals. Moreover, the n-dimensional uniform distribution is notable for its representation of the hypercube, which possesses clear boundaries between anomalous and original data. These findings were derived from a series of experiments exploring various data distributions as part of the fabrication process for images. In order to employ the GAN network trained on the normalized normal distribution, which is more convenient and effective for training purposes, it would be more appropriate to use the concept of a hypersphere. While it is possible to train the model on a normal distribution and subsequently use a uniform distribution with a relative entropy close to zero for image fabrication during inference, this approach has limitations. In rare cases, the generation of samples from the corners of the hypercube may result in heavily distorted images, although they should still be considered non-anomalous, because of hypercube boundaries (see Figure 6.10).

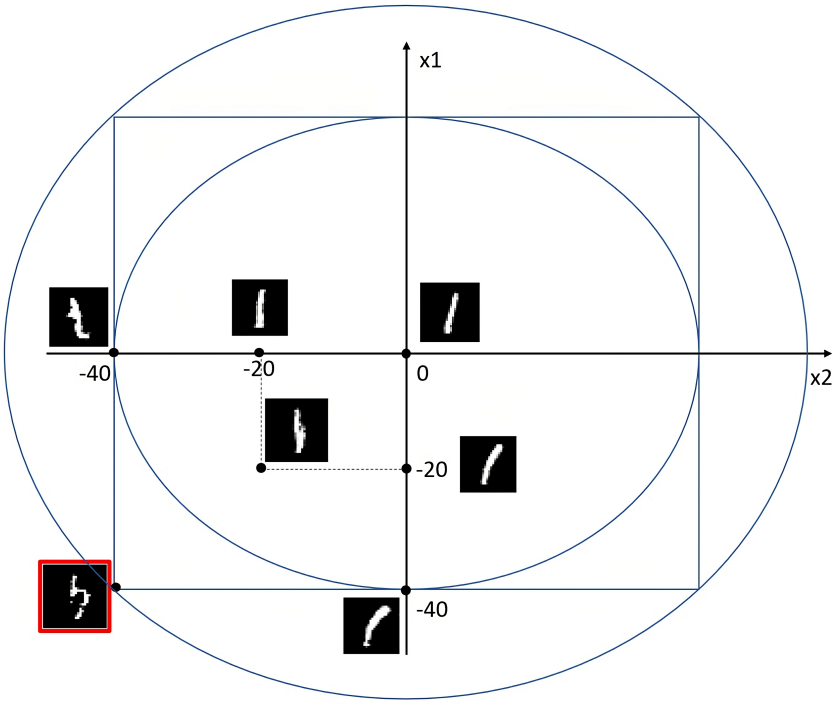


Figure 6.10: Simplified scenario of generating an anomalous image of class 1 still in the region of hypercube

Figure 6.10 represents the above-mentioned scenario where the combination of x_1 and x_2 results in an adversarial image that should be considered anomalous. Therefore, the hypersphere boundaries are better suitable for distance-based anomaly detection.

6.3.3 Hypersphere Distance-based Anomaly Detection

Given our previous assumption that the counterfeit samples bear a resemblance to the original images, and that we possess knowledge of the fabrication process used to create the forged images, it is clear that any anomalies could not have originated from the same process. Again, by training the Inverse Transformation model, which is the inverse function of the generator, we can detect these outliers that have been produced through different fabrication processes.

It is important to note that if the reconstructed responses for the training samples have their spheric coordinates within the unit hypersphere, any seeds obtained from anomalous samples must be located outside of it. With this aim, we train the inverse transformation model (see Chapter 5.2).

Metrics	Inverse Transformation
Detection Accuracy (%)	41.73
Reconstruction MSE metric	1.0859

Table 6.2: Evaluation of Inverse Transformation based on hypersphere distance

The anomaly detection experiments were conducted using the MNIST dataset. The latest findings of the inverse transformation training (refer to Table 6.2)) exhibit some advancements over the uniformity concept stated in Chapters 6.3.1 and 6.3.2. Nonetheless, the results lack reliability in detecting anomalies, and this may be attributed to the inadequacy of MSE loss as a sole loss function in this particular situation. A suitable substitute for the loss function could be the Chebyshev distance. Additionally, it is possible to implement the recursive reconstruction strategy to enhance the Inverse Transformation training. It could be done by the incorporation of a discriminator model acting as a critic based on the classification of images forged from reconstructed noise vectors generated by the Inverse transformation model.

Another problem that may arise when replicating this approach on more complex datasets. It could pose a challenge in training GAN, particularly in the aspect of hyperparameter tuning. To address this scalability concern, it is viable to modify the generative source by replacing the GAN's generator model with a diffusion model [188]. These models are more robust and exhibit superior performance in handling complex

data.

Diffusion models are constructed using basic image-denoising networks that minimize a convex regression loss (like L1). Unlike in the GAN setup, there is no minimax involved. Furthermore, GANs must generate a complete image in the forward pass, and in a way, they lack the capacity for image refinement during the generation process. In contrast, diffusion is a gradual and iterative approach that converts noise into an image incrementally. Therefore diffusion models have superior mechanisms for directing the image toward the desired outcome [189]. Similarly, we can adopt this process in our inverse transformation and iteratively reconstruct the input noise, which offers more control over the fabrication process, yielding potential benefits. To implement this concept, we take an iterative approach during image fabrication, which involves incorporating a U-net-like architecture between the generator model and inverse transformation (disassembler). This approach allows us to simulate this process to a certain extent within the architecture. Nonetheless, the domains of GAN and diffusion detection require more in-depth investigation thus, it is the area for further research.

6.4 Conclusion

The field of anomaly detection has been subjected to extensive research. Nevertheless, we do not consider the achieved results to be sufficient.

We conducted initial experiments in the field of anomaly detection using DGM, including the Autoencoder model and the GAN network. Our objective was to identify anomalous data samples in the data. In anomaly detection using AE, we evaluated the reconstruction error of the tested pattern as a metric for detection. The reconstruction error compared the input and reconstructed samples against each other, with the assumption that the original images would yield a smaller reconstruction error due to the AE's training on this data. Conversely, for GAN networks, we assumed that a well-trained generator would generate authentic fakes that the discriminator would classify similarly to the original data. Any data samples that were not available to the GAN network during training would be automatically evaluated by the discriminator as fakes, or anomalies. Our assumption was partially confirmed. However, the effectiveness of anomaly detection using standard DGM models, such as AE and GAN, depends on

the complexity of the original data and anomalies. The more complex the original data and the simpler the anomalous counterparts, the more difficult it is for DGM models to distinguish between these data.

For our next research, we have decided to focus on comparing the mechanisms of individual image data formation to detect anomalies by considering a significant difference in the noise used to create anomalous data from the characteristics of the original data as a metric. This approach is aimed at avoiding the problem of unstable anomaly detection that may arise due to data complexity. Our experiments consisted of two steps. First, we trained the GAN network on the training data, and upon successful training, we used the GAN network generator along with the Disassembler to reverse the decomposition of images into noise. We then compared the input and output noise based on Mean Squared Error (MSE) for training the Disassembler. We tested for anomaly patterns using statistical tests to determine whether the reconstructed noise consists of independent random variables with a uniform distribution. The experiments did not yield satisfactory results.

Lastly, during our experiments on relative entropy between uniform and normal distribution, we trained the GAN not only with normal distribution but also the uniform noise as an input. We observed poor convergence, and we could not achieve the same quality of results as with normal normalized noise. Therefore, we introduced a cascade (ladder) network architecture with two discriminators and two outputs for the generator model. Each discriminator evaluated one output of the generator, adding auxiliary information at a lower resolution, which facilitated model training. This modification led to better results, but they were still not comparable to those obtained with normal noise. Hence, we focused on training the GAN network with uniform noise to achieve effective training as with the normal normalized distribution. We verified the hypothesis that the variability of generated images is linked to the variability of generating noise. We used uniform noise with the same parameters as the normal distribution (with an interval corresponding to normalized normal distribution parameters - zero mean and unit standard deviation) and managed to influence the variability of the generated patterns by adjusting the mean value and width of the interval of the uniform distribution.

Chapter 7

Conclusion and Contribution

The main objective of this thesis was to contribute to the explainability of the deep learning systems domain. The non-linear recognition systems have demonstrated the ability to produce precise predictions, exceeding even human performance. Nonetheless, due to its intricate non-linear structure, the system is perceived as an inscrutable "black box" by the user, who is unable to gain knowledge from it. This manuscript contends that to facilitate user learning, an artificial recognition system must emulate the human cognitive approach to recognition.

This approach primarily explains the epistemological triangle, which deconstructs the recognition process into three components: acquisition of knowledge from data, model building, and evaluation of decisions. Thus, it is necessary to relinquish the primary advantage of machine learning, which involves building models directly from observed data. The architecture of these non-linear classifiers should be divided into a feature extractor and a classifier parts. The features represent the necessary data knowledge for building a robust model. However, explaining the features is a complicated process, much like mining knowledge from the observed data, and is essentially scientific in nature. While this process is reinforced by numerous scientific tools, it is necessary to develop those that streamline the interpretation of features. The proposed methods in this work detail the conception of an experimental tool intended for this very purpose. The process of inferring decisions from actual data, based on acquired knowledge, is a rigorously logical procedure. This is due to the fact that the current prevalent tool for constructing models, the digital computer, is founded upon the principles of two-valued logic. Therefore, even an explainable classifier must be able to form decisions in the

form of a logical expression. However, since we recognize diverse images in a single class, we used fuzzy logic for classification. In the result, the user learns which features are relevant (represented by a tertiary codeword) and to what extent they contribute to the degree of truthfulness of the classification.

In summary, post-hoc explainability enables users to acquire knowledge from non-linear recognizers and extract novel insights from the features uncovered by neural networks through deep learning. The concept of this work is divided into explainability of classification and interpretability of features, because mining of input data may result in new features that lack direct interpretability within the relevant application domain.

The supplementary objective of the thesis was a contribution to the anomaly detection domain. Based on an evaluation of the current state of the field, we have determined that the explainability mechanism fails to consider the possibility of inputs that deviate significantly from the training dataset. Consequently, the explanation methods may attempt to provide explanations for the outliers, without informing users of the incorrect usage of recognizers, which is a classification of anomaly. To address this issue and protect users, we decide to incorporate an anomaly detector as an integral component of the explainability system.

In the last concept, we changed the training approach for the proposed inverse transformation - the Disassembler, instead of the uniformity test we used the concept of unit hypersphere where the original data lies inside the sphere while the outliers are further away from the center - outside of it. With the use of a generator model from the GAN network we generate anomalies or authentic fakes in a controllable fashion - by sampling directly from the region of the hypersphere or outside of it. Then the Disassembler model was trained to reconstruct the created images as close as possible to their original counterparts (input noise vectors). By training the inverse transformation learns the underlying concept of distance between different types of images. Thus, during the test phase, the original images and authentic fakes should be reconstructed near the center of the hypersphere and vice versa.

Furthermore, we encountered difficulties when training GAN networks due to their unstable nature. The experiments on anomaly detection were performed on a simple MNIST dataset. Replicating this concept on more complex datasets like ImageNet

could be a problem in terms of training GAN on such datasets, more specifically in terms of hyperparameter tuning. To resolve this scalability issue, it is possible to change the generative source, i.e. replace the generator model from GAN with a diffusion model. These models are more stable and have better performance on complex data. In summary, **the main findings of the thesis** are:

1. Explainable decision-making must follow the human mode of decision-making as given by the epistemological triangle.

2. According to point 1, the recognition system must be divided into feature extraction and classification parts.

3. The explanation of extracted features is a process of exploration in which people acquire knowledge about features in the same way that science acquires knowledge about nature. This process must be supported by appropriate tools.

4. The explanation of the classification must be based on logic (binary or fuzzy).

5. Zadeh's fuzzy logic with features dividing into positive relevant, negative relevant, and irrelevant values gives a proper frame for the explanation of the obtained classification.

6. Features 'fuzzification'(transforming) affects the fitting of explainable classes with black-box predictions. Applying feature relevancies as a basis for nonlinearly transforming feature values into their truth values, the DecovnNet approach achieved 100% fidelity on the ImageNet database.

7. The first experiments indicate that the classifier in the form of fuzzy logic function can outperform the decision tree models.

8. The anomaly detector as a filter to avoid anomalies explanation is needed. Hawkins's definition [84] of the outlier as a sample with a different way of generation, provides guidance on applying the DGM for the outlier detection. We applied this idea in two ways:

    - If the generator seeds are independent pseudo-random variables with uniform distribution, the inverse transform of the tested image must give the same property for its outputs. The first results of this idea verification are not satisfactory. The reason seems to be the generation of seeds from a hypercube.
    - If the generator seeds are pseudo-random variables from the hypersphere with uniform distribution of the radius, the inverse transformation of the tested image must give the coordinates within the hypersphere. Due to the complexity of the inverse transformation training, this approach was left for further research.

Bibliography

[1] A.M. Turing. Computing machinery and intelligence. *Mind*, pages 1–20, 1950. doi: Mind59,433-460. https://www.cs.princeton.edu/~chazelle/courses/BIB/turing-intelligence.pdf.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, 2016. http://aima.cs.berkeley.edu/.

[3] D. A. Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1:1–1:15, 2012. doi: 10.1147/JRD.2012.2184356.

[4] Andre Vellino. Artificial intelligence: The very idea: J. haugeland, (mit press, cambridge, ma, 1985); 287 pp. *Artificial Intelligence*, 29:349–353, 09 1986.

[5] Andriy Burkov. *The hundred page machine learning book*. 2019. http://themlbook.com/.

[6] David James. *Introduction to Machine Learning with Python: A Guide for Beginners in Data Science*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 1st edition, 2018. ISBN 1726230872.

[7] Kunihiko Fukushima.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[10] Osvaldo Simeone. A very brief introduction to machine learning with applications to communication systems. *CoRR*, abs/1808.02342, 2018. URL `http://arxiv.org/abs/1808.02342`.

[11] Qiong Liu and Ying Wu. Supervised learning. 01 2012. doi: 10.1007/978-1-4419-1428-6_451.

[12] Peter Wittek. 5 - unsupervised learning. In Peter Wittek, editor, *Quantum Machine Learning*, pages 57–62. Academic Press, Boston, 2014. ISBN 978-0-12-800953-6. doi: https://doi.org/10.1016/B978-0-12-800953-6.00005-0. URL `https://www.sciencedirect.com/science/article/pii/B9780128009536000050`.

[13] Y Reddy, Viswanath Pulabaigari, and Eswara B. Semi-supervised learning: a brief review. *International Journal of Engineering Technology*, 7:81, 02 2018. doi: 10.14419/ijet.v7i1.8.9977.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

[15] Norbert Wiener. *Cybernetics ; or, Control and communication in the animal and the machine*. MIT Press, Cambridge, [second edition, 2019 reissue]. edition, 2019. ISBN 0-262-35591-4.

[16] K.-O. APEL. Charles s. peirce: From pragmatism to pragmaticism. 1981.

[17] Arash Shaban-Nejad, Martin Michalowski, and David L. Buckeridge. *Explainability and Interpretability: Keys to Deep Medicine*, pages 1–10. Studies in Computational Intelligence. Springer Science and Business Media Deutschland GmbH, Germany, November 2020. ISBN 9783030533519. doi: 10.1007/978-3-030-53352-6_1. Publisher Copyright: © 2021, The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG.; AAAI International

Workshop on Health Intelligence, W3PHIAI 2020 ; Conference date: 07-02-2020 Through 07-02-2020.

[18] Hussein Abdel-Jaber, Disha Devassy, Azhar Salam, Lamya Hidaytallah, and Malak EL-Amir. A review of deep learning algorithms and their applications in healthcare. *Algorithms*, 15:71, 02 2022. doi: 10.3390/a15020071.

[19] David Gunning and David Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, Jun. 2019. doi: 10.1609/aimag.v40i2.2850. URL `https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2850`.

[20] Michael Ridley. Explainable artificial intelligence (xai). *Information Technology and Libraries*, 41(2), Jun. 2022. doi: 10.6017/ital.v41i2.14683. URL `https://ejournals.bc.edu/index.php/ital/article/view/14683`.

[21] Wojciech Samek and Klaus Robert Müller. *Towards Explainable Artificial Intelligence*, pages 5–22. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, 2019. doi: 10.1007/978-3-030-28954-6_1. Funding Information: This work was supported by the German Ministry for Education and Research as Berlin Big Data Centre (01IS14013A), Berlin Center for Machine Learning (01IS18037I) and TraMeExCo (01IS18056A). Partial funding by DFG is acknowledged (EXC 2046/1, project-ID: 390685689). This work was also supported by the Institute for Information Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (No. 2017-0-00451, No. 2017-0-01779). Publisher Copyright: © Springer Nature Switzerland AG 2019.

[22] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022. URL `https://christophm.github.io/interpretable-ml-book`.

[23] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), aug 2018. ISSN 0360-0300. doi: 10.1145/3236009. URL `https://doi.org/10.1145/3236009`.

[24] Sidharth Mishra, Uttam Sarkar, Subhash Taraphder, Sanjoy Datta, Devi Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. Principal component analysis. *International Journal of Livestock Research*, page 1, 01 2017. doi: 10.5455/ijlr.20170415115235.

[25] Ivan Cimrák, Martin Klimo, Ondrej Šuch, and Katarína Bachratá. *Analýza procesov lineárnymi metódami*. 01 2012.

[26] Ofra Amir and Kobi Gal. Plan recognition and visualization in exploratory learning environments. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3:16–1, 2013. URL `https://www.dropbox.com/s/1goea6p2n7cnacn/tiisVLfinal%20%281%29.pdf?dl=0`.

[27] Amos Azaria, Zinovi Rabinovich, Claudia V. Goldman, and Sarit Kraus. Strategic information disclosure to people with multiple alternatives. *ACM Trans. Intell. Syst. Technol.*, 5(4), dec 2015. ISSN 2157-6904. doi: 10.1145/2558397. URL `https://doi.org/10.1145/2558397`.

[28] Or Biran and Courtenay V. Cotton. Explanation and justification in machine learning : A survey or. 2017.

[29] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *CoRR*, abs/1709.10256, 2017. URL `http://arxiv.org/abs/1709.10256`.

[30] Nicholas Jennings, Luc Moreau, David Nicholson, Sarvapali Ramchurn, Stephen Roberts, Tom Rodden, and Alex Rogers. Human-agent collectives. *Communications of the ACM*, 57:80–88, 11 2014. doi: 10.1145/2629559.

[31] Akiva Kleinerman, Ariel Rosenfeld, and Sarit Kraus. Providing explanations for recommendations in reciprocal environments. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 22–30, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi: 10.1145/3240323.3240362. URL `https://doi.org/10.1145/3240323.3240362`.

[32] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable agency for intelligent autonomous systems. *Proceedings of the AAAI Conference on*

Artificial Intelligence, 31(2):4762–4763, Feb. 2017. doi: 10.1609/aaai.v31i2.19108. URL https://ojs.aaai.org/index.php/AAAI/article/view/19108.

[33] Maha Salem, Gabriella Lakatos, Farshid Amirabdollahian, and Kerstin Dautenhahn. Would you trust a (faulty) robot? effects of error, task type and personality on human-robot cooperation and trust. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, page 141–148, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450328838. doi: 10.1145/2696454.2696497. URL https://doi.org/10.1145/2696454.2696497.

[34] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing, and interpreting deep learning models, 2017.

[35] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017.

[36] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models, 2018.

[37] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning, 2019.

[38] Zachary C. Lipton. The mythos of model interpretability, 2017.

[39] Derek Doran, Sarah Schulz, and Tarek R. Besold. What does explainable AI really mean? A new conceptualization of perspectives. *CoRR*, abs/1710.00794, 2017. URL http://arxiv.org/abs/1710.00794.

[40] Avi Rosenfeld and Ariella Richardson. Explainability in human-agent systems, 2019.

[41] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *ArXiv.org*, pages 1–66, 2017. doi: arXiv:1706.07269.

[42] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. *Examples are not enough, learn to criticize! Criticism for Interpretability*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf.

[43] M. Robnik-Sikonja and Marko Bohanec. Perturbation-based explanations of prediction models. In *Human and Machine Learning*, 2018.

[44] Dr. Matt Turek. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency*, 2017.

[45] Sameer Singh Marco Tulio Ribeiro and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv.org*, 1, June 2016.

[46] Avanti Shrikumar and et al. Not just a black box: Learning important features through propagating activation differences. *arXiv.org*, 3, April 2017.

[47] Patrick Hall and Navdeep Gill. An introduction to machine learning interpretability. *O'Reilly Media*, 2, April 2019.

[48] Alejandro Barredo Arrieta and et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Science Direct*, 58:82–115, June 2020.

[49] Arun Rai. Explainable ai: from black box to glass box. *Journal of the Academy of Marketing Science*, 48:137–141, December 2019.

[50] Wei Fan et al. Feiyu Xu. Explainable ai: A brief survey on history, research areas, approaches and challenges. *Natural Language Processing and Chinese Computing*, pages 563–574, November 2019.

[51] Zhengping Cheand et al. Interpretable deep models for icu outcome predictions. *American Medical Informatics Association*, February 2017.

[52] Michal Kolarik, Martin Sarnovsky, Jan Paralic, and Frantisek Babic. Explainability of deep learning models in medical video analysis: a survey. *PeerJ Computer Science*, 9:e1253, 03 2023. doi: 10.7717/peerj-cs.1253.

[53] Ilya Sutskever Joan Bruna Dumitru Erhan Ian Goodfellow Rob Fergus Christian Szegedy, Wojciech Zaremba. Intriguing properties of neural networks. pages 1–10, 2013. doi: arXiv:1312.6199.

[54] Christian Szegedy Ian J. Goodfellow, Jonathon Shlens. Explaining and harnessing adversarial examples. pages 1–11, 2014. doi: arXiv:1412.6572.

[55] Sakurai Kouichi Jiawei Su, Danilo Vasconcellos Vargas. One pixel attack for fooling deep neural networks. pages 1–15, 2019. doi: arXiv:1710.08864.

[56] Aurko Roy Martín Abadi Justin Gilmer Tom B. Brown, Dandelion Mané. Adversarial patch. pages 1–6, 2017. doi: arXiv:1712.09665.

[57] Andrew Ilyas Kevin Kwok Anish Athalye, Logan Engstrom. Synthesizing robust adversarial examples. pages 1–19, 2017. doi: arXiv:1707.07397.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[59] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[60] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579, 2015. URL `http://arxiv.org/abs/1506.06579`.

[61] Alexander Mordvintsev Olah, Chris and Ludwig Schubert. Feature visualization. 2017. doi: --. URL `https://distill.pub/2017/feature-visualization/`.

[62] Arvind Satyanarayan Ian Johnson Shan Carter Ludwig Schubert Katherine Ye Olah, Chris and Alexander Mordvintsev. The building blocks of interpretability. 2018. doi: --. URL `https://distill.pub/2018/building-blocks/`.

[63] Google LLC. Negative neurons - feature visualization. pages 0–1, 2018. doi: --. URL https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/feature-visualization/negative_neurons.ipynb.

[64] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks, 2016.

[65] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug play generative networks: Conditional iterative generation of images in latent space, 2017.

[66] Aditya Khosla Aude Oliva Antonio Torralba David Bau, Bolei Zhou. Network dissection: Quantifying interpretability of deep visual representations. pages 1–9, 2017. doi: arXiv:1704.05796.

[67] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks, 2015.

[68] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[69] Andrew Zisserman Karen Simonyan, Andrea Vedaldi. Deep inside convolutional networks: Visualising image classification models and saliency maps. pages 1–8, 2013. doi: arXiv:1312.6034.

[70] Julius Adebayo Maximilian Alber Kristof T. Schütt Sven Dähne Dumitru Erhan Been Kim Pieter-Jan Kindermans, Sara Hooker. The (un)reliability of saliency methods. pages 1–12, 2017. doi: arXiv:1711.00867.

[71] James Zou Amirata Ghorbani, Abubakar Abid. Interpretation of neural networks is fragile. pages 1–21, 2017. doi: arXiv:1710.10547.

[72] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013. URL https://arxiv.org/abs/1311.2901.

[73] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net, 2014. URL https://arxiv.org/abs/1412.6806.

[74] Abhishek Das Ramakrishna Vedantam Devi Parikh Dhruv Batra Ramprasaath R. Selvaraju, Michael Cogswell. Grad-cam: Visual explanations from deep networks via gradient-based localization. pages 1–23, 2016. doi: arXiv:1610.02391.

[75] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017. URL http://arxiv.org/abs/1706.03825.

[76] Miguel Lerma and Mirtha Lucas. Grad-cam++ is equivalent to grad-cam with positive gradients, 2022.

[77] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. *CoRR*, abs/1710.11063, 2017. URL http://arxiv.org/abs/1710.11063.

[78] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7): 1–46, 07 2015. doi: 10.1371/journal.pone.0130140. URL https://doi.org/10.1371/journal.pone.0130140.

[79] Zhang Chunkai, Xinyu Wang, Jiahua Zhang, Shaocong Li, Hanyu Zhang, Chuanyi Liu, and Peiyi Han. Vesc: a new variational autoencoder based model for anomaly detection. *International Journal of Machine Learning and Cybernetics*, 14:1–14, 10 2022. doi: 10.1007/s13042-022-01657-w.

[80] Ilker Onat and A. Miri. An intrusion detection system for wireless sensor networks. volume 3, pages 253 – 259 Vol. 3, 09 2005. ISBN 0-7803-9181-0. doi: 10.1109/WIMOB.2005.1512911.

[81] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection: A realistic modeling and a novel

learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–14, 09 2017. doi: 10.1109/TNNLS.2017.2736643.

[82] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017. URL http://arxiv.org/abs/1703.05921.

[83] F.Y. Edgeworth M.A. Xli. on discordant observations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 23(143):364–375, 1887. doi: 10.1080/14786448708628471. URL https://doi.org/10.1080/14786448708628471.

[84] D. M. Hawkins. *Identification of outliers*. Monographs on applied probability and statistics. Chapman and Hall, London [u.a.], 1980. ISBN 041221900X. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+02435757X&sourceid=fbw_bibsonomy.

[85] Lukas Ruff, Jacob Kauffmann, Robert Vandermeulen, Gregoire Montavon, Wojciech Samek, Marius Kloft, Thomas Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, PP:1–40, 02 2021. doi: 10.1109/JPROC.2021.3052449.

[86] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41, 07 2009. doi: 10.1145/1541880.1541882.

[87] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 2019.

[88] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 01 2019.

[89] Jaroslav Kopčan, Ondrej Škvarek, and Martin Klimo. Anomaly detection using autoencoders and deep convolution generative adversarial networks. *Transportation Research Procedia*, 55:1296–1303, 2021. ISSN 2352-1465. doi: https://doi.org/10.1016/j.trpro.2021.07.113. URL https://www.sciencedirect.com/

science/article/pii/S2352146521005287. 14th International scientific conference on sustainable, modern and safe transport.

[90] Lukas Ruff, Robert Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep support vector data description for unsupervised and semi-supervised anomaly detection. 06 2019.

[91] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/ruff18a.html.

[92] Ondrej Škvarek, Martin Klimo, and Jaroslav Kopčan. Pca tail as the anomaly indicator. In *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 613–619, 2020. doi: 10.1109/ICETA51985.2020.9379267.

[93] Ma Yao and Huangang Wang. One-class support vector machine for functional data novelty detection. In *2012 Third Global Congress on Intelligent Systems*, pages 172–175, 2012. doi: 10.1109/GCIS.2012.19.

[94] Shuhan Yuan and Xintao Wu. Trustworthy anomaly detection: A survey, 2022.

[95] Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2009.

[96] Eitan Richardson and Yair Weiss. On gans and gmms, 2018.

[97] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

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

[99] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

[100] Fei Tony Liu, Kai Ting, and Zhi-Hua Zhou. Isolation forest. pages 413 – 422, 01 2009. doi: 10.1109/ICDM.2008.17.

[101] Markus Breunig, Peer Kröger, Raymond Ng, and Joerg Sander. Lof: Identifying density-based local outliers. volume 29, pages 93–104, 06 2000. doi: 10.1145/342009.335388.

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

[103] Sreekanth Vempati, Andrea Vedaldi, Andrew Zisserman, and C. Jawahar. Generalized rbf feature maps for efficient detection. pages 1–11, 01 2010. doi: 10.5244/C.24.2.

[104] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[105] Guiqin Yuan, Bo Li, Yiyang Yao, and Simin Zhang. A deep learning enabled subspace spectral ensemble clustering approach for web anomaly detection. pages 3896–3903, 05 2017. doi: 10.1109/IJCNN.2017.7966347.

[106] Gyuwan Kim, Hayoon Yi, Jangho Lee, Yunheung Paek, and Sungroh Yoon. Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems, 2016.

[107] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Surya Deepak Turaga. *Outlier Detection with Autoencoder Ensembles*, pages 90–98. 06 2017. ISBN 978-1-61197-497-3. doi: 10.1137/1.9781611974973.11.

[108] Francois Bourdonnaye, Celine Teuliere, Thierry Chateau, and Jochen Triesch. Learning of binocular fixations using anomaly detection with deep reinforcement learning. pages 760–767, 05 2017. doi: 10.1109/IJCNN.2017.7965928.

[109] Chengqiang Huang, Yulei Wu, Yuan Zuo, Ke Pei, and Geyong Min. Towards experienced anomaly detector through reinforcement learning. *Proceedings of the AAAI*

Conference on Artificial Intelligence, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.12130. URL https://ojs.aaai.org/index.php/AAAI/article/view/12130.

[110] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013. ISBN 978-1-4614-6396-2. URL http://dx.doi.org/10.1007/978-1-4614-6396-2.

[111] Longbing Cao Anton van den Hengel Guansong Pang, Chunhua Shen. Deep learning for anomaly detection: A review. 2020. doi: arXiv:2007.02500.

[112] Billy Joe Franks Klaus-Robert Müller Marius Kloft Lukas Ruff, Robert A. Vandermeulen. Rethinking assumptions in deep anomaly detection. 2020. doi: arXiv:2006.00339.

[113] Sanjay Chawla Raghavendra Chalapathy. Deep learning for anomaly detection: A survey. 2019. doi: arXiv:1901.03407.

[114] Nada Sahlab Michael Weyrich Benjamin Lindemann, Benjamin Maschler. A survey on anomaly detection for technical systems using lstm networks. 2021. doi: arXiv:2105.13810.

[115] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. 11 2015.

[116] Dan Li, Dacheng Chen, Jonathan Goh, and See kiong Ng. Anomaly detection with generative adversarial networks for multivariate time series, 2019.

[117] Lucas Deecke, Robert Vandermeulen, Lukas Ruff, Stephan Mandt, and Marius Kloft. Image anomaly detection with generative adversarial networks. 09 2018.

[118] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, 2017.

[119] Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo Regazzoni, and Nicu Sebe. Abnormal event detection in videos using generative adversarial nets, 2017.

[120] Sertac Arisoy, N.M. Nasrabadi, and Koray Kayabol. Gan-based hyperspectral anomaly detection. pages 1891–1895, 01 2021. doi: 10.23919/Eusipco47968.2020.9287675.

[121] Vít Škvára, Tomáš Pevný, and Václav Šmídl. Are generative deep models for novelty detection truly better?, 2018.

[122] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[123] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[124] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski, editors, *Artificial Neural Networks and Machine Learning – ICANN 2011*, pages 52–59, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21735-7.

[125] Dipanwita Thakur, Suparna Biswas, Edmond Ho, and Samiran Chattopadhyay. Convae-lstm: Convolutional autoencoder long short-term memory network for smartphone-based human activity recognition. *IEEE Access*, PP:1–1, 01 2022. doi: 10.1109/ACCESS.2022.3140373.

[126] Raghavendra Chalapathy, Edward Toth, and Sanjay Chawla. Group anomaly detection using deep generative models, 2018.

[127] Andrea Asperti and Matteo Trentin. Balancing reconstruction error and kullback-leibler divergence in variational autoencoders, 2020. URL `https://arxiv.org/abs/2002.07514`.

[128] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[129] Lucas Deecke, Robert Vandermeulen, Lukas Ruff, Stephan Mandt, and Marius Kloft. Anomaly detection with generative adversarial networks, 2018. URL `https://openreview.net/forum?id=S1EfylZ0Z`.

[130] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection, 2021.

[131] Trevor Darrell Jeff Donahue, Philipp Krähenbühl. Adversarial feature learning. pages 1–18, 2016. doi: arXiv:1605.09782.

[132] Heinz von Foerster. Computing in the semantic domain. *Annals of the New York Academy of Sciences*, 184, 1971.

[133] Jean-Marie Klinkenberg Hébert Louis. Sign structures/signo - applied semiotics theories. URL `http://www.signosemio.com/klinkenberg/sign-structures.asp`.

[134] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436, 2015.

[135] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL `https://arxiv.org/abs/1312.6114`.

[136] Yang Zhi-Han. Training latent variable models with auto-encoding variational bayes: A tutorial, 2022. URL `https://arxiv.org/abs/2208.07818`.

[137] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. *CoRR*, abs/1702.08423, 2017. URL `http://arxiv.org/abs/1702.08423`.

[138] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[139] Zhizhong Huang, Shouzhen Chen, Junping Zhang, and Hongming Shan. PFA-GAN: progressive face aging with generative adversarial network. *CoRR*, abs/2012.03459, 2020. URL `https://arxiv.org/abs/2012.03459`.

[140] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 10 2017. doi: 10.1038/nature24270.

[141] Christopher J. Anders, Talmaj Marinc, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Analyzing imagenet with spectral relevance analysis: Towards imagenet un-hans'ed. *CoRR*, abs/1912.11425, 2019. URL `http://arxiv.org/abs/1912.11425`.

[142] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. *CoRR*, abs/1806.07538, 2018. URL `http://arxiv.org/abs/1806.07538`.

[143] Leander Thiele, Miles Cranmer, William Coulton, Shirley Ho, and David N Spergel. Predicting the thermal sunyaev–zel'dovich field using modular and equivariant set-based neural networks. *Machine Learning: Science and Technology*, 3(3): 035002, jul 2022. doi: 10.1088/2632-2153/ac78c2. URL `https://dx.doi.org/10.1088/2632-2153/ac78c2`.

[144] Sayantan Kumar, Sean C. Yu, Andrew P. Michelson, and Philip R. O. Payne. Self-explaining neural network with plausible explanations. *CoRR*, abs/2110.04598, 2021. URL `https://arxiv.org/abs/2110.04598`.

[145] Milad Moradi and Matthias Samwald. Post-hoc explanation of black-box classifiers using confident itemsets. *CoRR*, abs/2005.01992, 2020. URL `https://arxiv.org/abs/2005.01992`.

[146] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Interpretable & explorable approximations of black box models. *CoRR*, abs/1707.01154, 2017. URL `http://arxiv.org/abs/1707.01154`.

[147] Martin Klimo, Peter Lukáč, and Peter Tarábek. Deep neural networks classification via binary error-detecting output codes. *Applied Sciences*, 11(8), 2021. ISSN 2076-3417. doi: 10.3390/app11083563. URL `https://www.mdpi.com/2076-3417/11/8/3563`.

[148] Xianpeng Guo, Biao Hou, Zitong Wu, Bo Ren, Shuang Wang, and Licheng Jiao. Prob-pos: A framework for improving visual explanations from convolutional neural networks for remote sensing image classification. *Remote Sensing*, 14(13), 2022. ISSN 2072-4292. doi: 10.3390/rs14133042. URL https://www.mdpi.com/2072-4292/14/13/3042.

[149] Dumitru Erhan, Y. Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*, 01 2009.

[150] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL https://arxiv.org/abs/1409.1556.

[151] Christian Tomani and Daniel Cremers. Challenger: Training with attribution maps, 2022. URL https://arxiv.org/abs/2205.15094.

[152] Hani Hagras. Toward human-understandable, explainable ai. *Computer*, 51:28–36, 2018.

[153] Junkang An and Inwhee Joe. Attention map-guided visual explanations for deep neural networks. *Applied Sciences*, 12(8), 2022. ISSN 2076-3417. doi: 10.3390/app12083846. URL https://www.mdpi.com/2076-3417/12/8/3846.

[154] Isel Grau, Dipankar Sengupta, Maria M. Garcia Lorenzo, and Ann Nowé. Interpretable self-labeling semi-supervised classifier. In David W. Aha, Trevor Darrell, Patrick Doherty, and Daniele Magazzeni, editors, *Proceedings of the 2nd Workshop on Explainable Artificial Intelligence*, pages 52–57, July 2018.

[155] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[156] Explainable ai demos'. URL https://lrpserver.hhi.fraunhofer.de/handwriting-classification.

[157] Sahil Verma, John P. Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *CoRR*, abs/2010.10596, 2020. URL `https://arxiv.org/abs/2010.10596`.

[158] Y. Lecun. The mnist database of handwritten digits. 2010.

[159] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[160] M. Buzgo. Image classifier with explainable features. 2022.

[161] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J. Anders, and Klaus-Robert Müller. Toward interpretable machine learning: Transparent deep neural networks and beyond. *CoRR*, abs/2003.07631, 2020. URL `https://arxiv.org/abs/2003.07631`.

[162] Grégoire Montavon. Gradient-based vs. propagation-based explanations: An axiomatic comparison. In *Explainable AI*, 2019.

[163] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(65)90241-X. URL `https://www.sciencedirect.com/science/article/pii/S001999586590241X`.

[164] L.A. Zadeh. Soft computing, fuzzy logic and recognition technology. In *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228)*, volume 2, pages 1678–1679 vol.2, 1998. doi: 10.1109/FUZZY.1998.686373.

[165] Richard Wesley Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950.

[166] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[167] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer*

Vision and Pattern Recognition, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[168] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[169] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[170] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. ISSN 00401706. URL `http://www.jstor.org/stable/1267351`.

[171] Martin Klimo, Jaroslav Kopčan, and L'ubomír Králik. Explainability as a method for learning from computers. *IEEE Access*, 11:35853–35865, 2023. doi: 10.1109/ACCESS.2023.3265582.

[172] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[173] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2016.

[174] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

[175] Jason M. Klusowski. Analyzing cart, 2020.

[176] Lior Rokach and Oded Maimon. *Classification Trees*, pages 149–174. 07 2010. doi: 10.1007/978-0-387-09823-4_9.

[177] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1:14 – 23, 01 2011. doi: 10.1002/widm.8.

[178] Gavin Cawley and Nicola Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 07 2010.

[179] Damjan Krstajic, Ljubomir Buturovic, David Leahy, and Simon Thomas. Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of cheminformatics*, 6:10, 03 2014. doi: 10.1186/1758-2946-6-10.

[180] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 01 1989. doi: 10.1023/A:1022604100933.

[181] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948. URL `http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48`.

[182] Mario Berta, Kaushik P. Seshadreesan, and Mark M. Wilde. Rényi generalizations of the conditional quantum mutual information. *Journal of Mathematical Physics*, 56(2):022205, feb 2015. doi: 10.1063/1.4908102. URL `https://doi.org/10.1063%2F1.4908102`.

[183] Package modelica.math.distributions.truncatednormal. `https://doc.modelica.org/Modelica%203.2.3/Resources/helpMapleSim/Math/Distributions/TruncatedNormal/index.html`. Accessed: 2023-04.

[184] Jaroslav Kopčan, Martin Klimo, and Ondrej Škvarek. Do neural networks recognize patterns as well as students? In *2022 20th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 338–343, 2022. doi: 10.1109/ICETA57911.2022.9974725.

[185] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[186] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[187] Abhay Yadav, Sohil Shah, Zheng Xu, David Jacobs, and Tom Goldstein. Stabilizing adversarial nets with prediction methods. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Skj8Kag0Z`.

[188] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[189] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

List of Author's Publications

Conferences

AFD001 Explainable AI: a brief introduction / Jaroslav Kopčan - In: **MiST 2021** Mathematics in science and technologies: proceedings of the MIST conference 2021 / Bachratá Katarína, Bohiniková Alžbeta – 1. vyd. – [S.l.] (Slovensko) : [s.n.], 2021. – ISBN 9798748088183, s. 52-59 [online]
Kopčan Jaroslav(100%)

ADF001 PCA Tail as the Anomaly Indicator / Ondrej Škvarek, Martin Klimo, Jaroslav Kopčan. In: **ICETA 2020** [electronic]: 18th IEEE International conference on emerging elearning technologies and applications: Information and communication technologies in learning: proceedings. - 1. vyd. - Denver: Institute of Electrical and Electronics Engineers, 2020. - ISBN 978-0-7381-2366-0. - s. 613-619 [online]. SCOPUS
Kopčan Jaroslav(33%) Klimo Martin(33%) Škvarek Ondrej(33%)
There are **two existing citations** for this paper:

1. 2022 HAJTMANEK, R. et al. One-parameter statistical methods to recognize DDoS attacks. In: Symmetry. ISSN 2073-8994, 2022, vol. 14, iss. 11, s. 1-25. SCOPUS; WoS

2. 2022 ŠUCH, O., FABRICIUS, R., TARÁBEK, P. Introducing students to out-of-distribution detection with deep neural networks. In: ICETA 2022. ISBN 979-8-3503-2032-9, s. 627-633. SCOPUS

ADF002 Anomaly detection using Autoencoders and Deep Convolution Generative Adversarial Networks / Jaroslav Kopčan, Ondrej Škvarek, Martin Klimo. In: **TRANSCOM 2021** 14th International scientific conference on sustainable, modern and safe transport [electronic]. - ISSN 2352-1465 (online). - 1. vyd. - Amsterdam: Elsevier, 2021. - s. 1296-1303 [online]. SCOPUS
Kopčan Jaroslav(33%) Klimo Martin(33%) Škvarek Ondrej(33%) There are **two existing citations** for this paper:

1. 2022 HAJTMANEK, R. et al. One-parameter statistical methods to recognize DDoS attacks. In: Symmetry. ISSN 2073-8994, 2022, vol. 14, iss. 11, s. 1-25. SCOPUS; WoS

2. 2022 ŠUCH, O., FABRICIUS, R., TARÁBEK, P. Introducing students to out-of-distribution detection with deep neural networks. In: ICETA 2022. ISBN 979-8-3503-2032-9, s. 627-633. SCOPUS

V2 Do neural networks recognize patterns as well as students? [electronic] / **Jaroslav Kopčan, Martin Klimo, Ondrej Škvarek.** In: **ICETA 2022** [electronic]: 20th Anniversary of IEEE International Conference on Emerging eLearning Technologies and Applications: proceedings. - 1. vyd. - Piscataway: Institute of Electrical and Electronics Engineers, 2022. - ISBN 979-8-3503-2032-9 (online). - s. 338-343 [online]. SCOPUS
[Kopčan Jaroslav (37.50%) - Klimo Martin (37.50%) - Škvarek Ondrej (25%)]

International Impacted Journals

V3 Explainability as a method for learning from computers [electronic] / **Martin Klimo, Jaroslav Kopčan, and Ľubomír Králik**. In: **IEEE Access** [electronic]: practical innovations, open solutions. - ISSN 2169-3536 (online). - Roč. 11 (2023), s. 35853-35865 [online]. Spôsob prístupu: https://ieeexplore.ieee.org/document/10097489
[Klimo Martin (37.50%) - Kopčan Jaroslav (37.50%) - Králik Ľubomír (25%)]