Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Sondre Engen & Piraveen Perinparajan

Extending Model-Induction Methods by Case-Based Reasoning for Explaining Deep Neural Networks

Master's thesis in Computer Science
Supervisor: Agnar Aamodt
June 2019

NTNU
Norwegian University of Science and Technology

Sondre Engen & Piraveen Perinparajan

Extending Model-Induction Methods by Case-Based Reasoning for Explaining Deep Neural Networks

Master's thesis in Computer Science
Supervisor: Agnar Aamodt
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Preface

This master thesis was written for the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU) as part of Norwegian Open AI lab's increased effort to create a bigger focus on explainability in AI systems.

This thesis assumes the reader has some degree of familiarity with AI and the concepts related to it. New techniques and concepts from state of the art literature will be reviewed and explained according to our interpretation.

We want to take this opportunity to express our gratitude to our supervisor, Agnar Aamodt, for his valuable guidance and feedback throughout this project. Also, our sincerest thanks to Kerstin Bach for helping us with myCBR and providing help troubleshooting the system and Amar Jaiswal for taking the time to discuss ideas related to CBR.

The thesis has a GitHub repository[1] with all the code for the implemented system.

Sondre Engen & Piraveen Perinparajan
Trondheim, June 2019.

[1] `https://github.com/piravp/XAI-CBR`

Abstract

Artificial Intelligence is in continuous development in an attempt to solve new and challenging problems. With the success of AI, the issue of explainability has become more apparent. Many of the algorithms used are considered opaque models unable to explain their decision to a user, including deep-neural networks are one of these.

In this thesis, we have taken a closer look at previous attempts to explain AI, both with and without, case-based reasoning. The use of model-induction methods have been at the center of our research. We have designed an architecture for a system that is capable of explaining a deep-neural network, by combining case-based reasoning and model-induction methods. To showcase the system, we implemented a small part of the designed system to showcase how it could be used.

Contents

List of Figures

List of Tables

Listings

Acronyms

AI Artificial Intelligence. 1

ANN Artificial Neural Network. 7

CBE Case-Based Explainer. 40

CBR Case-Based Reasoning. 4, 7, 81, 82

CEM Contrastive Explanations Method. 36

CRED Continuous/discrete Rule Extractor via Decision tree Induction. 35

DBN Deep Belief Network. 8

DeepLIFT Deep Learning Important FeaTures. 23

DeepRED Deep neural network Rule Extraction via Decision tree induction. 35

DL Deep Learning. 1

DNN Deep Neural Network. 2, 4, 7, 58, 82

GAM General Additive Models. 9, 31, 33

GDPR General Data Protection Regulation. 2

k-NN k-nearest neighbors. 33

LIME Local Interpretable Model-Agnostic Explanations. 21

LORE LOcal Rule-based Explanations. 35

LRP Layer-wise Relevance Propagation. 24

RBM Restricted Boltzmann Machine. 8, 30

RNN Recurrent Neural Network. 32

SENN Self-Explaining Neural Network. 31

SHAP SHapley Additive exPlanations. 25

SVM Support-Vector Machine. 33

XAI eXplainable Artificial Intelligence. 2, 4, 7, 81

Chapter 1

Introduction

In 1.1 we present a brief example on the power of AI. We continue by talking about the problem and motivation for this thesis in 1.2. Section 1.2 highlights one aspect of Artificial Intelligence (AI), namely explainability in AI, which has become increasingly important to consider as AI continues to rise in popularity. The next sections present the thesis goal, research questions, and research method. We wrap up this chapter by presenting the structure of the remaining thesis in section 1.8.

1.1 Introduction

In recent years, AI has seen increasing growth in popularity [47]. This is partly due to increased and cheaper computing power that has become available as well as the growing availability of big data. In particular, the interest in Deep Learning (DL) has exploded over the recent years. Its ability to solve problems in different domains has caught the attention of various industries. DL is being used in autonomous vehicles, detection of cancer cells, speech-recognition and finance to mention a few.

For a long time, the Chinese board game, Go, was considered impossible for machines to master because of its complexity [83]. However, in 2016, Google DeepMind's AlphaGo was able to defeat the worlds best Go player, Lee Sedol. This came as a surprise even within the AI community. In the aftermath of AlphaGo's deciding move (move 37), there was both surprise and confusion as to why it acted as it did [30].

1.2 Problem and Motivation

Trust

There is little doubt that DL has helped revolutionize some industries and has helped solve problems that were previously not solvable. However, it has come with a prize. A weakness of many of these systems is their inability to explain their advice and decisions to human users. Despite the head-scratching many scientists experienced after AlphaGo's deciding move, it didn't cause any real danger to human life. There are however other high-stake domains where the lack of explanation might. A medical doctor, for instance, may get the correct disease classification from an MRI image classifier, but might find it hard to trust the system as long as no justification or explanation of how it reached its conclusion can be given. The need for explainable AI systems has boosted a new interest in this area, also exemplified by the recent research program in the US by Defense Advanced Research Projects Agency (DARPA) targeting eXplainable Artificial Intelligence (XAI) [21].

Privacy

In April 2018, EUs General Data Protection Regulation (GDPR) came into force and started applying to any company processing EU citizens personal data [37]. Under Article 22, *"The data subject shall have the right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her"*. Where *profiling* is *"any form of automated processing of personal data consisting of the use of personal data to evaluate certain personal aspects relating to a natural person"* which, could also apply to any ML algorithm that make decisions from personal data. It was believed that this meant the users had the *right to explanation* - a right to be given an explanation for a decision made by an algorithm. However, a review by Mittelstadt et al. [66] highlighted how this is not the case. Nevertheless, this aspect is still important to consider as it should be expected that, although not mandatory by law yet, we can expect the right for an explanation to become legally mandated at some point in the near future with AI's increased usage. A system built with algorithms solely based on black boxes would have difficulties meeting this future requirement.

As AI systems continue to partially or fully replace humans, the task of opening the "black-box" model has become increasingly important. In order to enable an explanatory functionality, certain properties, knowledge, and underlying methods of a system will need an explicit representation at some level. A symbol-processing system could work jointly with a Deep Neural Network (DNN) to give the total system the prediction accuracy of the DNN while also enabling the desired explanation capability. The explanation knowledge may be of various types, including rules, deeper models or past cases. In addition to the benefits it would provide to the system developer in terms of developing and evaluating mistakes, it would also be a valuable step

towards providing trust in the system with respect to high stake decisions [93], and uncovering deeper meaning behind a DNN's decisions.

CBR as Explainer

The motivation to use CBR as the "engine" for our explanation system is to assess whether the advantages of CBR in problem solving can be transferred to generate useful explanations. Our hypothesis is that advantages such as utilizing previous experiences and the ability to incorporate domain knowledge and context can bring forth a new dimension to the system, thereby generating more useful explanations to the user. Based on our literature review from the specialization project, we did not find any evidence of others who have created an implementation or a proof of concept where CBR is used to explain a black-box model. Furthermore, CBR is regarded as better suited to explain the decisions of complex models, such as a DNN, as CBR bases its reasoning on previous experiences according to Nugent et al. [69].

1.3 Approaches

This thesis has a heavy focus on *model induction* techniques. Hence, it's important to establish the definition of model induction early on. Model induction can be explained as various techniques that experimentally infer from a black box model, such as a DNN, to explain the decision of the black box.

A *model agnostic* approach is when the model to generate explanations can handle any machine learning model regardless of the underlying architecture, due to its separation from the architecture itself. This is done by disregarding the internal and only looking at the input and output. The good thing about these approaches is that they can build on top of any black-box without the need to alter the underlying model performing the prediction, only expanding with explanatory abilities.

1.4 Goals and Research Questions

With this in mind, we have formulated goals and research questions which we seek to answer in this thesis. We have picked up from where we left our research with the specialization project [27] when doing further research for this thesis. However, the research conducted for the master thesis has been narrowed down in order to focus on only what is required to design a more detailed architecture and creating a proof of concept.

Goal: We want to extract information from a DNN using model induction techniques extended by case-based reasoning to derive an explanation.

The extracted information should be passed through the case-based explainer which processes the information with previous information to create knowledge used to generate an explanation to the user. We hope by combining local interpretation[1] techniques along with a case-based reasoning framework, we should be able to create a global interpretation[2] of a DNN model, and use said interpretation to generate explanations to the user.

Ultimately, we want to end up with a proof of concept for a system that has the capability to create a convincing explanation using the described architecture. We will base the design of our proof of concept on previous research from our specialization project and further research performed during our current work for the master thesis.

Research Question 1: What is the current status of eXplainable Artificial Intelligence (XAI) and eXplainable Artificial Intelligence (XAI) related to Case-Based Reasoning (CBR)?

In addition to the research that we did in the specialization project, we think there is more research to be done in order to create a more detailed architecture. Thus, we will continue our work with the research on XAI and XAI related to CBR.

Research Question 2: How can model-agnostic methods be combined with Case-Based Reasoning (CBR) to gain knowledge of an underlying Deep Neural Network (DNN)?

We want to research whether model-agnostic methods can be combined with CBR to gain knowledge of an underlying DNN. This includes a method which is applicable to most of the popular DNN-architectures. We consider this a desirable property to not limit which datasets the explainer-module is applicable to.

Research Question 3: Can our proposed system provide a step towards achieving user-understandable explanations?

We want to see if there is a way to design a system which can generate user-understandable explanations.

Although it may be possible to design such a system, it is worth noting that this is a proof of concept. Hence, a full implementation is unlikely given the limited amount of time.

[1] A simplified model that only approximates decisions made about a few data points, typically only one.
[2] A simplified model that approximates decisions made for "all" possible data points.

1.5 Research Method

The research method will consist of design science and then creating a running example. In practice, both of these methods are connected to the research questions, respectively. The first part, design science, consists of developing an architecture to show how a DNN extended by model induction methods and the CBR framework can work together to explain outputs from a DNN. The second part, consist of testing how well the system performs on an example case.

1.6 Overall Process

As this thesis was a continuation of our previous work in the specialization project, it was clear that we still lacked relevant projects to design a system based on, and as such, a majority of our time was used to do further research in the field. The majority of the first months were spent finding more research. As the field is in continuous development, new papers were discovered regularly.

From this research, we spent a fair amount of time planning our architecture and expanding the previous architecture from the specialization project. As it is impossible to verify a good architecture at first glance, and very difficult to draft the final details, we left the architecture as general as possible, with the expectation to include final details from our implementation later on in the project.

The development process started in early April. We spent a good amount of time planning the process, which would ultimately help in the implementation. As we were two student working on this theses, we split the development task into two parts. Figuring out the CBR framework and how to use it, and developing the black-box (neural network) on a pre-selected dataset with corresponding explanatory libraries. When both of these parts were finished, it was simply a matter of connecting the two.

In terms of developing our architecture, we had to spend some time figuring out how to apply the libraries we chose to use. We initially chose to use myCBR for the CBR implementation. This turned out to be somewhat of a challenge as there were some problems with working on a different OS than the developers, and some features which were lacking in the original implementation of the REST API. These problems were fixed by the developers, and the various features which were needed on our side, were simple to implement with the help of Kerstin Bach.

The XAI libraries that were of interest, which we based our explanation engine on, had a good amount of open code associated with them, however utilizing these code bases was very difficult and time consuming, with some of the specifics functions used being deprecated and had to be replaced, and the code itself was usually not user friendly with no comments.

The second part of the implementation was the testing phase, where we would use our architecture on a particular problem to see what explanation produces. The quality of the explanation was difficult to quantify, and couldn't really be measured directly.

1.7 Thesis Structure

In the remainder of the thesis the structure is set up in the following manner:

- **Chapter 2 - Background Theory**: Core concepts the thesis is based on.
- **Chapter 3 - Background Research**: A summary of relevant research from the specialization project with regard to the state of the art in XAI.
- **Chapter 4 - Architecture**: A deep dive into the final architecture that was designed and discussions related to this.
- **Chapter 5 - Implementation**: We present our implementation and show a running example.
- **Chapter 6 - Discussion**: General discussion.
- **Chapter 7 - Conclusion and Future work**: Conclusion and future work.

Chapter 2

Background Theory

This chapter will provide the reader with background theory needed to read this thesis. It will cover topics such as Artificial Neural Network (ANN), eXplainable Artificial Intelligence (XAI), Case-Based Reasoning (CBR) and knowledge containers. These topics are covered from 2.1 to 2.3. For our discussion related to XAI, we make a distinction on terms that are often used interchangeably in XAI research in 2.4. We conclude this chapter by including some relevant background theory from the specialization project in 2.8. This chapter lays the groundwork for discussions further in the thesis.

2.1 Aritifical Neural Networks and Deep Neural Networks

Artificial Neural Network (ANN) is a field in computer science which is loosely modeled after the human brain. It works by combining individual neurons to form what is known as a layer. Any network needs to have at least an input and an output layer. The input layer takes in a given number of inputs and outputs a prediction, e.g. whether the system detected a cat or a dog. What differentiates ANN from Deep Neural Network (DNN) is the number of hidden layers. Any additional layer between the input and output layer is known as a hidden layer. More than one hidden layer constitutes a deep neural network. ANN has been successfully applied to many different domains such as image recognition, speech recognition, music generation, recommender systems and more. ANN takes in instances of inputs which it then tries to create internal patterns of. Typically, as the training converges, the system manages to capture a higher degree of abstraction for each layer. For instance, when dealing with images, the layers may be broken down into these level of abstractions: individual pixels, edges, objects, and finally complete image.

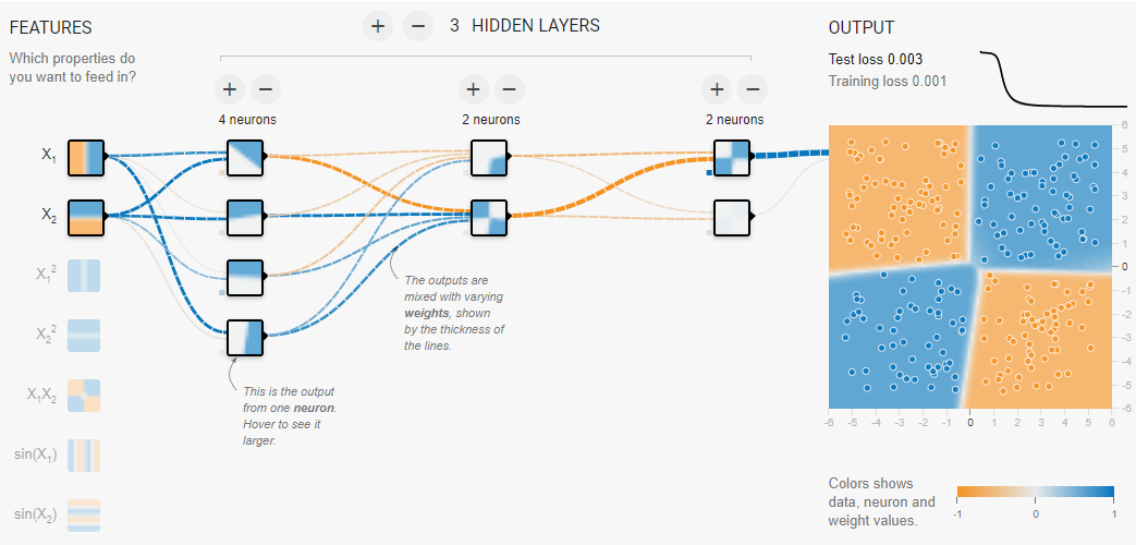


FIGURE 2.1: Visualization of the internal patterns forming inside a neural network. The greater the edge, the greater the contribution to the final prediction. [89]

As illustrated in figure 2.1, the higher level of abstraction unfolds between each layer. Between the input and last hidden layer, it has managed to figure out a pattern in the data.

2.1.1 Deep Belief Networks

A Restricted Boltzmann Machine (RBM) is a shallow two layer net, with no intra-connected layers. RBM can be combined, forming a type of neural network called Deep Belief Network (DBN) [45], where each layer in the Deep Belief Network (DBN) contain different features. The goal of a RBM is to reconstruct the input with high-level concepts learned in an unsupervised fashion, one layer at a time, where the input of one layer is the target of the next, similar to an auto-encoder. Once all layers are trained, they can be taught to classify by fine-tuning on supervised data through the whole network using previously learned internal features.

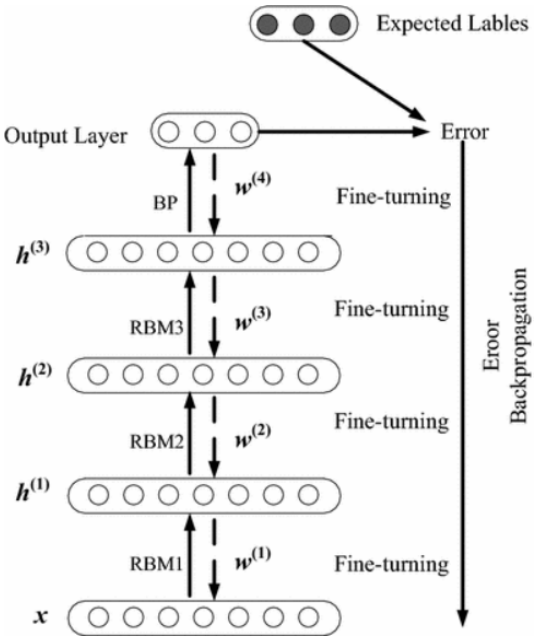


FIGURE 2.2: Inner structure of a Deep Belief Network, with internal RBM layers with one-layer back-propagation at the end. x is the input layer. It is trained using a combination of unsupervised- and supervised learning [97].

2.1.2 General Additive Models

General Additive Models (GAM) are viewed as a powerful interpretable model in machine learning and statistics [42]. The purpose of GAMs are to maximize the quality of prediction on a dependent variable Y from various distributions by estimating non-parametric functions of the predictor variables (x) that are connected to Y via some link function. The model relates to any univariate (function of only one variable) response variable Y, to some predictor variable x_i, with E being an exponential family distribution (e.g. normal, binomial or Poisson) along with a link function g. f_i can be any arbitrary non-linear function [92].

$$g(E(Y)) = \sum f_i(x_i) \tag{2.1}$$

2.2 Case-Based Reasoning

Case-Based Reasoning is the process of solving new problems based on the solution of similar past cases, where each case is an experience. The principle is broadly based on how humans solve problems: solving new problems with past experiences in similar situations. It commonly consists of a 4-step process as described by Aamodt et al. [2]:

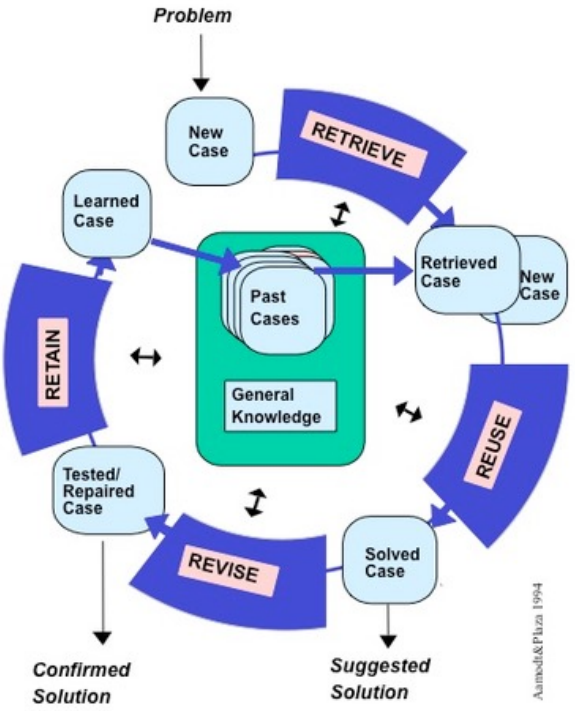


FIGURE 2.3: CBR Cycle [2], with a overview of how a problem is solved. A case is generated from a problem (problem characterization) and the steps are completed in sequence to solve the problem.

- **Retrieve**: Given a target problem, in the form of a case consisting of a problem and its corresponding attribute, the system searches among previously solved cases to find similar cases with a solution.
- **Reuse**: Map the solution from previous cases to the new problem. Adapting their solutions to generate a new case that fits the new problem.
- **Revise**: Verify if the generated solution solved the target problem by applying the solution. If the solution didn't solve the target problem, revise if possible, or evaluate by other means.
- **Retain**: Anything that needs to be stored is stored in this step. This could either be a complete case *with* a solution or a case which contains some status that it is waiting for solution to be added at a later time.

In the context of XAI, the explanation is regarded as the solution in CBR [55], sometimes referred to as CBE (Case-Based Explanation).

2.2.1 Knowledge Containers in CBR

Within the CBR cycle, we have 4 knowledge containers which interplay to perform the various steps in the CBR cycle. These are commonly know as:

- **Vocabulary**: How a case is represented, the terminology used to represent information/-data. Every attribute definition, the possible values for each attribute, the attribute weights, etc. We also have several sub-containers within: Retrieval Attributes, Input Attributes and Output Attributes.
- **Similarity Measure**: Holds all the knowledge that is needed to compute the similarity between cases. This includes measures to compute the similarity between whole cases, but also between single attributes.
- **Case-Base**: Contains the experiences in the form of cases. These are usually stored in pairs of a problem description and a solution, or only the problem description if unsolved.
- **Solution Transformation**: Also called the adaptation container. This container is responsible for adapting previous cases' solution to fit a new problem.

2.2.2 Knowledge, Information and Data

These are terms that are used frequently in this thesis. A definition is needed to get a clear distinction between them as they are often used together. We embrace the definition given by Aamodt [1]: *Data* is syntactical entities, i.e. patterns with no meaning, *information* is data with meaning, i.e. interpreted data, while *knowledge* is information that is learned, i.e. information that has been processes and incorporated into the system. The role of knowledge is to enable the system to interpret data to information to elaborate and derive new information to learn more.

In terms of generating explanations to a system, knowledge is at the core of the reasoning process. However, it remains a challenge to obtain and use this knowledge to explain a solution with regards to the domain knowledge of the system.

2.3 Knowledge-Intensive CBR

A CBR system is a lazy-learner. It doesn't generalize in the learning-phase, but simply stores the cases in its memory and utilizes the 4-step process to solve new problems at run-time. This

approach, however, is severely limited with respect to the amount of knowledge the system contains. Each case usually only contains *data*, which will make it difficult to create any explanation or reasoning beyond what's already encapsulated in it. A user might draw some information from looking at a specific case, but again, an explanation would be difficult to draw without knowledge related to this information.

FIGURE 2.4: Knowledge-Intensive Dimension w.r.t. CREEK and a Knowledge-Poor System - Knowledge vs Data [1]

The knowledge-intensive dimension in figure 2.4 specifies how a knowledge-intensive system with domain knowledge compares to a regular CBR system which only stores cases. We have multiple features along the dimension. The most important one related to explanations, however, is the similarity metric. If we are able to measure the similarity of new problems compared to previous problems, we should be more adapt to use our previous experience and knowledge.

2.4 Explainability vs. Interpretability vs. Transparency

In existing research, the terms *explainability* and *interpretability* are often used interchangeably. In fact, depending on the scientific community addressing these terms, there are disagreements on how to define each of these terms in and of themselves [39]. More specifically, according to Lipton [58], there is no clear agreement as to what interpretability actually means among the researches who use these terms. In this section, we want to make the reader aware that there is a distinction here worth noting for further discussions.

Tomsett et al. [90] define explainability as to which extent the system can provide clarification for its outputs. They define transparency as the level to which a system can provide information about its internal working and the data it has been trained on. While interpretability is the level to which an agent (meaning the user) gains, and can make use of both the information embedded within an explanation given by the system and the information provided by the system's transparency level. This gives us these three definition, that we use as our interpretation:

Explainability - Explainability can be understood as the justification outputted by the system with regards to a decision or belief.

Interpretability - Interpretability has no formal definition which is agreed upon. It's highly subjective, but for our purpose, we define it as the systems ability to present its reasoning in understandable terms to the user.

Transparency - While interpretability is to what degree we can gain or understand information from the model with respect to the user, transparency only refers to the ability to inspect the underlying model workings.

2.5 Approaches to XAI

DARPA presents three categories for explainable AI [21]. These are somewhat outdated, and as such, we combine these categories with the sub-categories presented by Molnar [67].

2.5.1 DARPA's Categories

We start by presenting DARPA's three categories. **Interpretable Models** consist of using machine-learning techniques that learn more structured, interpretable or causal models. This implies model transparency [24]. The explanation can be obtained from the model itself using decision trees, bayesian rule lists and so on.

Model induction is a technique that experimentally infer from a black-box model to approximate an explainable model to explain the decisions of the black-box.

Deep Explanation is a modified or hybrid deep learning approach that learns more explainable features, explainable representations or explainable generation facilities.

2.5.2 Molnar's Categories

Model-Specific vs. Model-Agnostic
Model-agnostic refers to the ability to explain any model, regardless of its inner workings. These tools usually work by examining the input-output pairs from the model. By definition, these do not have access to the model internals, such as weights or structural information.

Model-specific are interpretable tools limited to specific model classes. For instance, they may only work for neural networks, or regression weights in a linear model. Gradient based methods belong to this category. By definition the interpretation of intrinsically interpretable models are always model-specific.

Intrinsic vs. Post-Hoc
Intrinsic models are models that are inherently interpretable. In essence, this means there are

no post-processing required to make them interpretable. Despite being inherently interpretable, it is possible to apply post-hoc methods to decide feature importance.

Post-Hoc is any method where the model is made interpretable *after* training. Post-hoc methods are required for models which are considered to be black-boxes, such as deep neural network, to allow for any interpretability at all.

Local vs. Global

Local models generate explanation based on a few, typically only one, datapoint. The explanation is meant to give the user an idea of why a particular prediction was given. This is typically achieved by slightly changing the features to find the prediction boundary.

Global models give a more comprehensive overview of the model by explaining the behaviour of the model as a whole. The trained model, knowledge of the algorithm and the data is needed to create a global model.

2.6 Open vs. Closed Black-Box

There is no clear formulation as to what exactly constitutes an open or a closed black-box system in XAI, as a closed black-box usually refers to a system where only the input and output can be viewed. To make a black-box more interpretable one can open the box. By opening the box, we take a look under the hood and use specific knowledge about the underlying machine-learning algorithm as an extra layer of information, e.g. by looking at activation patterns (neuron activation) in an ANN or use information from the weights to calculate the importance of the input features (deconvolution, layer-wise relevance propagation) on the output. With respect to an open black-box, there is also the possibility of changing elements of the model to be more transparent, e.g. changing the architecture of an ANN to be more transparent.

2.7 Data Types

Within machine learning, there are multiple different data types used for training the different models. It's important to differentiate the most common types used. The most classical type is *tabular*, where every record shares the same set of features, and each feature can be either numerical, categorical or boolean. The other common type is *images*, where each record is an image with a corresponding target value. The final type is *textual*, which is any string representation, usually used for language modelling, topic classification etc. There are also some other types, albeit less common, including sequential data, sound frequencies, movement patterns or a mix of all of the above.

2.8 Background Theory from Specialization Project

There are central concepts, such as how to explain what constitutes a good explanation and how this relates to the type of user, presented here from the specialization project which is important for further reading.

2.8.1 Explanations Goals

Research by Miller et al. [64] indicate that most research conducted in the field of XAI do not have a solid definition of what constitutes a good explanation, instead relying on the author's intuition.

The primary goal when we talk about explainability in this thesis is making whatever explanation that is presented understandable to the user. It is hard to define what can and can not be classified as a good explanation [64]. Giving one type of explanation might make more sense in one context, whilst other aspects need to be considered for other users. Before tackling the problem of explainability, it would be helpful if we had some explanation goals or properties which we could use to evaluate an explanation system as a whole. Sørmo et al. [88] present five explanation goals that can be regarded as inherently important in any explanatory systems:

1. **Transparency**: Explain how the system reached the answer.
2. **Justification**: Explain why the answer is a good answer.
3. **Relevance**: Explain why a question asked is relevant.
4. **Conceptualization**: Clarify the meaning of concepts.
5. **Learning**: Teach the user about the domain.

Transparency may be most relevant to knowledge engineers who wish to debug the system. It is also useful where it may be life critical for the user to know how the system reached the answer, e.g. a physician treating a cancer patient. Ensuring trust is probably the most important property of an explanation system. Explaining why the system should be trusted, through *justification*, would therefore, be helpful in this regard.

G. Ras et al. [72] highlight another set of properties which they argue any explanation system should possess. The closer the system is to these properties, the better it should perform in a general sense.

1. **High Fidelity**: The degree to which the interpretation method agrees with the input-output mapping of the system. In other words, to the degree the system is able to generate

correct explanations with respect to the input and output of the system it's set out to explain.

2. **High Interpretability**: To which extent a user can obtain true insight into how actionable outcomes are obtained. High interpretability is closely related to what degree a user would be able to learn from the system.

   (a) **High Clarity**: The degree to which the resulting explanation is unambiguous, i.e. the explanation cannot be misinterpreted.

   (b) **High Parsimony**: An parsimonious explanation is a simple explanation. Related to Occams's razor.

3. **High Generalizability**: The degree to which the system can be applied to different underlying architectures.

4. **High Explanatory Power**: How many different questions the system can answer.

2.8.2 Users

Another aspect that is important to consider, is **who** will be utilizing the system. Different users have different needs [73] from among the goals presented by Sørmo et al. [88].
The users can be divided in two categories. One is the **lay user** of the system. This can be the *owner* of the system, the *end user*, the *data subject* or the *stakeholders*. Each of which has different needs. The *owner* is concerned with the explainability question about the capacity of the system, a justification of how the system came to a conclusion and aspects of accountability. The *end user* is concerned with the capabilities of the system and requires justification regarding the predictions made by the underlying system. The *data subject*, the entity whose information is being processed by the underlying system, is mostly concerned with the ethical and moral aspects that results from utilizing the decision from the system, i.e. the consequences of trusting the system on important matters. The *stakeholders* are mostly interested in the ethical and legal concerns raised in every phase of the system.
The other is the **expert user**: *engineers* and *developers*. The *engineers* are interested in an explanation of the functional nature of the system, i.e. what effect the various hyper-parameters have on the performance and how this can be used for model debugging. The *developers* seek to understand the goals of the system and figure out if they have been met by the underlying model and whether it understands the behaviour of the model in various use cases. Another expert user not mentioned by Ras et al. [73], is the *scientists* who seek to gain new knowledge or learn from the model by understanding the deep underlying relation on an explanation to understand more of the domain in which the model operates.

2.8.3 User Satisfaction

Alan Cooper's book [18] compares poorly designed software with inmates running an asylum. It is an analogy for how software designed by programmers, rather than an interaction designer, will suffer from a lack of user-oriented design. Miller et al. [64] argue that AI is facing a similar fate - despite the efforts being made to explain AI. Ultimately, the authors fear that AI researchers are designing agents for themselves rather than for the intended users. Based on an in-depth survey, Miller et al. present several key concepts within XAI which are user-oriented. One of these are contrastive explanations. Contrastive explanations may be more beneficial both to the user and the designer of the system. Explaining "Why A and not B?" is often easier than giving a full cause attribution because the system only needs to understand the difference between case A and B. If the system was only to explain, "Why A?", there is going to be an implicit contrast case assumed by the system which may be different from what the user had in mind. By formulating the contrast case explicitly, Miller et al. argues that this is more beneficial for both the user and the system.

Some of the research done in the field has to do with finding how different kind of justification types perform with respect to user satisfaction. Herlocker et al. [44] conducted an experiment on a movie recommender system and found that most users were satisfied when the system provided the neighbors' rating or when shown a single strong feature, such as a favorite actor. They were less receptive to complex justification techniques, such as a full neighbor graph. This is an indication that lay users prefer simple and concise explanations over more detailed complex ones.

Symeonidis et al. [87] show that providing the most important feature along with the user's past history were rated as more satisfactory by users. Bilgic et al. [11] has conducted a review of previous studies and found that feature-based justification was superior to neighbor-based justification.

Chapter 3

Background Research

In this chapter, we will summarize what we found from our previous specialization project, and expand on this in this chapter by exploring new areas related to CBR and explanation generation. Chapter 3.2 - 3.4 are grouped accordingly to three of the categories that were described in the previous chapter. In 3.5, we present what has been done on case-based explanations. This chapter is a direct attempt at tackling research question 1 on what is the current status of XAI and CBR.

3.1 Notable Research from the Specialization Project

This section has a selection of previous research from the specialization project [27].

International Joint Conference on Artificial Intelligence 2017 XAI (IJCAI 2017) [46]

"How should explainable models be designed?", "What type of user interactions should be supported?", "How should the quality of an explanation be measured?" and "What can we learn about XAI from fields which are not directly related to ML?". These are some of the questions raised by the organizers. IJCAI2017 is described by the organizers as a platform for researchers to learn and share their recent work in XAI. The main themes for this proceeding are: 1) Deep learnings techniques, 2) Other types of ML and knowledge acquisition models and 3) Application of symbolic logical methods to facilitate their use in applications where supporting explanations are critical.

The IJCAI2017 conference had meta-papers discussing and addressing the research field. One of the papers made the case on why it's important to work alongside the social- and behaviour-science community to gain a better understanding of what constitutes a good explanation. Another paper highlighted how racial, gender and other biases present in the society have proven

to influence the decisions of AI systems and how these biases can be exposed with the assistance of XAI. Other papers presented various proposals for practical implementations in different domains ranging from planning to robotics using techniques such as feature analysis, Visual Question Answering (VQA), Probabilistic Setential Decision Diagram, Explainable Principal Component Analysis, and Gray-Box Decision Characterization. The practical papers show promising results, with the meta-papers highlighting important aspects worth dedicating more research to.

International Joint Conference on Artificial Intelligence 2018 XAI (IJCAI 2018) [22]
IJCAI 2018 extends the conference from IJCAI17. The conference featured a broad range of papers in XAI from planning to Reinforcement Learning. With the introduction to GDPR, we see an emphasis on explanation in terms of different user needs. Systems which are used directly by users, like recommendation systems, planning, and RL, are the reoccurring topics on explanation. Some of the main topics included: improved trust in the system and increased transparency and interpretability. The methods used to achieve this ranged from combining models of varying transparency to achieve better explainability to modifying the underlying models to be more transparent. Some techniques included decomposing the reward in a RL setting using salience maps to aid visual explanations, modifying the underlying black-box, evaluating planning and backtracking in search, among many others.

International Conference on Cased-Based Reasoning 2018 (ICCBR 2018) [65]
ICCBR is the first conference with a focus on using CBR to explain intelligent systems. The conference had three position papers and five research and application papers. Two of the papers focused on combining CBR with other techniques to create an explainable recommender system. Among several interesting techniques, the first paper introduced a simple yet effective technique for a more convincing explanation called **odds ratio**. It's a way to tell the user that A was chosen over B because it was twice as likely, instead of just "A was chosen because it was more likely than B". Other interesting ideas was combining an ensemble of learners to produce several explanations that could be merged into one explanation using fuzzy logic. The first part of the conference concluded with Adam J Johs et al. review on how to measure the quality of explanations in explainable case-based reasoning (XCBR). They found that the current measurement of explanation tends to be binary: either *is of quality* or *is not of quality*. The authors therefore call for a framework for quality measurement.

Attribution

An attribution method is a local (instance based) explanation for how each feature or attribute contributed to the final prediction, e.g. which pixels in an image were related. This is also called

the *sensitivity mapping* between input and prediction.

Perturbation

Perturbation is when we change some input with respect to possible alternative input to observe to what degree an input feature affects the output. The major drawback to this approach is the computational cost, which for models with sufficiently large input spaces is not possible to evaluate every possible perturbation for. Another drawback is the *saturation problem*[82]. This is when changes to the input have no observable effect, leading the observer to conclude the input features are not relevant, when in reality the input is simply being saturated (contribution is being drowned out). Illustrated with a simple example: let's say we have a simple function $y = max(x_1, x_2)$ when $x_1 = 1, x_2 = 1 \implies y = 1$. If we perturb $x_1 = 0$ to observe changes, we wont see any difference to the output, but in reality x_1 is just as important as x_2 on the function. One solution to the saturation problem is to perturb combinations of inputs. However, this increases the computational cost even further.

Shapley values

The Shapley value is from cooperative game theory and measures the average marginal effect of including input over all possible orderings in which inputs can be included [82] [78]. In other words, it says something about how important each player(feature) is to the overall cooperation, and what payoff(score) can be expected from said player(feature).

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}(x_{S \cup \{i\}}) - f_s(x_s)}] \tag{3.1}$$

The Shapley value, defined in equation 3.1, where F is the set of all features and S is all the feature subsets we are interested in computing. To compute the importance value for each feature, which represents the effect on the model prediction of including that feature, a model $f_{S \cup \{i\}}$ is trained with that feature present and another f_S with the feature withheld. The two models' predictions are compared to the current input. x_s is the values of the input set S. The Shapley values then computed are used as feature attributions, which are weighted for all permutations [60]. The actual Shapley value has in most cases a high computational cost, with the requirements to retrain with the number of features in S and calculate for every possible permutation of features of interest.

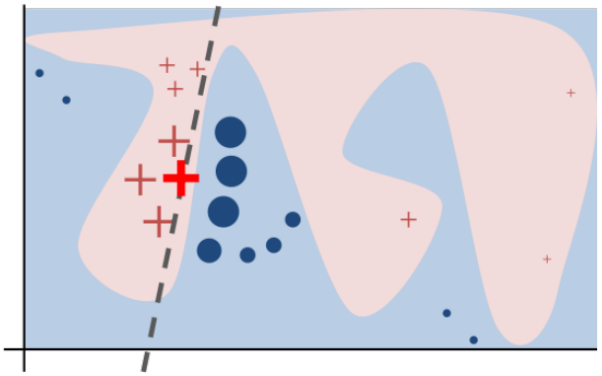


FIGURE 3.1: The intuition for LIME. The red cross is observation and the rest are perturbations of this observation or closely related observations. The background represent the model's complex decision function which we try to approximate. The dashed line represent the local linear approximation [93].

LIME

Tulio Ribeiro et al. [93] created a model-agnostic tool to create explanations of the black-box algorithm used - a tool known as Local Interpretable Model-Agnostic Explanations (LIME). It works by approximating (learning) the black-box model locally around the prediction, creating multiple linear explanations for a given black-box. LIME asserts that it is possible to fit a simple model around a single observation, which will approximate how the model behaves locally. This approximation is created by perturbing the observations in a number of different ways with respect to the feature dimensions to generate locally similar observations for our area of interest. The perturbed observations are predicted by the black-box model, which an interpretable model approximates the prediction of (see figure 3.1). This can be used to explain a given feature dimension along with the similarity weight between the perturbations and observations later to a user. As a user would easily get swamped with information for every feature, submodular pick was used to only select the instances with greatest coverage (broad and non-redundant) over the feature space.

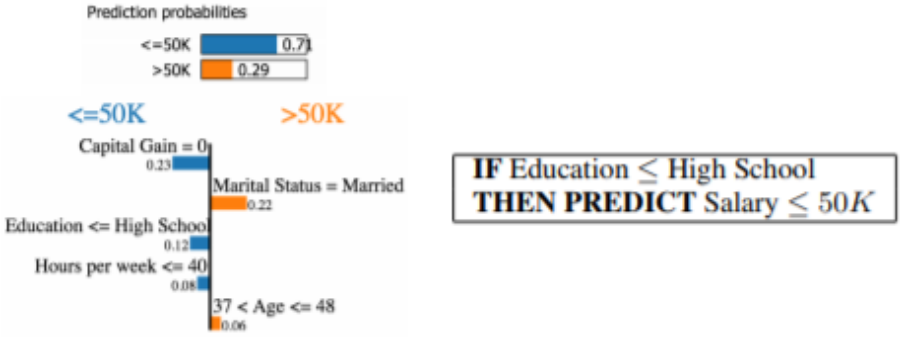


FIGURE 3.2: Explanation from LIME(Left) and aLIME(Right), aLIME with a rule based explanation, whilst LIME use an overview of each attributes importance [76].

Anchor

aLIME (anchor LIME) extends LIME by explaining individual predictions with if-then rules, combining the benefits of local model-agnostic explanations with the interpretability of rules where an anchor (explanation) is a rule that sufficiently (Probably Approximately Correct) explain a prediction, and any changes to the rest of the instance do not break the connection with the anchor to explain a prediction. The explanations are more interpretable to a user, and outperform regular LIME in terms of coverage with its flexibility to alter greediness (how specific to the particular input) of an explanation [76]. Later an improvement of aLIME called Anchor was introduced. It focuses on solving one big problem with LIME, namely the "unclear coverage" problem [75]. LIME is unclear of the coverage of a local explanations, which can mislead users into thinking a local explanation applies to unseen instances when it does not, with no clear boundary of a given local explanation. Anchors only present explanations that are sufficiently meet the conditions for the model with high probability. There are two algorithms which Anchor uses to generate these rules: Bottom-up Construction and Beam-Search Construction. Bottom-up try to find the rule with best coverage with a greedy search, which is guaranteed to find short anchors. It does, however, have a few shortcomings when finding this anchor: it only maintains a single rule at a time that it incrementally augments, losing any sub-optimal choices. Beam-Search instead maintain the best candidates, with a guided beam search(explore most promising candidates) among multiple candidates to identify anchors with high coverage.

Formally an anchor A is defined by Ribeiro et al. [75] as a rule (set of predicates). With the black box as $f : X \to Y$ and $x \in X$ where X is all possible instances. The goal is to explain $f(x)$ to a user, where $f(x)$ is the individual prediction on instance x. This is done by perturbing the instance x to some perturbation distribution D. In tabular classification problems, D is the data-set. $\mathbb{E}$ is the calculated error rate of the equation. A is an anchor on instance x. If an anchor fit on x, $A(x) = 1$, where $A(x)$ returns 1 if all its feature predicates are true for the instance x. An example of an anchor that would fit any instance, would be the instance itself. E.g. $if\ x_0 = x_0\ and\ x_1 = x_1 \ldots$ the anchor fit.

$$\mathbb{E}_{D(z|A)}[\mathbb{1}_{f(x)=f(z)}] \geq \tau, A(x) = 1 \tag{3.2}$$

With a threshold of τ on precision of the anchors A, with sample z from a section of the validation data.

We are given two measures for each anchor A: precision and coverage.

$$\text{prec}(A) = \mathbb{E}_{D(z|A)}\ [\mathbb{1}_{f(x)=f(z)}] \tag{3.3}$$

$$\text{cov}(A) = \mathbb{E}_{D(z)}[A(z)] \tag{3.4}$$

In our example, precision equation 3.3 is defined as: *of all instances in D, where the set of predicates in A apply, how many share the same prediction.* Coverage equation 3.4 is how "wide" the anchor can be applied to the perturbation distribution. *How large percentage of the validation data D can the anchor A be applied to.*

An intuitive comparison between Anchor and LIME is shown is figure 3.3.

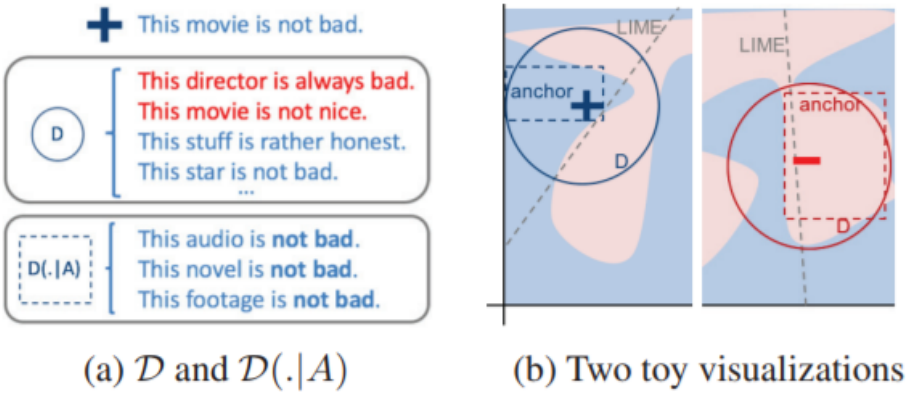


FIGURE 3.3: Explanations on the left, with LIME on top and Anchor on bottom. b) is an intuition of how Anchor differ from LIME's explanation [75].

DeepLIFT

Deep Learning Important FeaTures (DeepLIFT) [82] which is based on the idea of computing importance in terms of difference from a reference state to the problem, with reference state being the default or neutral input, to mitigate internal zero-gradients saturation (vanishing gradients between internal layers during backpropagation). Using the reference, we calculate the difference-from-reference of the features, which is the contribution or "blame" of an input feature (rescale rule). The improved version of DeepLIFT optionally gives separate consideration to negative and positive contributions, using the RevealCancel Rule. The intuition is to consider the impact of the positive terms in the absence of negative terms, and the impact of negative terms in the absence of positive terms. We are thus able to alleviate some of the issues with positive and negative terms canceling each other out. DeepLIFT can with this reveal dependencies not caught by other approaches by using the RevealCancel rule. Figure 3.5 show an example of this. It can also be considered a faster approximating of the Shapley values of each attribute.

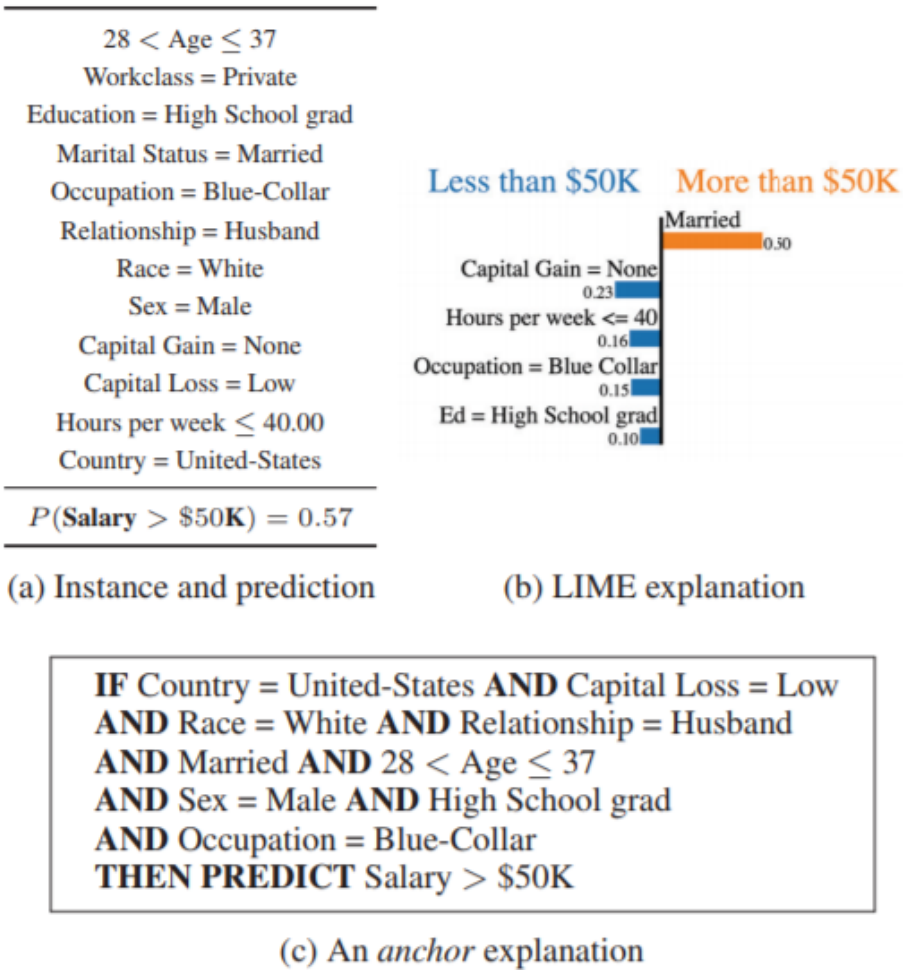


FIGURE 3.4: Difference between an explanation from LIME on a target instance vs Anchor: Anchor has a series of specific IF-ELSE rules, while LIME only has feature attribution [75].

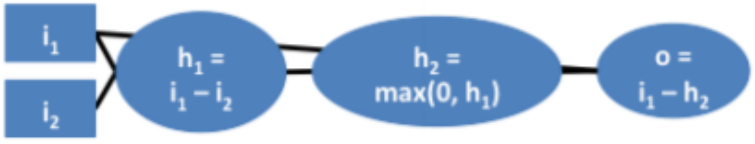


FIGURE 3.5: Network computing $o = min(i_i, i_2)$, assume $i_1 = i_2 = 0$, when $i_1 < i_2$ then $\frac{dy}{di_2} = 0$ and when $i_2 < i_1$ then $\frac{do}{di_1} = 0$. Using LRP or Integradent Gradients at this network would result in importance assigned exclusively to either i_1 or i_2. With the RevealCancel rule, DeepLIFT assigns $0.5min(i_1, i_2)$ attribution to both inputs [82].

Layer-Wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) [12] is a method to calculate the importance of each feature by applying a propagation rule that distributes class relevance found at a given layer onto the previous layer, from the output back to the input. Overview of this process is depicted in figure 3.6. The explanations are presented as these relevance distributions among the input features (pixels in this instance). Later Binder et al. [13], proposed an extension to the original

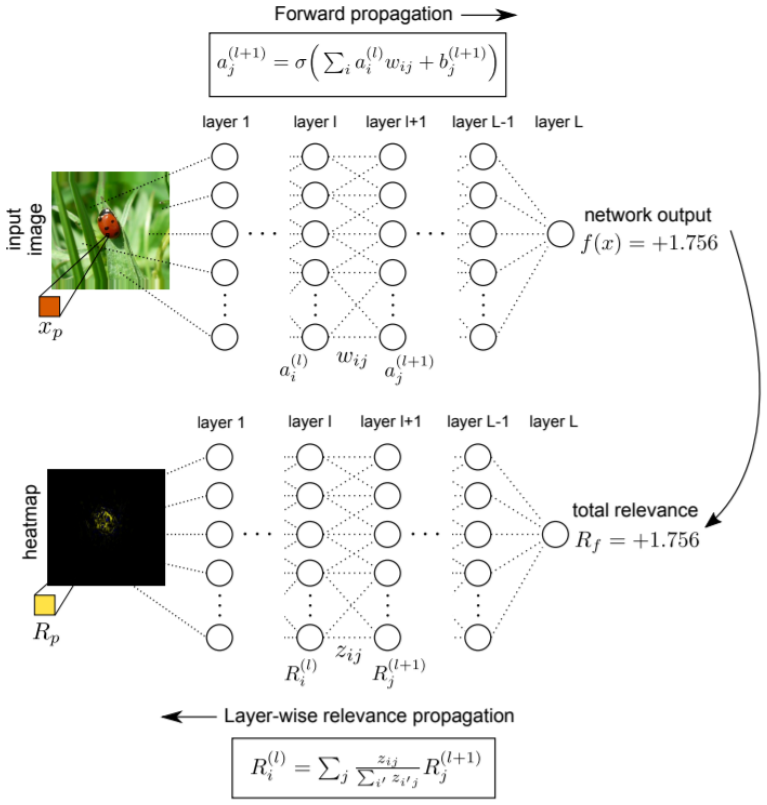


FIGURE 3.6: Explanation from Layer-Wise Relevance Propagation on image classification task, using pixel decomposition heatmaps with relevance score R [12].

framework to handle product-type non-linearity networks with local re-normalization (e.g. CNN) layers, based on first-order Taylor expansions.

SHAP

SHapley Additive exPlanations (SHAP) [60] is a unified framework based on previous methods such as LIME, DeepLIFT, Layer-Wise Relevance Propagation, Classic Shapley Value Estimation and game theory (Shapley values 3.1), which is used to interpret predictions. SHAP uses a variation of equation 3.1, called *shapley sampling values*, which applies sampling approximations and approximates the effect of removing a variable from the model by integrating over samples from the training set. Removing the need to re-train and the differences needed to calculate to fewer than $2^{|F|}$. Essentially turning the Shapley Values method into an optimization problem. With a focus on some desirable properties which previous methods do not fully adhere to, namely local accuracy.

Local accuracy, equation 3.5: When approximating the original model f for a specific input, the explanation model g should at least match the output of f for the simplified input x', with M number of simplified features.

Missingness (equation 3.6): Features missing (no value) in the original input should have no

impact/contribution.

Consistency (equation 3.7): If a model changes so that some simplified input's contribution increase or stay the same regardless of the other inputs, that input's attribution should not decrease.

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{3.5}$$

$$x'_t = 0 \implies \phi_i = 0 \tag{3.6}$$

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i) \tag{3.7}$$

With Shapley values upholding these properties, Lundberg et al. [60] derive the SHAP value equation 3.8. $|z*|$ is the number of non-zero entries in z' and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

$$\phi_i(f, x) = \sum_{z' \subseteq} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)] \tag{3.8}$$

The author also presents some model-specific approaches with Shapley values. Kernel SHAP (Linear LIME + Shapley values), Linear SHAP, low-Order SHAP, Max SHAP and DeepShap (DeepLIFT + Shapley values).

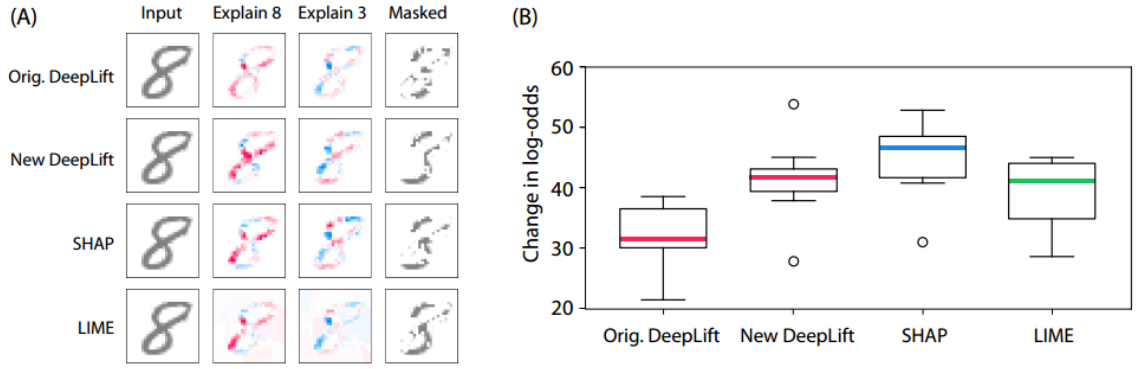


FIGURE 3.7: Comparison between the different attribute estimators on MNIST dataset, where the black-box model(using CNN) predicts it could be an 8 or a 3. A) Red areas increase the probability, whilst Blue decrease the probability. Masked show the removed pixels in order to get from 8 to 3. B) show the change in log odds when masking over 20 images, which also show how DeepLIFT is a close approximation to SHAP [60].

Integrated Gradients

Sundararajan et al. [86] present two axioms that attribution methods must satisfy: *sensitivity* and *implementation invariance*. *Sensitivity* means that if for every input and baseline that differ in one feature, but have different predictions, then the differing feature should be given a non-zero attribution. i.e. features that cause a disturbance/change to the prediction should

be regarded as having high correlation to the prediction. The second axiom is *implementation invariance*, where two networks are functionally equivalent if the outputs are equal for all inputs, despite different underlying architectures, the attribution of each feature should remain identical for each network.

According to Sundararajan et al., other attribution methods like LRP and DeepLIFT do not meet both these demands. LRP and DeepLIFT break implementation invariance since they both use discrete gradient calculation, which relies on the chain-rule to compute the discrete gradients, which itself doesn't satisfy this axiom. Integrated Gradients combines the implementation invariance of gradients along with the sensitivity of techniques like LRP or DeepLIFT, and is defined as the path integral of the gradients along the straight line path from the baseline x' to the input x.

$$IntegratedGrad_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x')}{\partial x_i} d\alpha \tag{3.9}$$

$$IntegratedGrad_i(x) \approx (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m} \tag{3.10}$$

The equation 3.9, references the integrated gradient along the i^{th} dimention, with input x and baseline x', where F is the prediction function. Equation 3.10 is an approximation of equation 3.9 using m number of Riemann approximations over the integral to speed up the calculation. The paper also mentioned how 20 to 300 steps usually were enough in practice to approximate the integral within 5% error.

Name	Authors. Ref.	Year	Model-agnostic vs. model-specific	Intrinsic vs. Post-hoc	Local vs. global
SHAP	Lundberg et al. [60]	2017	agnostic/specific(extensions)	post-hoc	local
Shapley values	Roth [78]	1988(1953)	agnostic	post-hoc	local
LIME	Tulio Ribeiro et al. [93]	2016	agnostic	post-hoc	local
ANCHORS	Ribeiro et al. [75]	2018	agnostic	post-hoc	local+
DeepLIFT	Shrikumar et al. [82]	2017	specific	post-hoc	local
LRP	Binder et al. [12]	2016	agnostic(DNN)	post-hoc	local
Integraded Gradients	Sundararajan et al. [86]	2017	specific(DNN)	post-hoc	local

TABLE 3.1: Overview of the most notable XAI papers researched during the specialization project. Some have no clear distinction between the categories, and as such contain both.

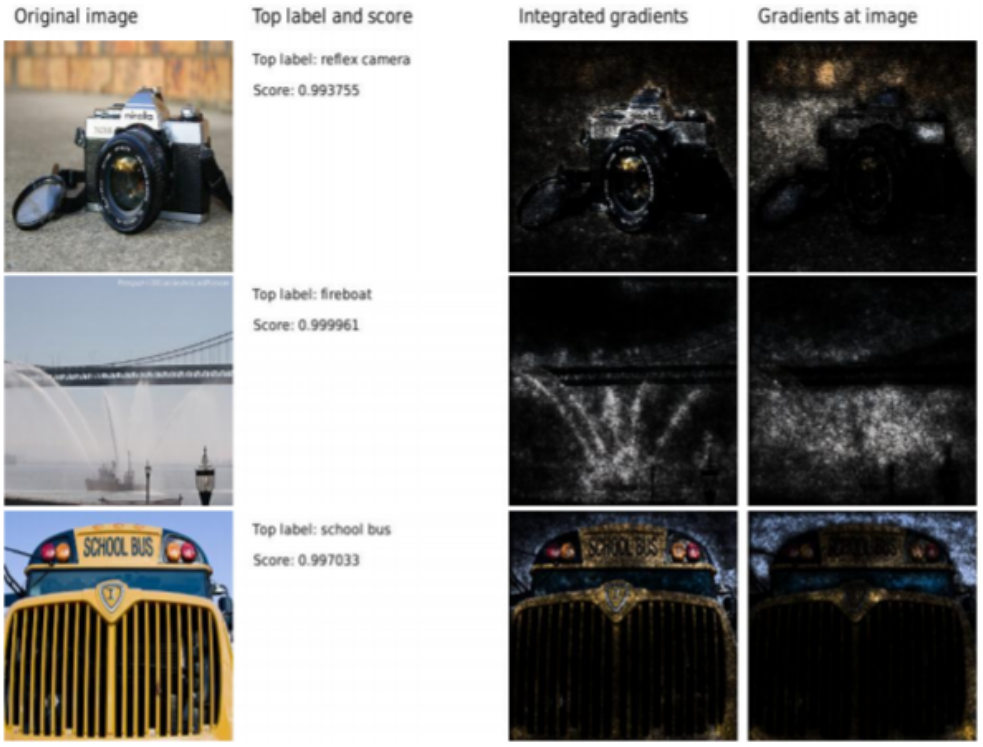


FIGURE 3.8: Comparison of integrated gradients with the gradients at the images. Visualization of *Integrated gradients* and visualization of $gradients * image$. The integrated gradients are better at reflecting distinct features of the image important for the classification task [86].

The results shown in figure 3.8, show how integrated gradients compare to regular gradients.

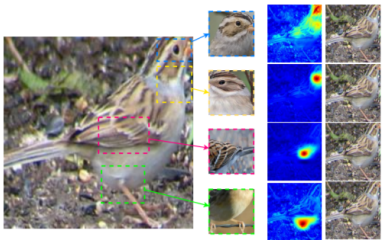
LIME, SHAP (not to be confused with Shapley values), and Anchors are all model-agnostic approaches which work independently of which algorithm was used to learn the classifier. Integrated Gradient and LRP only function on DNNs, whilst DeepLIFT only works with some DNNs.

Table 3.1 show an overview of the most notable papers research from the specialization project and which sub-category they fit in.

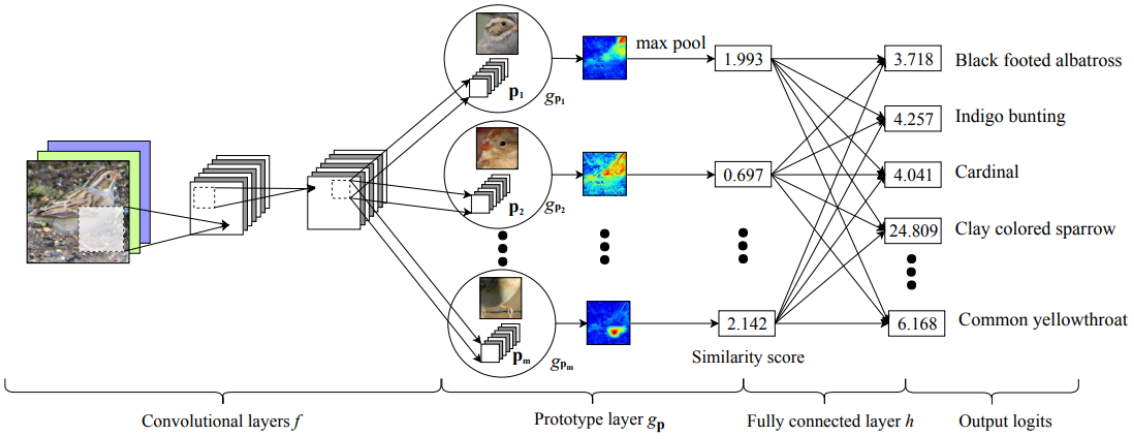
3.2 Deep Explanation

Deep explanation is a modified or hybrid deep learning approach which learns more explainable features or explainable representations for DNNs. Since it's specific for DNNs, it can be considered to be model-specific. The deep explanation approach is among the least explored in the field, and the number of papers available shows this.

Chen et al. [16] created a deep neural architecture that dissects the image by finding prototypical parts related to the image classification, and use previous evidence for the prototypes to make a prediction. The architecture is built using a sequence of convolutions layers, a prototype layer, a fully connected layer, followed by the output logits layer. The training of this network is done in separate stages to target individual layers: the network as a whole, followed by training of the prototypical layer, followed by an optimization of the last layer. The explanation generated from this network is shown in figure 3.9.



(A) Explanation



(B) Architecture

FIGURE 3.9: (A) Image of a clay colored sparrow and the learned prototypical parts of a clay colored sparrow used to classify the bird's species. The explanation is the prototype comparison between similar cases. (B) the altered architecture used to achieve these results. [16]

This prototype classification approach from Chen et al. is not new, previous attempts have achieved similar results as shown in Li et al. [57]. Where CBR is built into the network itself, with a special encoding layer in to automatically find the best prototypes, these prototypes (cases) are compared to new encoded input instances, where the most probable prototype give the corresponding prediction. The explanation generation part is shown in figure 3.10 with

the reconstructed input part, which lives in the same space as the encoded inputs, can be used to visualize the learned prototypes during training, and partially trace the path of a new classification task, with the activation weights to each prototype. Unlike CBR, this system does not consist of the typical retain step, the case-base is instead filled during training of the prototype classifier network h.

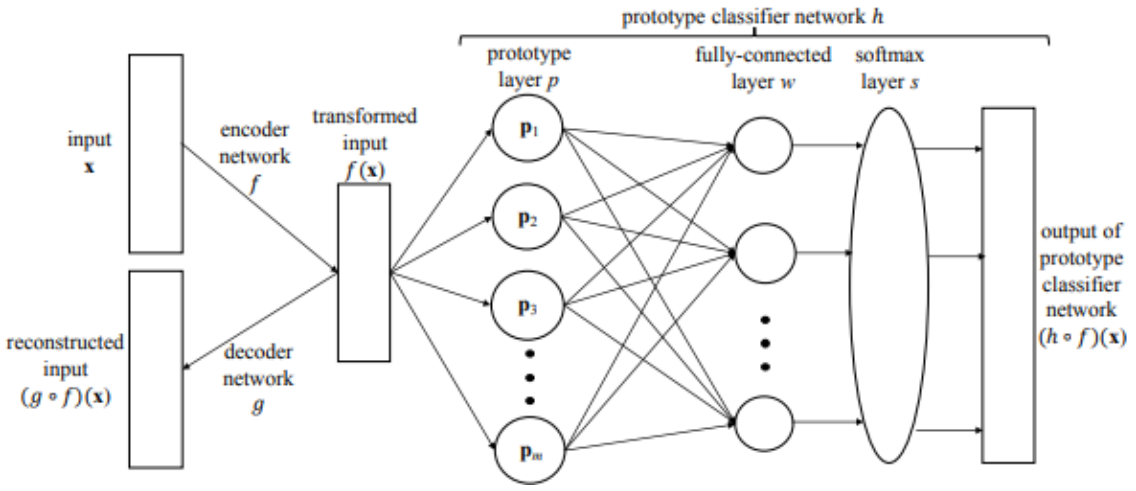


FIGURE 3.10: Network Architecture. Reconstructed input is the explanation. Prototypes are the learned cases during training. Classification part consists of retrieval from most similar prototype activations [57].

Tran et al. [91] trained a deep belief network, and showed how to extract knowledge from the individual Restricted Boltzmann Machine (RBM) in the network. Using an extension of *modus ponens* logical inference rules into *confidence* rules of biconditionals (if-and-only-if). From these rules, they gained the ability to interpret the decision of each layer in the network, which could be ultimately used to explain. Knowledge extraction from a deep neural network is unfortunately considered not to be practical due to an increased computational overhead requirement, as this would likely be done end-to-end.

Lei et al. [56] incorporate rationale (a set of reasons or a logical basis for a course of action or belief) generation as an integral part of the overall learning process, by combining two modular components: a generator that specifies a distribution over possible rationales and an encoder that uses their rationales to map to task-specific target values. The example in figure 3.11 show that the rationale is simply the specific sequence of words that justify the classification value. The authors compare this to a linear approximation of the rationale and note that it suffers a loss of prediction accuracy when approximating the model locally [93] with a simple model. Experimentally this holds up with letting the DNN generate these local justifications as well, as a model is not linear in every region.

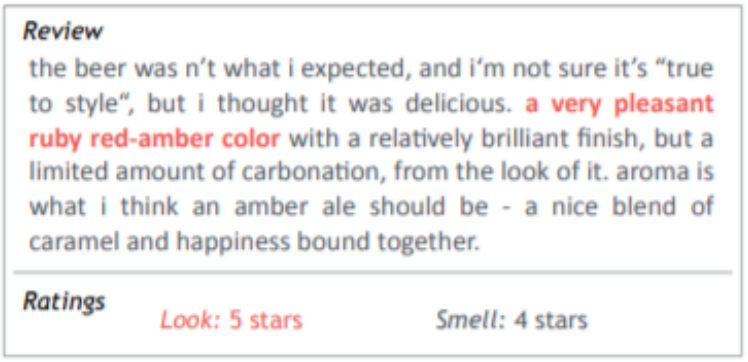


FIGURE 3.11: Example of a beer review ranking in two categories. The justification for *look* is highlighted as the rationale [56].

Tsang et al. [92] treats the knowledge learned in a DNN entangled between the intermediate hidden layers with the shared representation learning during training. Disentanglement is referred to as extracting an interpretable representation from the neural network, either during or after training. The aim of Tsang et al. is to learn or uncover General Additive Models (GAM) blocks with interactions as a subset of a feed-forward network with fully-connected layers, to obtain the intrinsic lower-order and interpretable structure of the network. Supported by regularization during training, limits the maximum interaction order, and encourage smaller interaction orders and block sparsity. They show that even simple neural networks do indeed entangle information between layers. Figure 3.12 show this framework on a single layer, where we have to select a number of GAM blocks B beforehand, either a large number and cancel out unused blocks, or a smaller amount to keep interpretability. To explain the decisions of the network, we use the GAM blocks to map out their corresponding value graphs. Note that this approach "only" applies to one layer at a time, working with multiple layers at a time is still limited to run-time and the complexity of the network.

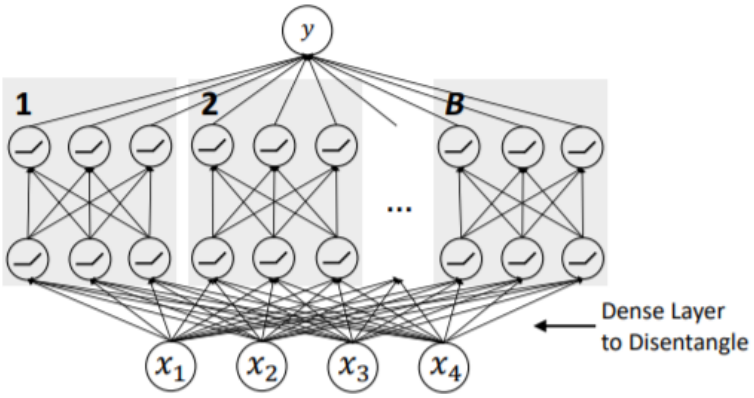


FIGURE 3.12: NIT(Neural Interaction Transparency) Architecture to disentangle a dense layer. Each B (GAM) block correspond to a single interaction or univariate variable [92]

Alvarez Melis et al. [3] designed a Self-Explaining Neural Network (SENN), which operates as a simple interpretable model locally, allowing for point-wise interpretation. This was achieved with

a regularization scheme that ensures that the model behaves like a linear model locally by training the function $h(x)$ (model) as an auto-encoder, enforcing diversity through sparsity regularization, and providing interpretation on the concepts by prototyping. Overview of the architecture is shown in figure 3.13. It consists of 3 components: a concept encoder that transforms the input into a small set of interpretable basis features, an input-dependent parameterizer that generates relevance scores, and an aggregation function that combines to produce a prediction. The robustness loss over the parameters $\theta(x)$ encourages the full model to behave as a linear function locally on $h(x)$, to encourage more interpretable (linear relation) explanation on a given prediction.

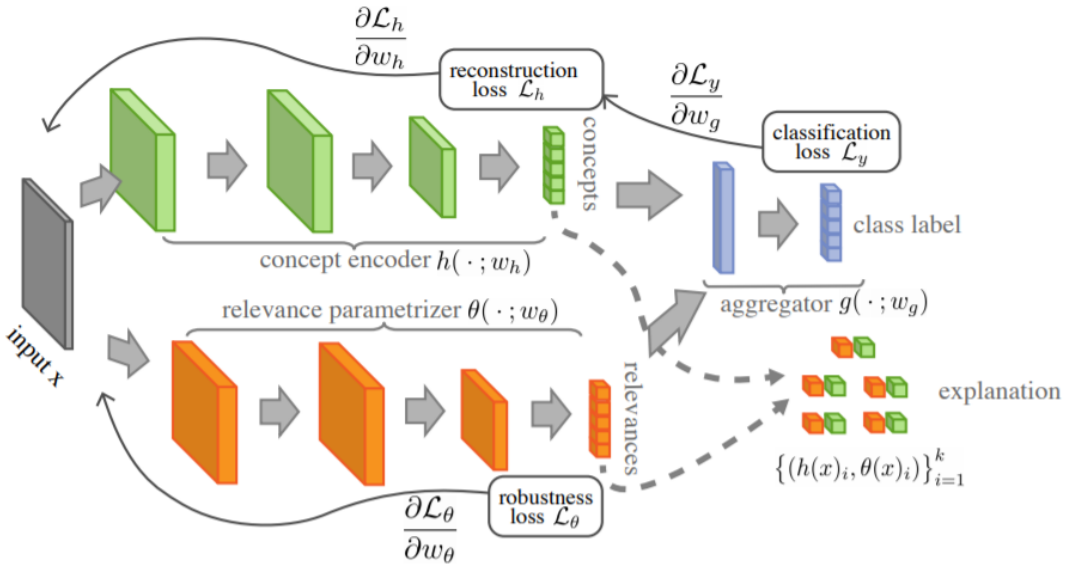


FIGURE 3.13: SENN architecture overview [3]

Krakovna et al. [51] induct the black-box model using Hidden Markov Models (HMM) on a Recurrent Neural Network (RNN). It's a hybrid approach to approximate the decisions of the RNN to increase the interpretability of the network, combining the predictive power of RNNs and the interpretability of HMMs. Results show that the RNN and HMM learn complimentary features which can be used to explain the prediction.

3.3 Interpretable Models

Any learning algorithm that is also inherently interpretable by design is said to be an interpretable model. Only a select few models can be placed in this category. The biggest concern of interpretable models has to do with the limitation of the method used to learn the problem, as each has its own weakness in term of knowledge representation and knowledge reasoning on the problem and the complexity of the domain. All of which effect interpretability as stated by Freitas [32]. Decision trees, which are inherently interpretable, works by following a set of if-then

clauses structured as a tree. The problem with this method is that it is not very scalable in terms of interpretability. The larger the tree grows with respect to the complexity of the problem, the less interpretable it becomes. The same can be said for other rule-based methods which explain in a similar fashion to decision trees. Linear models like Support-Vector Machine (SVM) are usually a bit more interpretable from their transparency, where the vectors can be used to explain the decisions on a new problem. Unfortunately, these suffer the same fate as decision trees: as we increase the number of dimensions (read: features), it's very difficult to interpret the prediction into an explanation. k-nearest neighbors (k-NN) faces the same challenges: depending on the similarity metric used, it can be interpretable at every level. However, this is not practical as we increase the number of dimensions, as we're likely to lose the bigger picture. Simply knowing feature $x_1 - x_3$ were deciding factors is not enough. CBR on the other hand, can be considered somewhat interpretable, depending on the complexity of the similarity measure used and previous cases, where we don't need to focus on all the dimensions, rather on only the ones that were most important for the retrieval. Where as k-NN looks at all the features, CBR usually only looks at the most important features.

The method that is the most interpretable according to Freitas [32], is rule based classification methods. Their textual nature allow for easy readability by an user. Individual rules are also very modular, allowing us to look at a select few relevant rules at a time to see the "local patterns" as the explanation.

Malioutov et al. [61], approach the problem of achieving interpretable models as a sparse signal recovery problem from boolean algebra known as Boolean compressed sensing. Using threshold group testing to formulate the interpretable rules in a non-heuristic approach. They developed a linear programming relaxation duality-based technique (1Rule), which guarantees to approach a near-optimal solution after training a classifier only on a small subset. The rules proposed were said to be interpretable from the linearity and the small size. The accuracy trade-off was minimal, and the accuracy was said to be better than existing interpretable methods such as RuSC(Set covering approach rule learner), RuB(Boosting approach rule learner), DList (Decision lists algorithm), C5.0 (C5.0 Release 2.06 algorithm with rule set option in SPSS), and CART(Classification and regression trees algorithm in Matlab's classregtree function).

Lou et al. [59] extended on regular General Additive Models (GAM) with pairwise interactions between features into a system called Generalized Additive Models plus Interactions (GA^2M). GAMs are the sums of univariate models. They have a high degree of interpretability because of their simplicity, but as a result, they suffer in accuracy. GA^2M, consists of univariate terms and a small number of pairwise interactive terms, which can easily be visualized to the user. To deal with the large number of feature permutations, they developed a method called FAST for ranking all possible pairs of features as candidates for inclusion into the model. GA^2M with

FAST had almost the same performance as the best fully-complex (black-box) models on a number of real datasets (in 2013).

Lakkaraju et al. [53] created a rule-based framework, called *interpretable decision sets*, which seeks to create rules in a flat structure as opposed to a hierarchical structure like in a decision list. The idea is that each rule, in the form of if-then clauses, can stand on their own not depending on other rules to be true. The user should be able to read each rule on its own and understand what leads to a specific classification. The authors claim this approach is more favorable than Decision List to achieve interpretability.

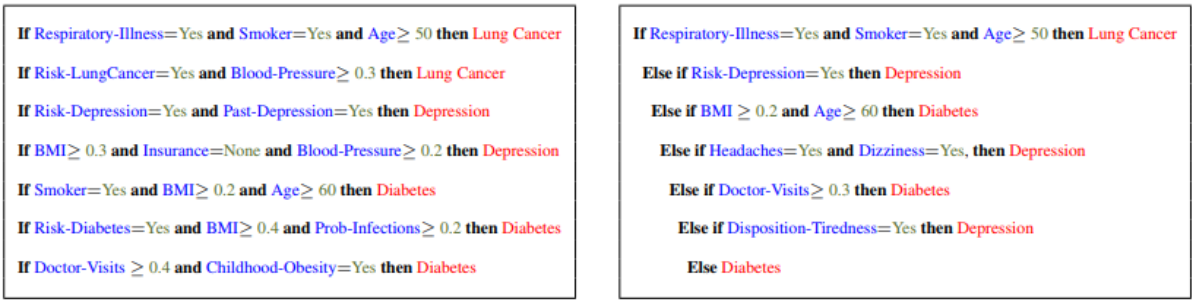


FIGURE 3.14: A comparison between an interpretable decision set (left) and a decision list (right). The rules are created from the same dataset.

Figure 3.14 compares the type of rules that would be generated for a decision set and decision list on the same dataset. Each new rule in a decision list (right) is dependent on the previous rule not to be true to be considered and therefore has to be read in chronological order. Each rule in the decision set (left) on the other hand is independent of any other rule and can therefore be considered in any order.

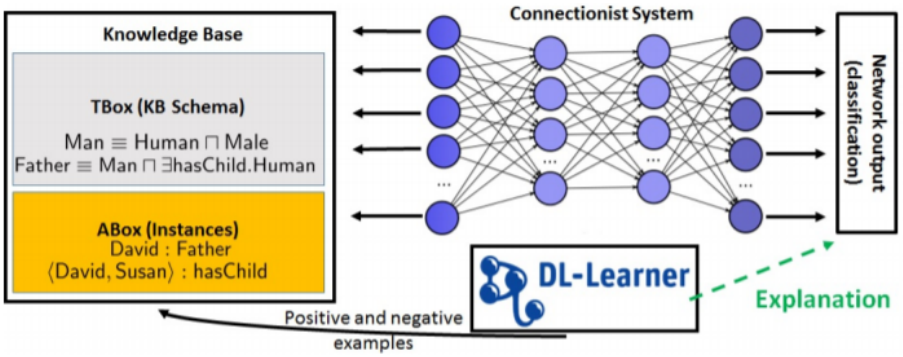
3.4 Model Induction

A very common technique for explaining a black-box is treating the black-box as an oracle. The oracle is then queried to generate more training instances to help with common problems such as an incomplete or small training set. This is later used to train a transparent model or more interpretable model to hopefully derive explanations that are consistent with the black-box.

Sarker et al. [80] combined a trained neural network with semantic web technologies to explain the input-output mapping by leveraging data from the World Wide Web as background knowledge to obtain more concise explanations. It works by using a DL-learner (Description Logic) as the key tool to arrive at the explanations. The trained ANN act as an oracle and its inputs are used to map to the background knowledge base from the Suggested Upper Merged Ontology (SUMO) [1]. The oracle distinguishes between positive and negative examples. The DL-learner

[1] `http://www.adampease.org/OP/`

is then run on the example set to provide explanations based on the concepts encoded in the background knowledge. If the network is erroneous, we can use the knowledge base to look for missed annotations. The figure 3.15 show the proposed solutions (right) that the Descriptive Logic learner proposes to the examples. The problem with this approach is that we don't know how the network solved the problem, as we don't know its internal knowledge and reasoning, even if the explanations hold with the background knowledge.



(A) Semantic Web Tech Archicecture

∃contains.Window	(1)	∃contains.LandTransitway	(6)
∃contains.Transitway	(2)	∃contains.LandArea	(7)
∃contains.SelfConnectedObject	(3)	∃contains.Building	(8)
∃contains.Roadway	(4)	∀contains.¬Floor	(9)
∃contains.Road	(5)	∀contains.¬Ceiling	(10)

(B) Proposed solution

FIGURE 3.15: (A) The conceptual architecture. (B) The proposed solutions presented by the DL-learner on images classified with scene annotations(objects). Every solution that explain the distinction of the examples with respect to the knowledge base. [80]

Deep neural network Rule Extraction via Decision tree induction (DeepRED) [98] is an extension to the decompositional rule extraction algorithm Continuous/discrete Rule Extractor via Decision tree Induction (CRED)[81] to more than one hidden layer. Rule extraction is done in a step-wise process, with one layer at a time where one layer is used to explain the next. As a result, one is left with a rule-set that describe each layer of the DNN by their respective preceding layers, which are then merged to mimic the whole network along with input pruning techniques from Augasta et al. [8] to achieve more comprehensible and generalized rules.

Guidotti et al. [40] created an agnostic method that uses a local interpretable predictor on a syntactical neighborhood generated by a genetic algorithm called LOcal Rule-based Explanations (LORE). It works similarly to LIME [93] in the sense that it first learns a local interpretable predictor on a neighborhood, with the exception of permuting using a genetic algorithm instead of randomly improving the value of the generated instance. From this, it derives a meaningful explanation consisting of a decision rule which justifies the reasons for the decisions, along with a set of counterfactual rules to suggest changes that can be made on the input features to change

the predicted outcome. The rules are generated using decision trees on the neighbourhood to mimic the behaviour of the black-box locally to maintain fidelity.

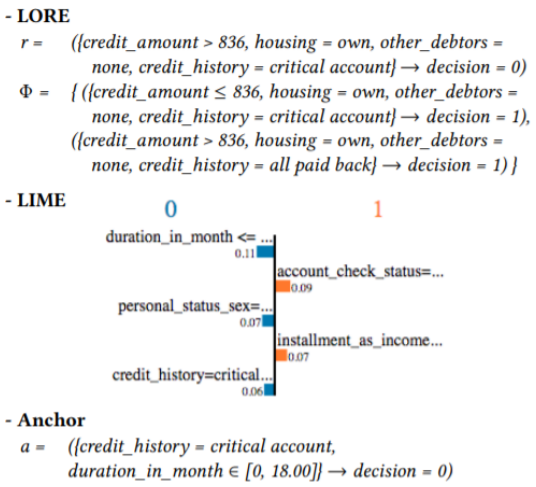


FIGURE 3.16: Comparison of explanation given from LIME, LORE and Anchor. Note the extra counter-factual from LORE [40]

Figure 3.16 show the comparison between LORE, LIME and Anchor. The big difference is in the length of the explanation between LORE/Anchor and LIME, with LIME yielding a lot more detailed (more specific) local explanation. Anchor gives a better global representation of the explanation, whereas LORE gives the full local path in the Decision Tree. In addition, LORE gives a counterfactual explanation as well, generated from the closest difference in the decision tree.

Dhurandhar et al. [23] created a model-agnostic explanation method called Contrastive Explanations Method (CEM) that generates contrastive explanations. These are created by finding a minimal amount of features in the input that are sufficient in themselves to provide the same classification (pertinent positives) and a minimal amount of features that should be absent or hidden in the input to prevent the result (classification) from changing (pertinent negatives). These are found by treating the PP and PN as separate optimization problems and solving using fast iterative shrinkage-thresholding algorithm from Beck et al. [10]. This was later improved on by Mousavi et al. [68], by using the convolutional autoencoder (CAE) close to the data manifolds. The figure 3.17 show that CAE improves the results from CEM. The highlighted areas are a better representation of how humans perceive numbers. The PP results show a clear view of which pixels correspond to the classification of the numbers, whilst PN shows the smallest change needed to alter the prediction result.

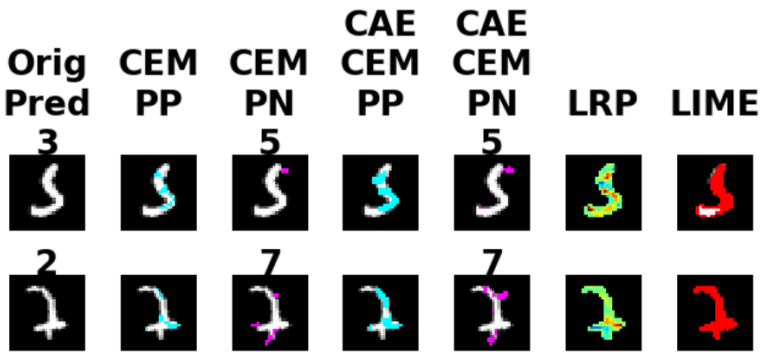


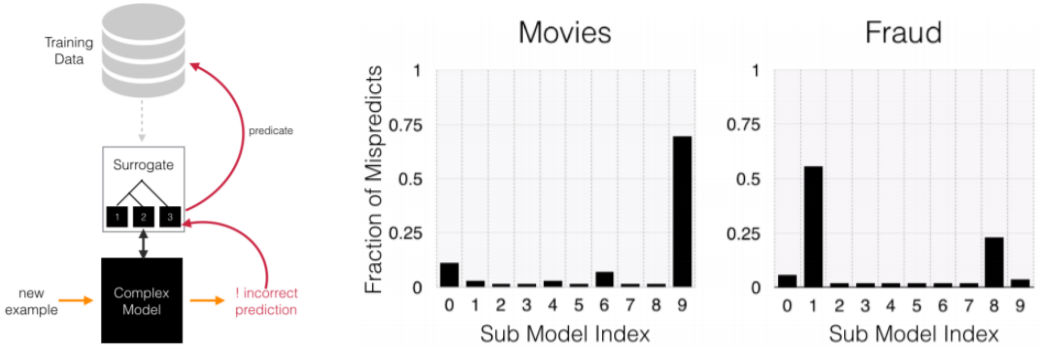
FIGURE 3.17: Explanations on the MNIST dataset. Comparsion between CEM [68] with or without CAE, LIME [93] and LRP [12]. PP/PN are respectively cyan/pink. For LRP, green is neutral, red/yellow is positive and blue is negative relevance. For LIME red is positive and white is neutral [23].

Henelius et al. [43] created an iterative algorithm to find the attributes and dependencies used by any classifier. The problem of finding groups of attributes whose interaction affect the performance is treated as an optimization problem. A greedy algorithm called *GoldenEye* algorithm was proposed to solve this using fidelity as the performance metric which effectively randomizes the attributes to select permutations, grouping the attributes, and finally pruning redundant attributes to increase fidelity. The resulting output is an increased interpretation of attribute groupings that contributed to a given prediction, e.g. {{1,2}, {3}}, for grouping of attribute 1 and 2, with singleton group 3 as high contributors.

MES (Model Explanation System) is a model-agnostic explainer from Turner [94]. The general approach is an augmentation of tools from Gelman et al. [34] of posterior predictive assessment of model fitness via realized discrepancies. MES only seek explanations for individual cases using a Monte Carlo algorithm, by deriving a scoring system to find the best explanation based on formal requirements. The explanations are derived from the interpretability of SVM (or logistic regression) models that best fit the data.

Plumb et al. [71] created a model-agnostic explainer called MAPLE(Model Agnostic suPervised Local Explanations). It provides both example-based and local explanations while also being able to detect global patterns. MAPLE combines the idea of using random forest as a method for supervised neighborhood selection for local linear modeling from SILO(Supervised Local modeling method) [14] with feature selection methods from DStump [48]. For a given point, SILO defines a local neighborhood by assigning weights to each training point based on appearance frequency in the tree. DStump defines the importance of a feature based on how much it reduces the impurity of the label when split at the root in the random forest trees. The local explanation given was suggested to be more faithful than LIME [93], and the global patterns could be uses to diagnose limitations in its local explanations.

Krishnan et al. [52] created Partition Aware Local Model (PALM), a tool that learns and summarizes the responsibility from the training examples used to train the model to aid debugging (meant for developers). Figure 3.18 shows an overview of the tool. It works by approximating the complex model using a meta-model that partitions the training data as a decision tree to allow for examination, and a set of sub-models of arbitrary complexity (could be same as black-box) to retain the accuracy that approximate the patterns within each data partition. The most informative neighborhood is the explanation for a given prediction result. This would give the developer a view of which training data was responsible for a particular sub-model.



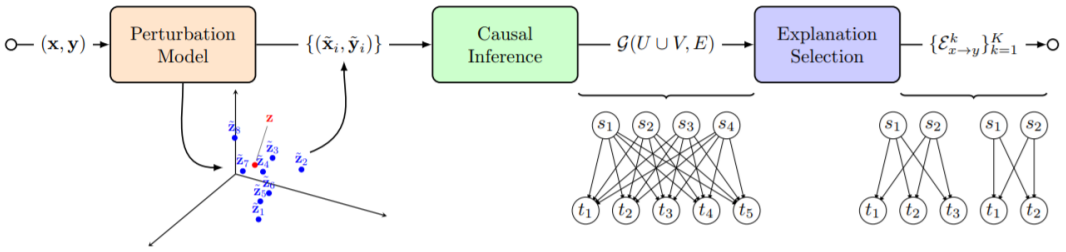
(A) Architecture (B) Dataset example

FIGURE 3.18: (A) The architecture with the surrogate models, and their connection to the training data. (B) Example from Fraud and Movies datasets, that show how mispredictions concentrate around specific sub-models, suggesting that there are specific regions of the feature-space most associated with mispredictions [52].

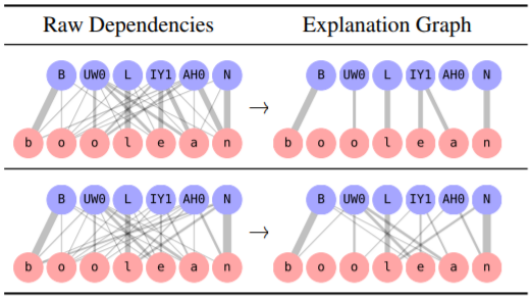
Alvarez-Melis et al. [4] created an agnostic explainer called Structured-Output Causal Rationalizer (SocRat). An explanation consists of a set of inputs and output tokens that are causally related under the black-box. An overview of the steps involved and explanation example shown in figure 3.19. It consists of 3 steps; The Perturbation Mode-step, where we perturb the black-box to generate key points. This is done using a Variational Auto-Encoder (VAE), that was extended to handle sequential data (NLP). The Causal Inference-step that infer causal dependencies between the original input and output tokens. This is achieved using a Bayesian approach to logistic regression. The final step, Explanation Selection, is done using a robust optimization technique seeking to minimize worst case cut values from Fan et al. [29], where the selection can be cast as a Mixed Integer Programming (MIP) problem.

The TREPAN algorithm from W. Craven et al. [95], approximate the model with M-of-N split points on a single decision tree. To keep interpretability at a manageable level, the tree size is limited. The explanation is a result of exploring the path in the tree.

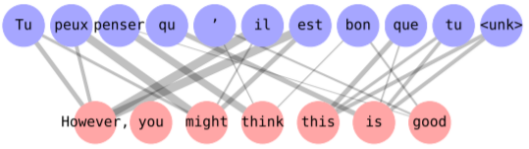
França et al. [31] looks at the problem of extracting first-order logic descriptions from neural networks trained to solve relational learning tasks. The Connectionist Inductive Logic Programming (CILP) was adapted to enable the application of a variation of TREPAN to extract first-order



(A) Schematic overview



(B) Selection

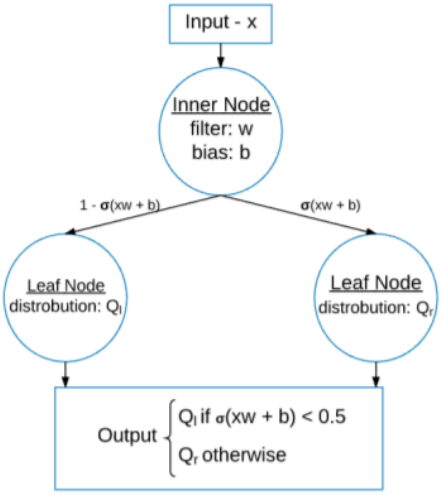


(C) Explanation example

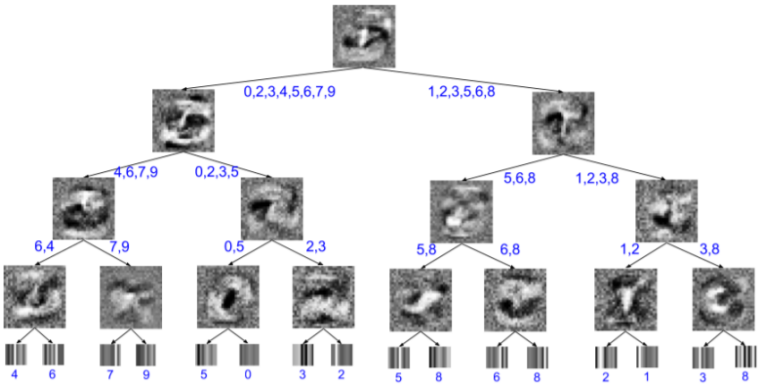
FIGURE 3.19: (A) Schematic overview of the architecture. (B) Example of going from raw dependencies in the data to selecting only the most important dependencies. The larger the edge (width), the greater the connection. (C) Explanation from a translation example [4].

logic rules into CILP++. These rules could then be used to reason about the knowledge they possess, and the choices the DNN make.

Frosst et al. [33] created a soft decision tree that generalizes better than one learned directly from training data. This was achieved by transferring the generalization abilities of a neural network to a soft decision tree (figure 3.20), training the decision tree to mimic the input-output function from the neural net. Unlike a decision tree which relies on hierarchical features, it relies on hierarchical decisions instead. The classification is based on an input example to select a particular static probability distribution over classes as its output.



(A) Binary Soft Decision Tree



(B) Explanation

FIGURE 3.20: (A) A soft binary decision tree with a single inner node and two leaf nodes. The output is the probability distribution over the classes. (B) The visualization of a soft decision tree of depth 4 trained on MNIST. The non-leaf nodes are learned filters, used to differentiate the differing inputs. The annotations at each node are the likely classification candidates. [33]

3.5 Case-Based Explanations

This section gives some instances of what has been done in terms of creating an explainable CBR system, also known as a Case-Based Explainer (CBE). The section also includes ways where CBR is used to either fully or partially explain systems.

Cunningham et al. [20] points to two possible approaches in CBE: knowledge-light and knowledge-intensive (KI) CBE. A knowledge-intensive approach may include rule-based methods which can be used to generate explanations, an example is SWALE [55]. Knowledge-light approaches, on the other hand, base their explanations on just similarity measures performed during retrieval. Although the more knowledge-intensive approach also takes advantage of similarity measures,

the explanations are expressed in terms of causal interactions rather than similarity as in a knowledge-light approach.

Although Caruana et al. [15] is a relatively early paper that touches on aspects that haven't been covered in recent years, despite AIs resurgence. They theorize that patterns that appear in the hidden units when two cases are similar can be exploited to find similarity, ultimately enabling explanations. Utilizing the weights connected to the hidden units does indeed imply the need to open the black-box, which is not where our focus is. However, considering the paper's discussion on a topic that isn't discussed much, especially with a focus on using CBR, we think it's an interesting proposal worth mentioning.

Kim et al. [50] present a new framework, namely Bayesian Case Model (BCM), which uses prototype clustering and subspace learning. Prototyping is the process of extracting the most representative exemplar from a cluster. Subspace learning is the process of only using features that are relevant in each cluster (subspace) - so-called hot features. For instance, if user A and user B are similar users, learning which action movies user A enjoys is irrelevant when we only want to know similar comedy movies for user B. Hence, BCM is a combination of Bayesian generative models and case-based reasoning. In BCM, explanations are represented as prototypes and subspaces. The algorithm will present a prototype, which is the most representative example from a cluster, along with the subspaces, which is what BCM considers to be the most important features.

MMD-critic, developed by Kim et al. [49], which is an extension of BCM, uses previous cases in the form of prototypes, much like BCM, to provide explanations. However, MMD-critic is based on the notion that prototypes alone aren't enough to provide interpretability. Instead, it learns prototypes *and* criticism. In addition to prototypes, MMD-critic selects criticism samples that are not explained well by the prototypes. Criticism can be viewed as a counter-example to the prototype of each respective cluster.

Table 3.2 shows an overview of all the papers research for the current thesis. Each paper is placed into categories as described in chapter 2.5.2. On model-agnostic vs model-specific, intrinsic vs post-hoc, and local vs global.

Name	Authors. Ref.	Year	Model-agnostic vs. model-specific	Intrinsic vs. Post-hoc	Local vs. global
ThisLooksLikeThat	Chen et al. [16]	2018	specific(DNN)	intrinsic	local
-	Li et al. [57]	2017	specific(DNN)	intrinsic	local
DBN-extraction	Tran et al. [91]	2018	specific(DBN)	post-hoc	global
-	Lei et al. [56]	2016	specific	intrinsic	local
NIT	Tsang et al. [92]	2018	specific(NN)	intrinsic	global
SENN	Alvarez Melis et al. [3]	2018	specific(DNN)	intrinsic	local/global
RNN-HMM	Krakovna et al. [51]	2016	specific	post-hoc/intrinsic	local/global
1Rule	Malioutov et al. [61]	2017	specific	intrinsic	global
GA^2M	Lou et al. [59]	2013	specific(GAM)	intrinsic	global
Decision Sets	Lakkaraju et al. [53]	2016	specific(data)	intrinsic	global
-	Sarker et al. [80]	2017	specific(DNN)	post-hoc	local/global
DeepRED	Zilke et al. [98]	2016	specific(DNN)	post-hoc	local/global
LORE	Guidotti et al. [40]	2018	agnostic	post-hoc	local+
CEM	Dhurandhar et al. [23]	2018	agnostic	post-hoc	local
GoldenEye	Henelius et al. [43]	2014	agnostic	post-hoc	local/global
MES	Turner [94]	2016	agnostic	post-hoc	local
MAPLE	Plumb et al. [71]	2018	agnostic	post-hoc	local/global
PALM	Krishnan et al. [52]	2017	agnostic	post-hoc	local/global
SocRat	Alvarez-Melis et al. [4]	2017	agnostic	post-hoc	local/global
TREPAN	W. Craven et al. [95]	1999	agnostic	post-hoc	global
CILP++	França et al. [31]	2015	specific	post-hoc	local/global
SoftDT	Frosst et al. [33]	2017	specific	intrinsic/post-hoc	local/global
SWALE	Leake [55]	1995	-	intrinsic	local/global
CBE for non-CBR methods	Caruana et al. [15]	1999	specific	post-hoc	local
BCM	Kim et al. [50]	2014	specific	post-hoc	local
BCM+	Doshi-Velez et al. [25]	2017	specific	post-hoc	local

TABLE 3.2: Overview of XAI papers researched for this thesis. Some have no clear distinction between the categories, and as such contain both

Chapter 4

Architecture

This chapter begins by summarizing the proposed architecture from the specialization project. Chapter 4.2 looks at XAI and CBR both separately and in tandem related to designing an architecture. Finally, we present the final general architecture in 4.3.

4.1 Summary of Proposed Architecture

Through our work with the specialization project we proposed an architecture for a XAI system. To the best of our knowledge, there are no existing implementations of what we seek to do. Nonetheless, we proposed a case-based explainer which encapsulates the DL classifier. A sketch of the previously proposed architecture is shown in figure 4.1.

Our initial idea was that the DL classifier would be fed with a labeled dataset to perform supervised learning. After training the model, we would then be able to use one of the model induction techniques which utilizes the gradients to compute weights. The weights would then need to be converted to some other form of representation which could be presented to the user as a potential explanation, e.g. by presenting the most significant features. The specifics on how to convert said weights to a more human interpretable representation was left for the master thesis. The purpose of the CBE module was to redefine CBR's notion of "similar problems have similar solutions" to "similar problems have similar explanations". The idea was that, as the case-base grew, we could reuse or adapt a case whenever queried, reusing the information used to recreate the explanation from a previous case on a new problem. As we keep using the system, the more information we will have available to generate an explanation. Depending on how this information is obtained, we might be able to derive some useful general knowledge (i.e. knowledge-intensive CBR from section 2.3).

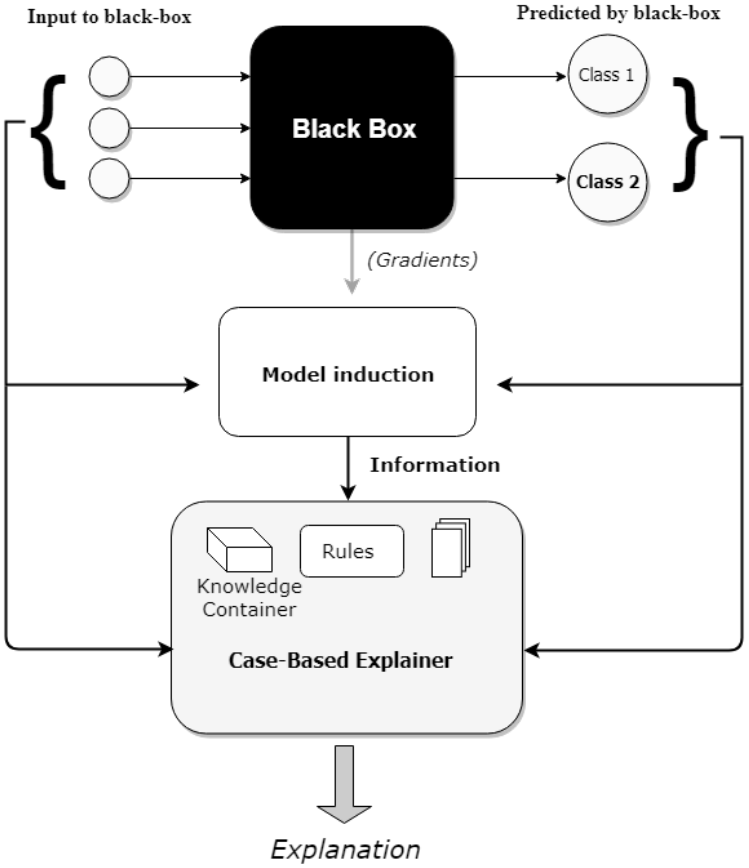


FIGURE 4.1: Both input and prediction (class 2 here) made by the black-box model is passed to both the model induction processor and CBE [27]

Upon further research, we found the approach of utilizing only weights to be lacking in explanatory abilities to the user. Because of this, we instead opted for a more knowledge-rich approach with generated rules.

4.2 Discussion on XAI and CBR

There is some criticism as to what is required to truly explain a black-box, and whether what has been proposed in previous research can even be classified as explanations [24]. Rudin [79] suggest that trying to explain a black-box system is in itself a bad idea. The approach of inducting the behaviour of the network with more transparent or interpretable models are said to not be faithful to the functionality of the original system. The fidelity of the system might get low and the approach of using the black-box as an oracle and training a transparent model as the explainer to draw explanations from the black-box can not have perfect fidelity with respect to the original model. Ultimately the criticism lies in using models that are not already somewhat interpretable, and that the accuracy-interpretability trade-off where we chose models that give better accuracy and decrease interpretability, which is an idea that often appears in XAI papers,

is said to be a myth [24]. Complex models are not actually required for achieving high accuracy, but instead easier to work with as a result of being more developed than interpretable models.

Alvarez-Melis et al. [5] used the metric of robustness using local Lipschitz-constants on interpretability methods (LIME, SHAP, etc.), where robustness is a measure of variation to the input with respect to the explanation (attribution) generated, where similar input-output pairs should give rise to similar explanations. Experiments showed that model-agnostic perturbation based methods were more prone to instability than gradient-based interpretability methods. Both methods however performed for the most part poorly on the robustness metric. Alvarez-Melis et al. hypothesize that since the underlying model itself is not robust, the interpretability method suffers the same fate. However, it's not clear whether robustness is an essential property to uphold as it only takes into account the sensitivity of the explanations. These gradient-based methods are also potentially vulnerable against adversarial attacks, as was demonstrated by Ghorbani et al. [35] where small random perturbation to the input can change the feature importance, and even give rise to drastically different interpretations without changing the prediction.

Yeh et al. [96] formulated the metric of sensitivity. Sensitivity is the characterisation of how an explanation changes with varied input. Results show that the less sensitive the interpretation method is, the more faithful the explanation otputted turns out to be. Yeh et al. also shown how we could optimize the black-box model with respect to this property using adversarial training to reduce its sensitivity.

To achieve truly explainable AI, Doran et al. [24] suggest combining a comprehensible model with a reasoning engine. A comprehensible model is explained as a system that emits symbols along with its output to allow the user to relate properties of the inputs to their output. The figure 4.2 shows that this would include a black-box that provides more information than only the output, where local interpretation in symbolic form is used by the reasoner along with a knowledge-base to explain a given prediction.

A knowledge-intensive case-based reasoner would be a system able to fulfill the part of the reasoner in this system, with the inclusion of feedback and adaptation of explanations. The comprehensible part would need to be acquired in another way, but we hope that any model induction technique, more specifically local interpretation that derive symbolic information or what could be considered to be general knowledge from the data, will suffice as a first step.

Most philosophers, psychologists and cognitive scientists in the field suggest that all "why"-questions are in fact contrastive. As such, an explanation engine would need to be able to differentiate cases to be able to give these type of explanations, resulting in a more human-interpretable system [64]. Some of the methods discussed do generate contrastive explanations [23] [40]. However, these type of contrastive explanations are only a local suggestion for changing

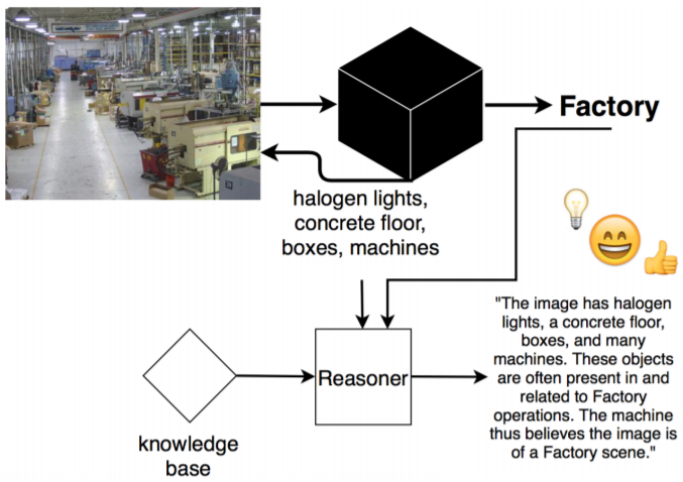


FIGURE 4.2: Augmenting comprehensible models with a reasoning engine, from a image scene classification problem [24]

the input of the model to achieve a change for a single prediction instance, and as such cannot give a global overview of the model.

When it comes to CBR, there are mainly two approaches that are used. **Instance** based, and **general knowledge** based, as shown along the knowledge-intensity dimension in section 2.4. With *instance* based, each case is a single instance, where a similar case solution can be used with some adaptation to solve for a given problem instance. For *general knowledge* based, we can have single instances with some form of general external knowledge [81], or general cases as from Gu [38], where the general knowledge required to solve the problem, hopefully, is encapsulated by the system. Leake [55] mention that in a case-based model of explanations, that goals and experiences play a key role in explanation generation in order to guide the process in domains that are complex and imperfectly understood. The benefits of starting from previous relevant explanations, such as in CBR, are better candidate explanations that are supported by previous explanations, more effective explanation generation than when starting from scratch, and a more precise focus toward useful explanation goals and needs. This could imply that a generalized explainer would need to be able to utilize recollected instances, instead of relying on pure general knowledge, resulting in higher explanatory power from the system.

Another problem described in Leake [55] is that many explanation methods do not consider previous instances when creating an explanation for the input prediction, which limits the system in terms of their understanding of a domain. If new evidence were to disprove previously given explanations, we would have to change our (the system's) underlying understanding of the topic. This is in itself impossible unless we have a recollection of what has been explained in the past. This could be categorized as learning over time, as the explanations given improves over time. This learning could be further improved by incorporating feedback from the users of the system. If an explanation does not satisfy the user's demands, or is inconsistent with the user's

domain/experts beliefs, the system could keep track of these inconsistencies in its reasoning. As the systems own experience could be "lacking" in some areas. Supported by the meta-cognitive (knowing about knowing) CBR system by Craw et al. [19], where the CBR system itself can learn to recognize it's own faults, with curiosity about its domain. Such a system could improve upon the explanatory capabilities of the common CBR system. A big problem to the knowledge-based approach is providing the CBR system with this general knowledge. This is called the *knowledge acquisition bottleneck* [36], and is usually done by consulting domain experts. We hope to avoid this problem, by utilizing *active learning* [77], which is refereed to as query learning by Angluin [7], where the system itself is able to interactively query an information source to obtain the desired information to learn from. In our system this could be the oracle or model (black-box) we want to explain, along with different model-induction techniques, by querying it with varying inputs of importance which it can learn from, where the model-induction techniques each give a different perspective of the black-box, which can be used to generate a final explanation by the CBR system and expand upon the knowledge-base of the CBR system as a whole.

Even though many papers discussing CBE brings forth the argument of *only* presenting the nearest neighbor as a sufficient explanation, there are some challenges related to this. As discussed in Nugent et al. [69], although CBR offers some transparency, there is some knowledge hidden inside the knowledge containers which are not apparent to the user. McSherry [62] argues that presenting the feature values in the most similar cases may be misleading. In some cases, the presence of some feature values may be against the prediction, just presenting these to the user may not always be as useful. This speaks in favor of finding and highlighting features that are important factors when designing a successful case-based explanation, also pointed out by Nugent et al. [70].

4.3 Final Architecture

In this section we present and discuss the final architecture based on everything discussed in previous sections, namely combining model-agnostic methods from previous relevant research together with the CBR methodology. Later we will implement a select part of this, and run it on a selected dataset and network model.

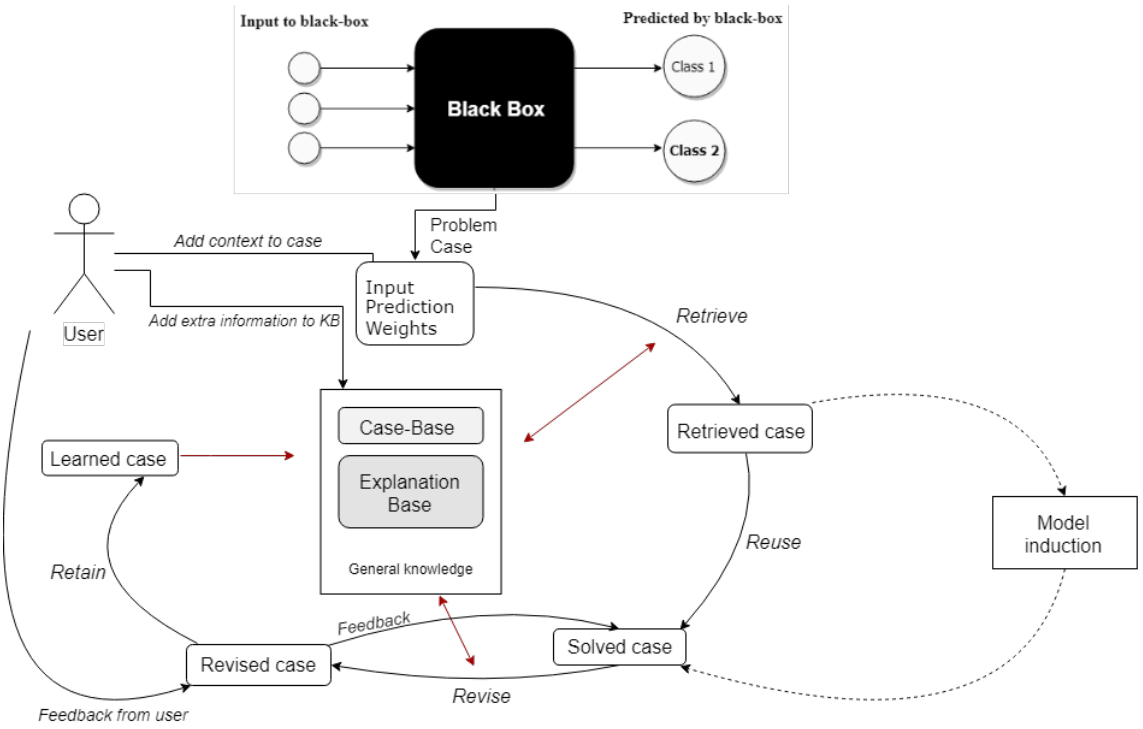


FIGURE 4.3: Overview of a modified CBR cycle for a general architecture.

Figure 4.3 shows an overview of the whole CBR cycle. Each step will be described in detail later in this chapter. First, the black-box takes an input instance, and give a prediction on this, along with attribution values from the black-box. This is combined to form the target problem case. If the user wants a specific explanation related to this, he can give some added information related to what kind of explanation is required to satisfy his demands. If there is more information regarding the case itself, it should also be given, to make it easier for the CBR system to find/create a good explanation. This case is now compared against the Case-Base for cases in the past that are similar (retrieve). If we find a previous case with a explanation, with the given context, we can utilize this again (reuse). If we didn't find a sufficiently similar case to reuse or empty Case-Base, we need to use our model-induction techniques to get the related general knowledge of this case. The next step is to generate a good explanation from all this information on the problem case. We can use the contextual information related to what particular question is being asked from the user as a guide. One kind of user might prefer a very detailed description of previously similar cases (if available), and a lot of the available inducted information. If the user is not happy with the explanation presented, the system need to keep adding/removing extra information (revise the explanation). Once a problem is sufficiently explained, it can be stored in the Case-Base (retain), along with what information was used to generate the explanation and how the explanation-knowledge was adapted to fit the user (from adaptation container), and the explanation presented to the user.

4.3.1 Context and Extra Information

An important feature in CBR is the ability to include extra information related to the domain. This can also increase the quality of the explanations of the CBR system [55]. Extra information could be some context related to the explanation goal, like the type of user which is requesting the explanation or some constraints on the system, like the maximum amount of time given to compute the explanation (exploitation vs. exploration). The challenge with this is how to include this kind of extra information without manually implementing every part ourselves for the different domains. Particularly the two types of contexts, context related to the problem instance, and the user context (goal of the explained).

There may be various information contained inside a case's context attribute. In the medical domain, it may be relevant taking into account whether the doctor is in a doctor's office or in an ambulance. A doctor utilizing the explainer system in an ambulance probably has less time than a doctor in the doctor's office as he's probably treating more critical patients. The system can be designed in such a way that it's able to present a more simplified version by using fewer resources, hence requiring less time.

Another instance of extra information is related to the problem after it has been solved. In our example, we might like to know the status of a patient several weeks after the treatment, in case of any complications which occurred afterward. The system could in this instance be curious to know what would have been the correct treatment, and maybe even why. This information could come in handy at a later time when the black-box suggests imperfect solutions that are not trusted by the user in regards to the problem.

4.3.2 Feedback

In order for the system to continue to learn, it should handle feedback from the different users. Stahl [85] presents feedback between each stage in the CBR cycle as an important extension required to realize advanced learning approaches, where the user can give feedback to each step in the CBR cycle to improve each element individually. Feedback on the most important stages in the CBR cycle should be implemented. One might give clues regarding the system's performance, or its understanding of the underlying domain. Particularly with context, learning when different contexts apply to different users and different situations would allow the system to answer a broad range of questions a user might have.

Since the black-box will undoubtedly give the wrong prediction from time to time, it would be helpful for the users in this instance to be given clues as to when this happens, as to not give false explanations. Feedback on the solution itself from the black-box could help in this regard, if we know that there is a high possibility of a mistaken proposal, it should be noted to the user,

and why the system thinks this. If the system can differentiate these situations, it could further increase its degree of trust to the user. E.g. we know that the instance we are predicting could be an edge-case, if the model-induction methods fails to find any clear representation, which again points to problems in the underlying black-box, or model-induction method. The system could even go so far as to alter the prediction itself in these instances, by utilizing the solutions of previous cases with a different prediction, as we know that the current prediction is most likely wrong.

Feedback is generally not easy to acquire, as it can be delayed or never given [54]. However in an explanatory setting, we think that it would be easier to collect this, as the process of generating explanations can involve the user to a greater extent.

4.3.3 Case Representation

The case-base is the collection of cases, with or without an explanation. These are used to capture the broad aspect of the case-base. Depending on the similarity measure used, we want to find a previous experience similar to the current experience to explain a very similar problem and utilise the same steps used to generate the explanation.

The case part of the CBR system, usually consists of a problem description part, and a solution description part, which is the solution to the specific problem in the case. In this system, the problem description part is the input instance to the black box, the prediction on this and the attribution weights from the black-box. The problem description part should also contain extra information, as context to the specific problem, that the black-box does not know about. The solution description part is instead the explanation description.

The attribution value can be generated using a multitude of algorithms, as previously mentioned in section 3.1. If we want to remain completely model-agnostic (SHAP, LIME, etc.) or more toward DNN specific (Integrated Gradients etc), there is also the time factor to consider. Methods that specifically target DNNs with internal gradients are faster than those that rely purely on permutations. As the complexity of the DNN model increases this will be further noticeable. There is also the sensitivity/robustness metric as previously discussed in section 4.2, where permutations generally perform worse on these metrics.

The solution part, on the other hand, is a bit more difficult to quantify. We do not want to rely only on the instance-based approach where each case has its own solution. Therefore we've decided to keep pointers to a *explanation-base* that solved the particular problem case, as this can be multiple, along with which cases were used.

This also allows us to recall which cases were used to generate the specific explanation knowledge as added information (provenance). The addition of provenance can be very useful in guiding maintenance on both the case-base and explanation-base later, as noted by Leake et al. [54].

The particular explanation presented to the user should also be included, as this can be helpful for reviewing the case-base later. Finally, which adaptation techniques were used to go from explanation-knowledge to explanation should also be noted in a case, as every step towards generating an explanation should be as transparent as possible. Direct adaptation on the explanation presented from an earlier case is however difficult and should be avoided, as it can lead to false explanations.

4.3.4 Explanation-Base

The explanation-base will be responsible for containing the explanation knowledge extracted from the case-base instances using model-induction techniques previously mentioned. As the techniques provide varying degrees of general knowledge, we need to utilize methods that provide some local-concise knowledge in the form of interpretable rules.

Any knowledge added by a domain expert *after* the prediction can't be guaranteed to comply with any patterns found by the neural network. Thus, using rules which are inserted into the explanation-base by a domain expert won't *necessarily* give the right explanation of a prediction. However, experts can be useful to check whether some rule generated by the system isn't making sense with what the expert already knows about the domain. This way, the experts' knowledge can be useful to maintain the case-base, to point out cases which we definitely know to be wrong. One can then decide what is the best action in order to correct cases in the case-base which are known to have based their explanation on the wrong case's explanation.

```
{
    "id-0": {
        "method": "Anchor",
        "explanation":["18<Age≤30" AND "Blood_Pressure>139" AND ...],
        "precision": [80%, 92%, ...],
        "coverage":  [40%, 10%, ...],
        "prediction": "Threatment_A",
        "provenance": Case_id
    },
    "id-1": {
        "method": "LORE",
        "explanation":["Age<30" AND "Blood_Pressure>139" AND ...],
        "prediction": "Treatment_A",
```

```
        "counter_factual":[
            {
            "values":["Age>30" AND "Blood_Pressure>139" AND ...],
            "prediction":"Treatment_B"
            },
            ...
        ],
        "provenance": Case_id
    },
    "id-2": {
        "method": "LIME",
        "features": ["Age=52", "Blood_Pressure=124", etc],
        "attribution":[0.30, 0.15,-0.6],
        "prediction": "Treatment_B",
        "provenance": Case_id
    },
    "id-3":{
        "method": "Integrated_Gradients",
        "features": ["Age=20", "Blood_Pressure=124", etc],
        "attribution":[0.11, 0.45, -0.5],
        "prediction": "Treatment_D",
        "provenance": Case_id
    },
    "id-4":{
        "method": "Expert(User)",
        "explanation": ["18<Age≤30" OR "129<Blood_Pressure≤139"],
        "prediction": "Treatment_B",
        "provenance": User_id
    }
    ...
}
```

LISTING 4.1: Proposed examples of different explanations from varying model induction techniques in JSON format

Listing 4.1 show an example of how different explanations from different model induction techniques could be stored. Anchor, LORE, LIME, user(expert) and Integrated Gradients in this case. Each explanation is part of the explanation-base, and can be used to explain a new instance in the future, given the provenance case (case that was used to create this knowledge instance) is similar enough that they fit. The first number is the ID of the explanation, that

is used to point to the explanation-base from the case, each with its own explanation method representation. The "provenance" case_id is the ID of a case in the case-base. The first method, Anchor, has precision values associated with it, which is a partial precision value of a partial anchor. Precision is defined as: *of all the instances in the dataset D, where the set of predicates of anchor A apply, how many share the same prediction?*. Coverage is how "wide" the anchor can be applied to the perturbation distribution. *How large percentage of the whole dataset D can it be applied to.* The explanation from LORE is very similar, but with a counterfactual explanation as well, which is the closest permutation on the instance to change the prediction. In this instance it is changing the Age parameter, that result in another treatment being proposed. The explanation from LIME is very similar to Integrated Gradients, but instead of using the internal gradients of the model itself, LIME uses only permutations on the model to find these values. Finally, we got the experts own knowledge, this could be a simple rule that the expert knows apply to the given domain. With a user id, to keep track of who added this rule.

Explanation-Base Maintenance

As the explanation-base continues to grow, it will be harder to keep track of everything the system think it knows. It will undoubtedly begin to fill with overlapping explanations. E.g. two very similar explanation with only minor differences in the *Age*, could be merged together to form only a single explanation to save some space.

In another instance, we could have conflicting explanations, in this case, we would need to take note of potential problems in the explanation-base. As we wouldn't know which of the explanations could be "correct" without feedback from an expert. This goes back to the concept of fidelity and to what degree the explanation conforms to the knowledge hidden in the black-box.

Case-Base Maintenance

Within case-base maintenance, we find some problems, namely the swamping problem, whereas the number of cases in the case-base increases over time, the cost of searching the case-base increases. As with the explanation-base, the case-base would need to be kept to a minimum. To mitigate this problem, a case-deletion strategy should be performed, with the sole purpose of removing cases which doesn't help us cover more "knowledge" ground (competence). We might also delete cases that might have unnecessary repetition. Smyth et al. [84] attempted to solve this with a deletion strategy that target cases that aren't crucial for the performance of the CBR system. A similar strategy could be implemented, by keeping track of how many times a case have been used, and the similarity between the cases in the case-base. Where cases that are the least similar to the rest of the case-base, are probably unique cases.

4.3.5 Knowledge-Containers

There are four knowledge-containers in CBR: vocabulary-, similarity measure-, adaptation- and case-base container.

The vocabulary container contains the usual concepts that are needed for the system to explain the domain. The similarity measure contains knowledge about how to retrieve against the different cases, but it also need to handle the extra contextual information and the added attribution weightings. In the solution transformation, or adaptation container, we will have to store how we adapted the given knowledge-instances for a specific user. Then there is the matter of presenting this knowledge to the user. This could be as simple as noting which combination of information receive good feedback, or a more complex adaptation that presents how the specific explanation generated came about as well, and other statistically significant information that can be found from the case-base that might be worth presenting. The case-base contains cases that are fully solved or partially solved. Partially solved means that although an explanation was generated, depending on the domain, the case cannot be labeled as "solved" as it may be dependent on some feedback that comes at a later stage. Note that "solved" in this context means an explanation was generated which also satisfied the expectation of the user.

In our system, we also introduce a new container, as part of the knowledge-base - the explanation-base container. It contains the different explanation-knowledge elements extracted from the case-base, model-induction methods or the user.

4.3.6 CBR Cycle

This section presents a detailed explanation on the different steps in the CBR cycle as presented in the architecture figure (figure 4.3).

Retrieve

In CBR, the retrieval step is probably the most important step in the cycle, as every step afterward is dependent on this step. It doesn't matter if the case-base contains the solution case to a new problem if it can't find the relevant solution in the first place.

In the retrieval step, we want to find previous instances with similar explanations to explain the new problem. The difficult part of the retrieve step is creating a good similarity measure to find similar problems. If the system cannot find a similar case that is sufficiently similar, we need to break out of the traditional CBR cycle and query the system's model induction method. This could, for instance, be LIME, LORE and/or Anchor. This is an indication that the case is unique in the sense that no similar case is present, and hence, our case-base will be prepared for

a similar case in the future. It's also important to keep the similarity function simple enough to be interpreted by the users. Otherwise, we would have difficulty fully explaining the retrieval step.

To improve upon the retrieval step, we suggest utilizing ideas from Caruana et al. [15], that use internal weights (activation pattern) for better similarity against other cases. However, this approach "opens" up and looks at the internal workings of the black-box(DNN). We think it might be better to keep it closed. Nevertheless, we can use part of the patterns, namely the attribution weights. As have been mentioned in previous sections, there are a numerous ways to find a good attribution weighting, either as a closed system, or an open system. We hypothesis that this would improve the retrieval step to find a similar case from the case-base.

Another factor that should also be considered is utilizing the additional contextual information to find cases that better match the current situation at hand, as the surroundings of a given instance could give better retrieval capabilities. If we know two cases has the contextual information of e.g. being in an ambulance, we know that their similarity is closer than another case in the hospital.

Reuse

Assuming we have found a sufficiently similar case in the case-base, we can utilize the general knowledge links in the explanation part of the case. We need to do the necessary adaptation to fit the new problem. This requires us to be careful in making sure that the explanation has a high degree of fidelity with respect to the problem-case and the black-box model. The general knowledge links are tested on the new problem, as we should only use parts that fit the problem instance.

If we didn't actually find any relevant cases or failed to fit the explanation part, we need to ask the model induction algorithms directly. This will give us some new general knowledge regarding the new problem.

When we have some general knowledge regarding the current problem, we can begin creating the explanation that will be presented to the user. The context from a target case can be used to further adapt the explanation towards this goal. Say we know the context of a previous problem solution case, such as the availability of some specific medicine or a time constraint to generate an explanation: the explanation for giving this medicine, could then be the that the optimal medicine wasn't available in the ambulance at that moment and that the patient was in dire need of stability.

Revise

To verify if the explanation satisfies the user-need, feedback is asked from the system. If the explanation does not meet the expectations of the user, alas he/she couldn't justify the prediction proposal with the presented explanation, the system would need to alter the explanation by adding more information regarding the model. If the system cannot give a good enough explanation, this problem case should be noted as a problem area that would need to be improved.

Checking the clarity of the explanation requires feedback, and checking to which degree the explanation agrees with the input-output mapping of the system requires domain knowledge or feedback from an expert.

Feedback could be used to verify if the solution fit the goals or question related to the explanation from the system. One type of user could be a doctor or a medical student, each requiring a different degree of explanation details to trust the system.

Retain

The goal of learning is to turn the newly solved case into a new experience by constructing a new case and/or modifying parts of the explanation-base to be able to handle new similar problems that may appear in the future. The cases stored in the case-base will be part of the learning along with the corresponding knowledge in the explanation-base used to generate an explanation onto the specific case, similar cases used, how the explanation was generated with respect to the user context and other related information.

The retain step is very closely linked with the maintenance of the system. In this step, the maintenance of the explanation-base and the case-base should be run. This is to decide whether it is actually worth storing a new problem case or instead only keeping parts of what has been learned is sufficient.

The CBR framework performs multiple induction techniques to extract information about the DNN. This information is analyzed and treated as explanation-knowledge which is used to explain a given query to the DNN. With every new case added to the case-base, we need to figure out if any of the previous cases are related, and update older cases that happen to fit the same explanation pattern in the explanation-base.

4.3.7 Explanation Generation

To generate explanations from the local interpretation knowledge extraction methods, we need to present explanations with regard to a specific input case to explain the prediction of the black-box.

Explanation representation is also very important, as the representation will affect the interpretability of the explanation given to the user. If the user cannot understand or relate to the explanation, the system has failed to explain the problem. Lay users, often prefer natural language representation of an explanation, and as such, using attributions or previous instances may be unproductive. For more knowledgeable users, these might be perfectly valid explanation representations.

Like the explanation in figure 4.2, the explanation could incorporate elements of uncertainty within the systems beliefs. E.g. the explanation could be based on statistical information gathered from the case-base. Including the possibility of the prediction being wrong or a statistically strong predictions. Like the inclusion of odds ratios, to support a prediction as "prediction A was 2 times more likely than B".

The nature of CBR, also allow raw input features to be used for explaining, where these are said to be the natural basis for interpretability, when the input is low-dimensional and individual features are meaningful. In high-dimensions, raw features such as pixels in an image is very hard to analyze, and often lead to unstable explanations that are prone to noise in the data [3].

An explanation example is given in the listing 4.2, that show the inclusion of multiple parts in the explanation. The prediction from the black-box, an explanation from one of the induction techniques, similar problems in the past with its explanation, and whether or not there are any conflicting records in the case-base that suggest different predictions.

```
Since the patient is in the age group a < x_1 < b and the heart rate is x_2, ...
The model suggests giving treatment Y.
This is because of the high blood pressure (higher than c), and above average liver function tests.

A similar problem from the case-base in the past: C_23,
    with the patient information ... given treatment Y, the patient recovered in 2 weeks.
    The treatment was approved by Person P.
    Explanation for the previous case was: (...)

There are no conflicting records from any of the X number of cases.
```

Listing 4.2: Example of explanation presented to the user

Chapter 5

Data, Implementation and Running Example

This chapter first presents the process of finding a dataset a give a description of the data. In the next section, we describe the DNN which was trained to be explained. In 5.3 we describe our implementation, which is a limited version of the architecture described earlier in chapter 4. Finally, in 5.4, we show a running example.

5.1 Dataset

Before starting the implementation, we did several rounds in our faculty building (IDI) to find researchers that may have datasets and/or DL models that could be interesting for our thesis. Unfortunately, few people seemed to have a trained model available. The few who did have datasets at hand were too complex to be understood or the dataset was image-based, a domain we had already ruled out. Most people kept pointing to the UCI repository for a dataset that would fit our need, which may be the closest unofficial collection of standard AI datasets.

After a quick review of the available datasets, we ultimately landed on a dataset which has already been utilized in many XAI papers before ([93], [40], [75] among others): the Adult-dataset is from the UCI collection [26]. It's a dataset that is not very well completed, in terms of top accuracy score, with a top-score less than 90% [1].

The dataset is a simple classification problem, with 12 attributes describing a given person (age, education, etc.) and the goal is to classify whether or not the salary exceeds 50.000 USD. It consisted of a total of 48K labeled instances, split between a training and testing set. The test

[1] `https://www.openml.org/t/7592`

set was further split in half forming a validation set, resulting in approximately 32K examples in training, and 8K in test and validation, respectively.

Before we could work with the dataset it needed to be pre-processed. This step consisted of trimming attributes that were deemed unnecessary and simplifying the values for the DNN to begin training. The pre-possessing steps and implementation were based on Ribeiro et al. [75], as we needed some key points from the dataset for the Anchor framework. The implementation code contains some minor modifications to work on updated software packages, and with a different discretizer. All of which can all be found in the source code of this project [28].

One important pre-processing requirement for utilizing the Anchor framework is *discretization*. In DNN it is the step of processing the features in the dataset that are continuous into discrete values. As this is a requirement by the model agnostic method, namely anchor, to successfully generate simple concise rules without having to deal with an infinite number of possibilities, we simply do this beforehand and treat each feature as a category. These discrete feature values need to be utilized late when we want to query with new data instances.

One problem with discretization is that is is difficult to know which method is best for a given dataset. E.g. say we want to predict the risk of Alzheimer: it would not be very helpful to split the data at early age groups, say 15 and 30. Bins for age x: $[15 \leq x,\ 15 < x \leq 29,\ 29 < x]$. This would result in a giant bin for 30+, where most Alzheimer patients are. To combat these types of issues, we need to make meaningful splits. One good method is the entropy-based discretization [63], which try to maximize how much a split match up with the classifier labels, i.e. generate a decision tree that best split the dataset with respect to information gain (entropy) on a particular continuous feature. These splits are later the bins used to discretize.

The attributes of a given person after discretization is as follows in listing 5.1. Each attribute is placed into a selection of categories, and the continuous variables into bins.

```
[age]: continuous → [age ≤ 21.50, 21.50 < age ≤ 23.50, 23.50 < age ≤ 24.50,
    24.50 < age ≤ 27.50, 27.50 < age ≤ 29.50, 29.50 < age ≤ 35.50,
    35.50 < age ≤ 61.50,age > 61.50]
[workclass]: [Federal-gov, Local-gov, Private, Self-emp-inc, Self-emp-not-inc,
    State-gov, Without-pay]
[fnlgwg]: Omitted. The number of people the census takers believes
    the entry represents (weighting).
[education]: [Associates, Bachelors, Doctorate, Dropout, High School grad,
    Masters, Prof-School]
[education-num]: Omitted (duplication of education).
[marital-status]: [Married, Never-Married, Separated, Widowed]
[occupation]: [Admin, Blue-Collar, Military, Other, Professional, Sales,
    Service, White-Collar]
[relationship]: [Husband, Not-in-family, Other-relative, Own-child, Unmarried, Wife]
[race]: [Amer-Indian-Eskimo, Asian-Pac-Islander, Black, Other, White]
[sex]: [Female, Male]
[capital-gain]: continuous → [High, Low, None]
[capital-loss]: continuous → [High, Low, None]
```

```
[hours-per-week]: continuous → [hours per week ≤ 31.50,
    31.50 < hours per week≤ 34.50,34.50 < hours per week ≤ 39.50,
    39.50 < hours per week ≤ 41.50,41.50 < hours per week ≤ 46.50,
    46.50 < hours per week ≤ 49.50, 49.50 < hours per week ≤ 65.50,
    hours per week > 65.50]
[native-country]: [British-Commonwealth, China, Euro-east, Euro-south,
Euro-west, Latin-America, Other, SE-Asia, South-America, United-States]
```

LISTING 5.1: Categories of the features in Adult dataset after preprocessing

For the prediction, we have the simple encoding of class 0 for prediction of less than 50000 USD, and class 1 for more than 50000 USD.

5.2 DNN

We needed a black-box model to explain. The DNN is made using a high-level neural network API named Keras[17], which makes it easy to implements the network and corresponding functionally on the network (training, saving/loading, prediction).

As the dataset which we were set to train on is a challenge for a DNN, we had to utilize a lot of different regularization techniques and architectures combinations for it to produce good results. During training, smaller networks were able to achieve approximately similar results. However, the training was less stable, which is why we went for a deeper approach.

```
Adam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.01, amsgrad=True)
# Sequential keras model
Dense(512, input_dim=input_dim,
    activation="relu",activity_regularizer=l2(l=0.001),
    bias_regularizer=l1_l2(l2=0.001,l1=0.001)), Dropout(0.6),
Dense(256, input_dim=input_dim,
    activation="relu",activity_regularizer=l2(l=0.001),
    bias_regularizer=l1_l2(l2=0.001,l1=0.001)), Dropout(0.5),
Dense(128, input_dim=input_dim,
    activation="relu",activity_regularizer=l2(l=0.001),
    bias_regularizer=l1_l2(l2=0.001,l1=0.001)), Dropout(0.4),
Dense(96, input_dim=input_dim,
    activation="relu",activity_regularizer=l2(l=0.001),
    bias_regularizer=l1_l2(l2=0.001,l1=0.001)), Dropout(0.3),
Dense(64, input_dim=input_dim, activation="relu",
    bias_regularizer=l1_l2(l2=0.001,l1=0.001)), Dropout(0.1),
Dense(32, input_dim=input_dim,
    activation="relu",bias_regularizer=l1_l2(l2=0.001,l1=0.001)), Dropout(0.1),
Dense(16, input_dim=input_dim,
    activation="relu",bias_regularizer=l1_l2(l2=0.001,l1=0.001)),
Dense(8, input_dim=input_dim,
    activation="relu",bias_regularizer=l1_l2(l2=0.001,l1=0.001)),
```

```
15 Dense(output_dim, activation="sigmoid") # output layer
```

LISTING 5.2: Keras network settings

From listing 5.2, the *input_dim* is a simple reference to the size(length) of the input data. In our case this is set to 72, with every feature is a one-hot-vector encoded with respect to each value. We list the parameters that were chosen for the optimizer, Adam, with its standard parameters. Each line of *Dense*, represent the parameters settings for one dense layer in the network. Beginning with the width of the layer, and activation function set to the Rectified Linear Unit (relu) function $f(x) = max(0, x)$. The first layer is, in this case, a Dense layer with 512 nodes, and 72 input nodes. The regularization of each layer is set to a combination of l2(ridge) and l1(lasso) regularization on differing weights, which penalizes specific weights in the network with respect to loss. After each layer, it follows a dropout regularization layer, which simply deactivates random inputs during training. This works by training different numbers of architectures in parallel, which help with generalizing the DNN. The final Dense layer is the output layer, which is simply the number of classes in the system. In our Boolean classification problem $output\_dim = 1$.

5.2.1 Results After Training the DNN

With a final score of 85.3% accuracy on the test set, this was the final model architecture on the DNN that performed best from our experiments. Figure 5.1 shows how the model accuracy peaks at approx epoch 48, resulting in it being the final model weights stored aside and used. The ROC (Reciever Operating Characteristics) curve, with a score of 1 equaling a perfect score, in terms of separability between the prediction classes.

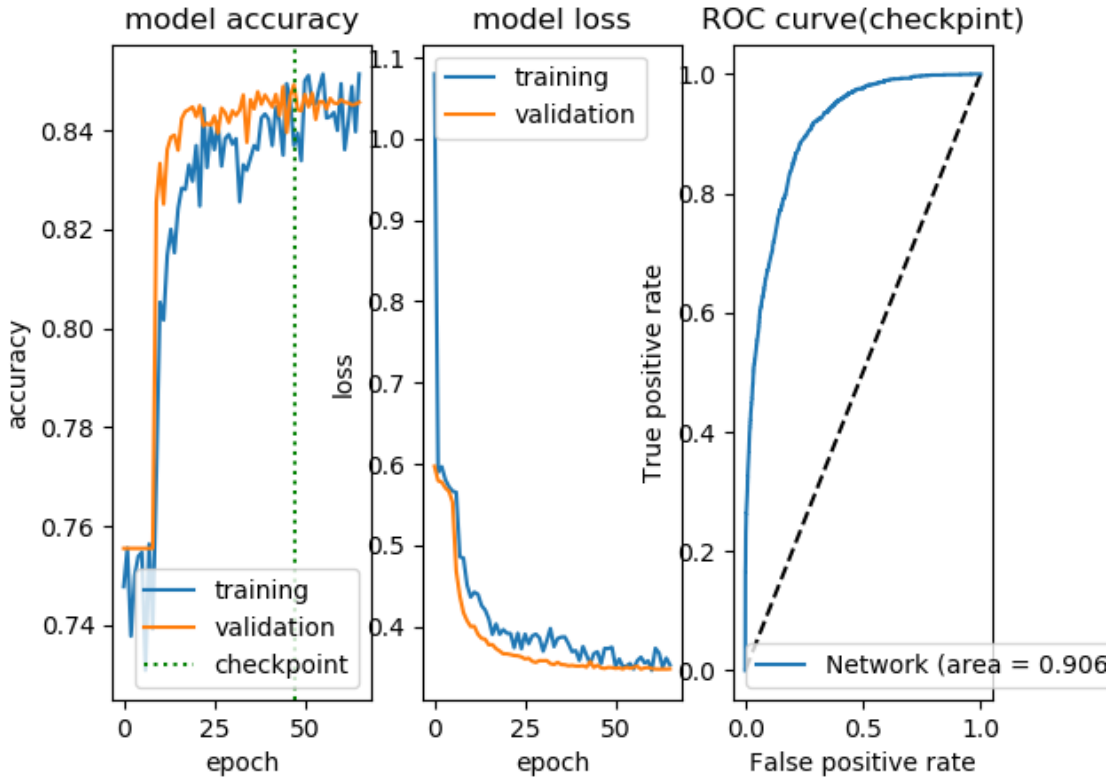


FIGURE 5.1: Overview of the training process on the network. with epochs=200, batch_size=120

5.3 Architecture of Implemented System

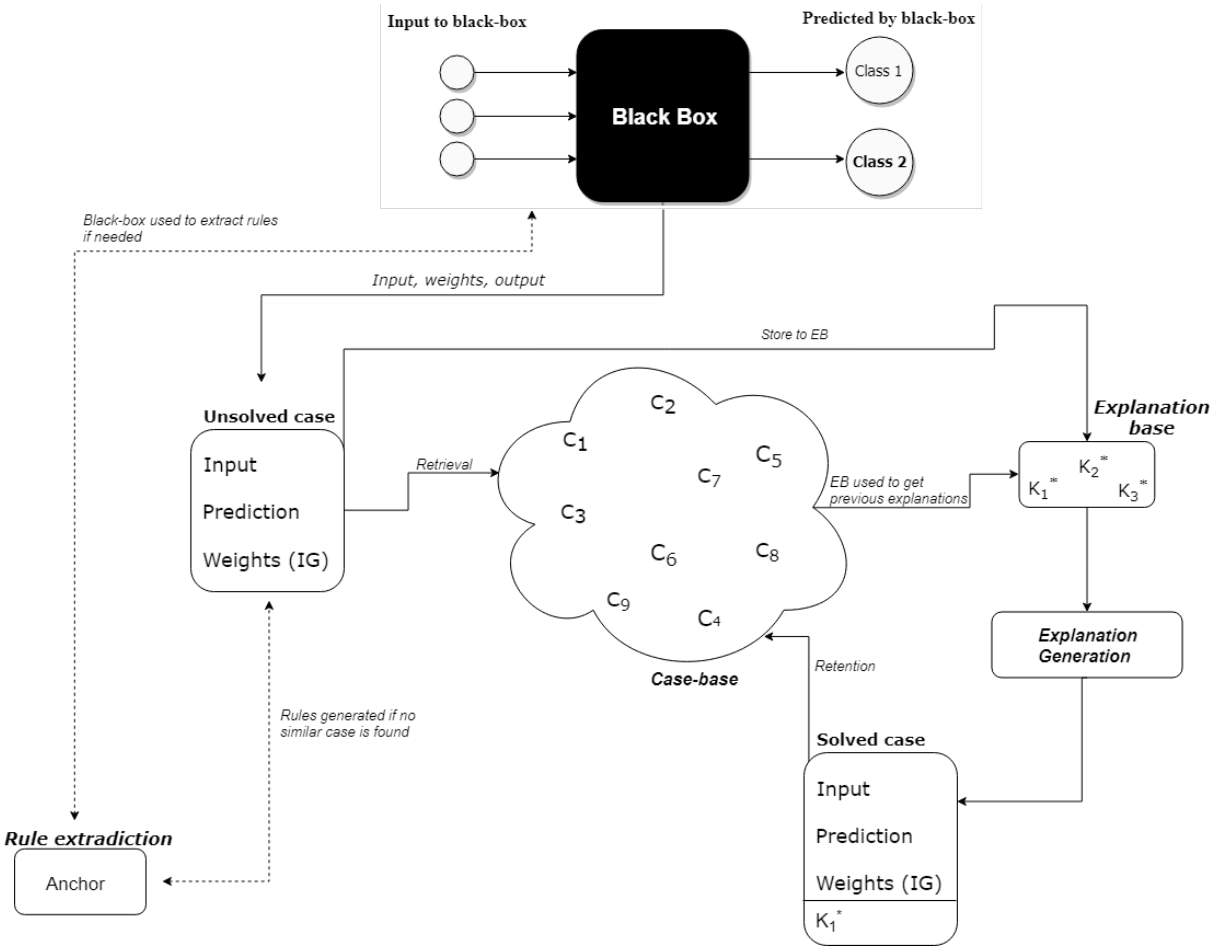


FIGURE 5.2: An overview of the implemented architecture.

Figure 5.2 depicts the architecture of our implementation. This is a specific implementation of the general architecture that is described in chapter 4. Many of the general ideas are not implemented yet, as these would be challenging to implement with the time constraint for this thesis. The purpose of this implementation is to show how this system can function at a minimum as a *proof of concept*. Figure 5.4 explains more detailed how the system works with regard to the CBR cycle. This figure gives a brief overview of how the system works. Both figures have some overlap, but this figure depicts some aspect that are not covered by figure 5.4. Most notably that the unsolved case contains no pointers to the explanation-base as it has not yet been solved.

Firstly the figure show the black-box (DNN) with its feature value inputs and prediction output. These, along with the weighting from Integrated Gradients, are included in a unsolved case. This case is queried against the case-base for similar cases. If a case is found we utilize the explanation-knowledge in the explanation-base to generate an explanation. If we don't find any

previous cases that are sufficient in solving the particular unsolved case, we need to query the rule extractions directly (Anchor in this instance), which we use along with other information to explain the problem.

The pseudo-code in algorithm 1, show in broad terms, how the implementation of the architecture works. It begins by pre-processing the dataset, as required, with already mentioned discretization techniques. In the next step, Anchor is initialized on the pre-processed dataset. Then, we find the new problem case, which consists of the attributes values X, the prediction from the black-box $y = f(X)$ and the attribution values from Integrated Gradients.

The second step is doing a retrieve from the case-base to find similar cases to be used for explaining the prediction. If we don't find any cases, we need to query the model induction method Anchor directly for explanation-knowledge.

The next step is checking if the explanation-knowledge retrieved fit our problem instance: whether the Anchor is a good fit on the problem case. If it does indeed fit (even if only partially), we can use it. Otherwise we query Anchor directly.

Once we have an anchor explanation that can be used, we go the explanation generation step. This step is simply a presentation of the anchor in readable form, along with the previous case instance used to find this explanation-knowlege in the first place.

Algorithm 1 Pseudo Code for Explaining prediction $y = f(X)$ from Black-box

Require: DNN available
Require: CBR running (myCBR REST API)
Require: Induction Methods available (Integrated Gradients and Anchor)
  Perform pre-processing steps on Dataset (discretization and cleaning)
  Initialize Anchor on pre-processed Dataset
  $X \leftarrow$ Features
  $y \leftarrow f(X)$
  $Case \leftarrow (X, y, Attribution from Integrated Gradients)$
  while *Explanation* is None **do**
    if Case-Base is empty **then**
      Get knowledge to explain from Anchor
    else
      Results $\leftarrow$ Query(Case against Case-Base)
      Check if retrieved cases explanation-knowledge from *Results* fit target case.
      if retrieved similar knowledge does not fit case **then**
        Get knowledge to explain from Anchor
      end if
    end if
    if No similar cases retrieved **then**
      Present Explanation to user as Anchor+Attributions
    else
      Present Explanation to user as previous similar knowledge + Attribution + Previous Cases
    end if
    Retain Case as new solved case in Case-Base and corresponding knowledge used in explanation-base
  end while

5.3.1 Attribution Weighting

To generate the attribution weighting, we have options such as DeepSHAP, which is a combination of SHAP and DeepLift, with DeepLift being a good approximation of integrated gradients [60]. Ultimately we decided to use Integrated Gradients [86], as it is said to work as well as other gradient methods, but being a bit more general to the type of DNN [6]. Weights generated by the induction methods can be used to attribute importance to the top k features. These weights may serve as a form of explanation which is separate from the rule-based explanation.

The weighting is in the same format as the input to the DNN. In our implementation we used a one-hot vector per feature, resulting in a vector of size 72. This needed to be reduced to the original 12 categories, as each features n-one-hot, represented the weights of the complete feature. E.g. given a weight-vector [0,0,0.24,0,0.4,0,...] with the first 5 indexes representing a one-hot vector representation of the first feature. The final weight for this feature is 0.24, and so on for each feature. The attributions are ranked with respect to the features in the following manner: [age, work class, education, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, country].

5.3.2 Model-Induction Method

After a thorough literature review, we narrowed the candidates down to LORE [40] and Anchors [75]. These give a good overview of how the model works on a local level. Anchor, however, gives a much better picture in relation to the specific area it describes as shown in figure 3.16, whereas LORE simply presents the rules without considering areas outside of its reach. The other difference between these is how the knowledge is represented. LORE uses decision trees, which gives the ability to learn from the different paths created, e.g. contrasting paths (differing prediction). Anchor on the other hand, maintain the knowledge in simple predicate logic rules. Another difference is in how the permutations are made against the black-box. LORE utilizes a genetic algorithm to optimize this permutation step, whilst Anchor uses the same permutation as LIME, a randomly distributed permutation.

With this in mind, we decided to use Anchors as the final explanation engine with the importance of keeping the explanation interpretable and concise with the rest of the feature space. As we've discussed in the general architecture chapter 4.3, combining several of the model induction methods in the future may be even better. Our implementation only utilizes Anchor.

An anchor example is shown in listing 5.3, with the various values returned. Precision is defined as: *of all instances in dataset D, where the set of predicates in A apply, how many share the same prediction.* Coverage is how "wide" the anchor can be applied on the validation dataset. *How large percentage of the validation dataset D can the anchor be applied to.* These two measures are a big part of selecting good anchors. If the precision is set too high, the coverage will fall short, resulting in anchors that only fit a very small selection of problems. During testing, it was found that a 95% precision was a good target value resulting in a coverage usually between 1% and 5% of the validation dataset. Occasionally, the Anchor generated surpassed these numbers.

This anchor in text reads: IF marital status = Never-Married AND education = High School grad THEN predict salary > 50000. Whats interesting about this particular anchor, is that with a relatively large coverage of 18%, it can predict the correct class in 97% of the problems.

```
"4": {
    "__class__": "Explanation",
    "names": [
        1,
        4
    ],
    "feature": [
        3,
        2
    ],
    "precision": [
        0.9135338345864662,
        0.9766666666666667
    ],
    "coverage": [
        0.3202,
        0.1823
    ],
    "prediction": 0,
}
```


Listing 5.3: Example of Anchor in JSON-format

Meaning on 18% of the problems, we find the prediction better than the original networks overall 85%.

5.3.3 CBR Frameworks

There are various frameworks for implementing a case-based reasoning system. The top contenders are myCBRBach et al. [9] and jCOLIBRI Recio-García et al. [74]. As there is little up to date comparisons between the features in each framework, it is hard to know which one best fits our needs. Since we know that the faculty at NTNU both uses and actively develops myCBR, we thought it's better suited for us in terms of available resources to choose myCBR. The XAI and DNN algorithms presented are however implemented in Python, and there are, as far as we know, not any suitable implementation frameworks for a CBR system available in Python. Developing one specifically for our needs is infeasible given the time- and resource-constraint on the thesis.

myCBR[2] is an open-source CBR software development kit which is developed in a joint effort by Competence Centre CBR at DFKI in Germany, School of Computing ant Technology at UWL in UK and Norwegian University of Science and Technology in Trondheim. myCBR offers the myCBR Workbench which is a powerful GUI for similarity modeling, similarity-based retrieval and more. In addition, myCBR has recently added a REST API[3] which creates an interface to

[2] http://mycbr-project.org/index.html
[3] https://github.com/ntnu-ai-lab/mycbr-rest

the code running the myCBR engine. The REST API allows for seamless integration with other applications regardless of programming language. In our case, this has allowed us to use Python throughout the system, for both DNN and CBR, as myCBR operations can simply be queried using the REST API.

The myCBR REST API is very much a work in progress and it's safe to say we have had some difficulties along the way. The documentation haven't been as detailed as we would like and we have had difficulties running some of the operations. Luckily, one of the myCBR maintainers have been very helpful and provided guidance whenever needed. In addition to the REST API, we have utilized the myCBR workbench GUI for operations which are either not yet implemented in the REST or not as easy to implement using the REST API, such as similarity modelling.

5.3.4 Explanation-Base

The explanation-base was also implemented in Python. The pointers from each case will tell which explanation was used to explain a given instance. As the dataset is encoded, each value(name) and feature is labeled according to listing numbers. Listing 5.4 shows two explanations. The first explanation, with id 0, can be read as IF feature 3 = value 4 AND feature 4 = value 6, THEN predict class 0. *Precision* and *coverage* is calculated on a subset (10.000) of instances in the validation dataset. The precision on the full anchor 0, is 0.99 % correct on this subset, where the full anchor hold, with a score of 0.8 on instances in the dataset that contains only feature 3 = value 4. The coverage score is how large percentage of the dataset that the anchor can fit.

The storage of the explanation-base is a simple JSON file where each explanation is an explanation object. With this implementation, the storage is very flexible with respect to allowing different types of explanations in the future, other than only Anchors.

```
{
    "0": {
        "__class__": "Explanation",
        "names":    [4, 6],
        "feature": [3, 4],
        "precision": [0.8, 0.99],
        "coverage":  [0.4, 0.1],
        "prediction": 0
    },
    "1": {
        "__class__": "Explanation",
        "names":    [5, 7, 1],
        "feature": [7, 3, 4],
```

```
        "precision": [0.8, 0.92, 0.99],
        "coverage":  [0.4, 0.23, 0.1],
        "prediction": 1
    },
    ...
}
```

Listing 5.4: Explanation in JSON format

5.3.5 Case-Base

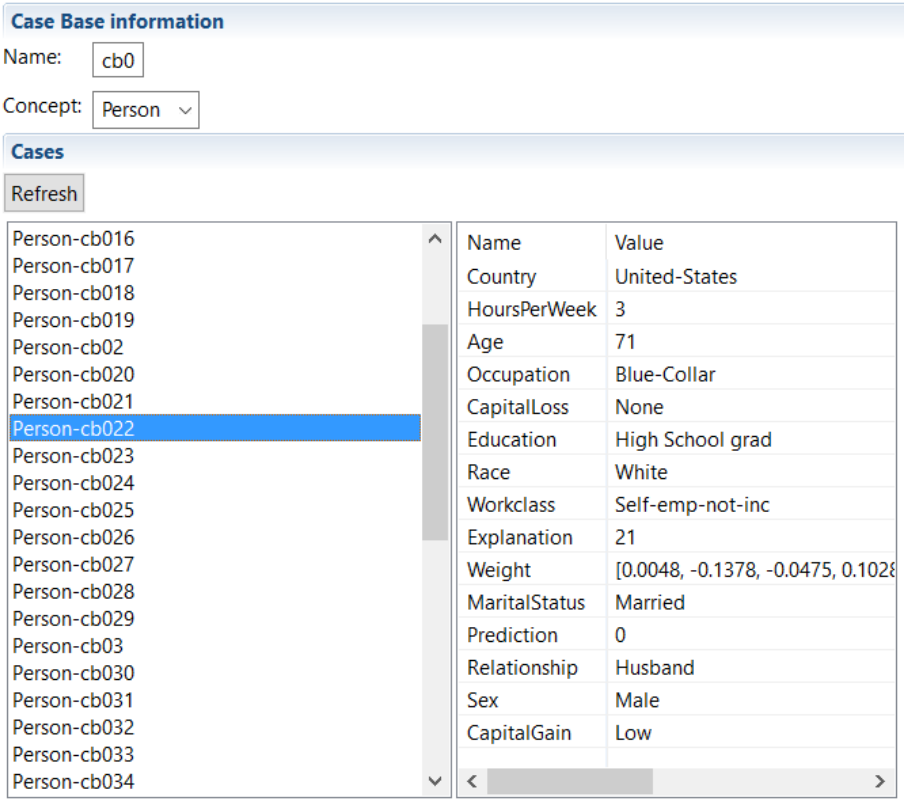


Figure 5.3: How a solved case looks inside the case-base in myCBR.

Figure 5.3 shows how a solved case looks inside myCBR. Apart from *Explanation*, *Weight*, which is weights from integrated gradients, and *Prediction*, which is the prediction from the black-box, the rest are input features. Considering the explanation for each case can become quite long if they were to be stored as strings, only a pointer to the explanation-base is stored in the solved case itself.

5.3.6 Modified CBR cycle

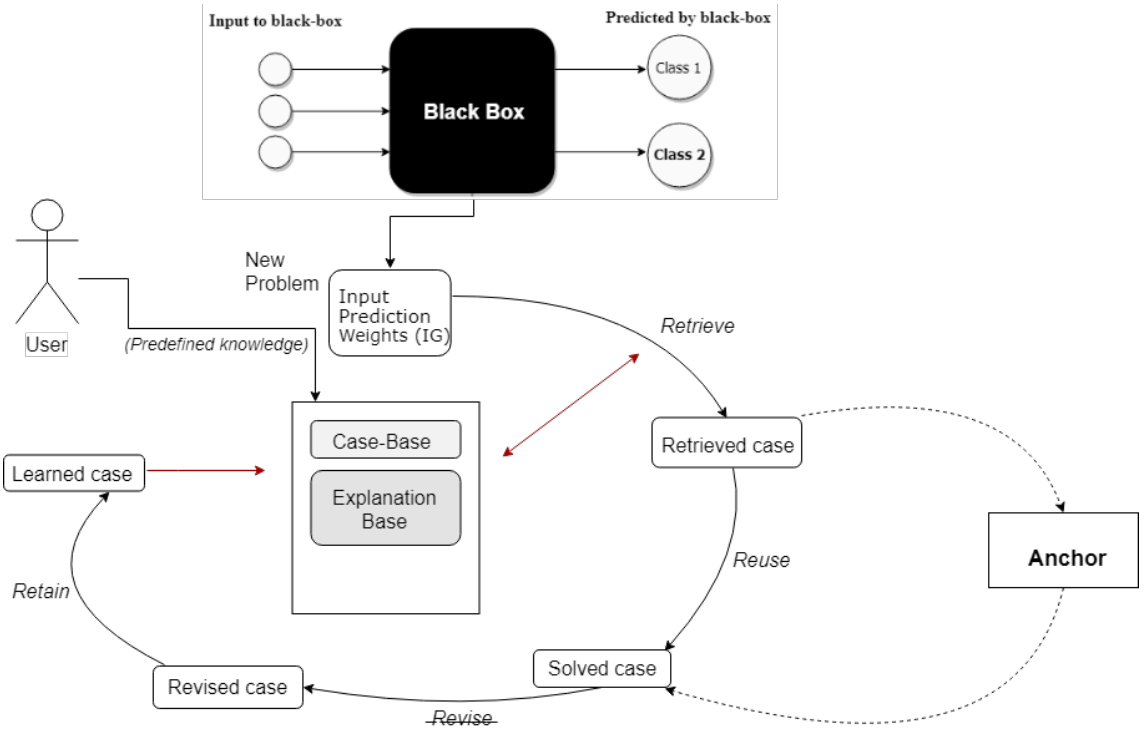


FIGURE 5.4: The lifecycle of each case which enters the modified CBR cycle.

Figure 5.4 shows specifically what happens with regard to the retrieve, reuse, revise and retain-step in the CBR cycle. How each step is related and how it relates to our architecture will become clear with a specific instance from the dataset. The given instance will enter the explainer part of the system as a query case with input (features from data point), prediction given by black-box and attribution given by integrated gradients. In the first step, the retrieve step, the query case is compared to cases in the case-base to find similar cases. Similar cases already retained in the system are useful as explanations have already been generated (from anchor) and can, therefore, be reused to explain new cases. This is where the various similarity measures which were predefined are used to find the most similar or the k most similar cases. Given that the system can't find any similar cases, Anchor is used to generate new explanations. The anchor step is skipped otherwise. As implementation of the revise step is left for future work, this step is skipped. The final explanation given by the system consist of presenting the previous case with the matching explanation (given that such a case was found), attribution weights showing the most important features and explanation generated from anchors for that specific case (in case a similar case wasn't found in the retrieve step). The learned case is then retained in the case-base for future instances. The explanation is stored in the explanation-base with a pointer to the explanation in the case itself.

Retrieval

To implement the retrieval, we utilised an amalgamation function within myCBR. An amalgamation function is a combination of functions of local similarity between the different attributes in each case. In the equation 5.1, we get the similarity between case **A** and **B**. w_i is the weighting between each attribute, in our system this was set to 1, as we didn't know the relative importance of each attribute related to the classification problem.

$$sim_g(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{n} w_i \cdot \text{sim}_{fn}(a_i, b_i) \tag{5.1}$$

We performed multiple tests to validate our query results against the explanations from Anchor. For features which were of type *integer*, we utilised the *openml* page on the adult dataset[4] to extract information on min/max/average (which is shown in figure 5.7 for the age-attribute). Features with the symbol type are considered individually. Some of the features are more straight forward such as *sex* and *prediction*. Others, such as *education* and *occupation*, need to be considered individually. Features with type string are left out when doing retrieval.

Retrieval with weights

In order to figure out if the weights hold any value in the retrieval step, we setup a simple test, With the use of the cosine similarity in equation 5.2, Euclidean distance in equation 5.3 and cosine with prediction used. This is to do retrieval between weight vector **A** and **B** corresponding to attribution weights from Integrated Gradients on each case. If the value is close to 0, then the weights are very similar. To validate the results, the baseline is simply the index of each case, instead of the cosine similarity query results.

$$\text{sim}_w = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.2}$$

$$\text{sim}_w = \text{euclid}(\theta) = d(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{5.3}$$

To improve upon the retrieval step, we utilized the idea from Caruana et al. [15], but instead of utilizing internal weights (activation patterns) for similarity, we simply connected the attribution weights to the retrieval. It was a hypothesis we had that this would help us in finding sufficiently

[4] `https://www.openml.org/d/1590`

similar cases in the case-base. To test this hypothesis, we derived a few tests on retrieval. Each case in the case-base had an explanation from Anchors, along with the test cases. The retrieval on a test case against the case-base was only performed with their weights from Integrated Gradients. Various similarity measures such as Euclidean distance, cosine distance, cosine distance with prediction similarity (0 if equal, 1 if not) and None (no ranking, only random listing), are used to test the case-base. We were interested in finding out whether or not the anchor explanations are equal. If the Anchors are identical, then we know that the similarity score should be very high. If the retrieved case was similar, the anchor explanation should be similar as well, at least partially.

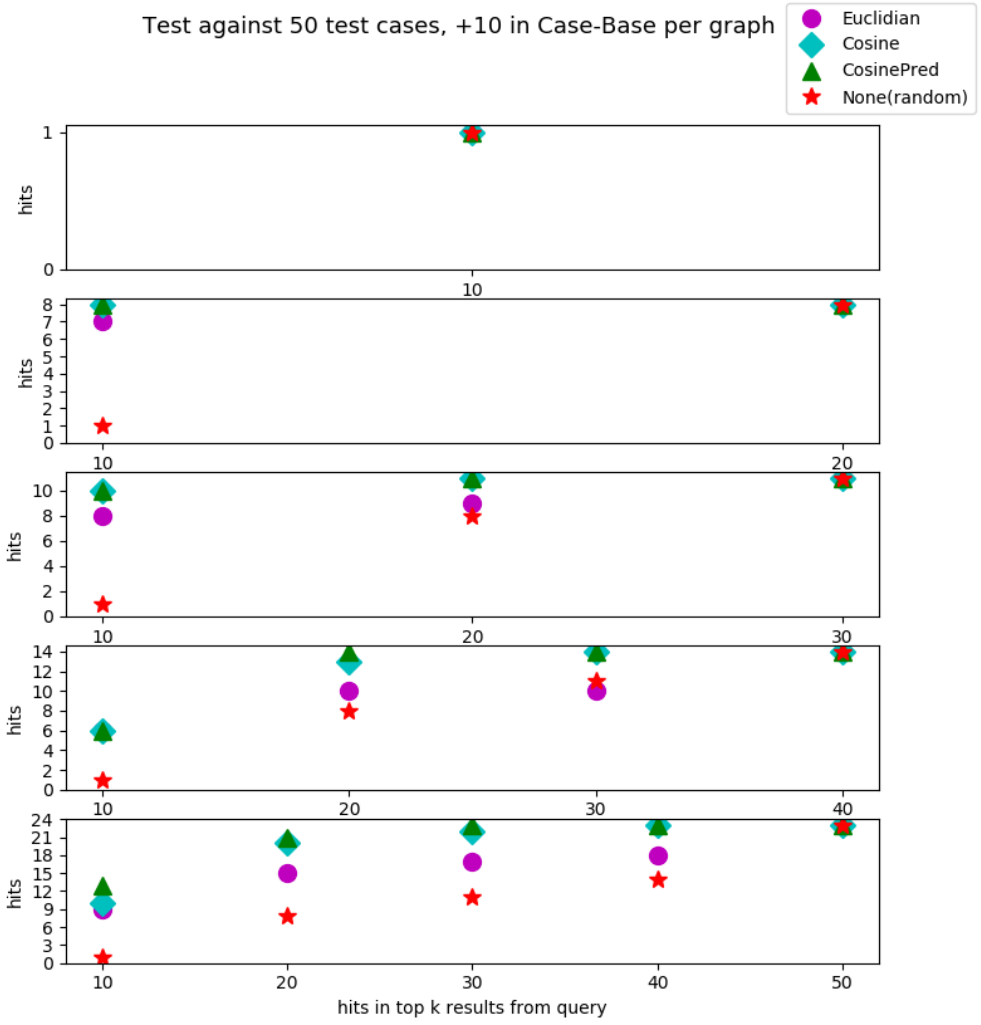


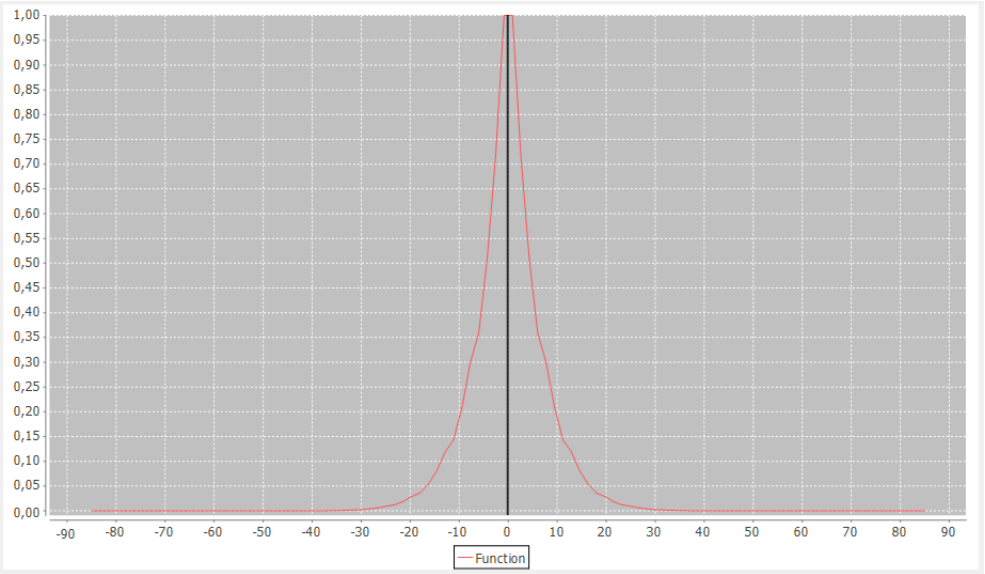
FIGURE 5.5: Results of query against case-base using only similarity between weights to test anchor similarity. Hit score is perfect anchor explanation matches.

The graphs in figure 5.5 show results from different similarity measure queries done against case-bases of different sizes (from 10 at top to 50 at bottom graph) to compare weights from integrated gradients in the cases. X-axis shows how many exactly similar anchors there are between the case-base and test-case from the top k cases in the query, while the Y-axis shows

the number of times the anchor on a test case gives an exact match to *one* cases anchor from the case-base, from top k in the query ranking. Each case-base is tested against a pool of 50 randomly picked cases from the test-set. The number of cases in the case-base is increased by 10 for each run to test various case-bases sizes. The cases used in each case-base are taken from the validation-set, and all of these cases have explanations that have been generated using Anchor. E.g. In a case-base with 5 cases (c_1 to c_5), and a query with 2 test-cases (tc_1 and tc_2). We get two query results per metric. For euclidean distance only we could get $[c_2, c_3, c_1, c_5, c_4]$ for tc_1 and $[c_4, c_2, c_1, c_3, c_5]$ for tc_2 respectively, the hit rate between the anchor in case tc and case c anchor could be [1,0,1,0,0] and [0,0,1,1,1]. The top-k is calculated as number of hits (successes) in from the first query result down to index k. The sum of these would be [1,0,2,1,1], and top-k with k incrementing by 2 (2,4,6) would be [1,4,5]. This is what is plotted for each metrics results queries, with k incrementing by 10 in figure 5.5.

The results show that using weights in the retrieval to check for the similarity between a test-case and a case in the case-base improve the results. We want the cases that give a correct solution to be as close to the top of the query as possible. In the last graph with 50 cases, about 12 of the 24 cases were found in the top 10 spots. This is by no means a perfect retrieval score with regard to the relevant cases in the top 10 query results, but it does show an improvement compared to the random similarity measure. The results were consistent with a random distribution of the cases.

Similarity Measures



(A) Similarity function for age

	Associates	Bachelors	Doctorate	Dropout	igh School gra	Masters	Prof-School
Associates	1.0	0.8	0.2	0.8	0.6	0.6	0.4
Bachelors	0.8	1.0	0.4	0.6	0.4	0.8	0.6
Doctorate	0.2	0.4	1.0	0.1	0.2	0.6	0.8
Dropout	0.8	0.6	0.1	1.0	0.8	0.4	0.2
igh School gra	0.6	0.4	0.2	0.8	1.0	0.3	0.2
Masters	0.6	0.8	0.6	0.4	0.3	1.0	0.8
Prof-School	0.4	0.6	0.8	0.2	0.2	0.8	1.0

(B) Similarity function for education

	ıer-Indian-Eski	ian-Pac-Island	Black	Other	White
er-Indian-Eski	1.0	0.0	0.0	0.0	0.0
ian-Pac-Islanc	0.0	1.0	0.0	0.0	0.0
Black	0.0	0.0	1.0	0.0	0.0
Other	0.0	0.0	0.0	1.0	0.0
White	0.0	0.0	0.0	0.0	1.0

(C) Similarity function for race

FIGURE 5.6: Similarity functions for three of the attributes in a case.

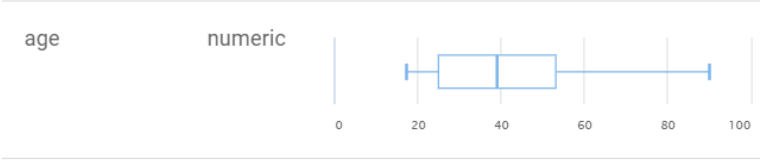


FIGURE 5.7: How the instances are distributed for age-attribute.

From figure 5.7, we can see various information on the age-attribute. It shows min at 18, max at 90 and average at 38.

Figure a) in fig 5.6 shows the similarity function for the age-attribute. It converges to 0 after reaching the deviation value equal to the average. After running some test, we've found that this is the function that best punishes any values that are overly deviant. A similar function is modelled for the other integer attribute, namely *hours_per_week*.

Figure b) shows the similarity function which has been modelled after which education degree has the most income. The scores are the result of running various tests and tweaking to find what works best for this particular attribute. Upon modelling the function for this particular attribute, we found that it required a lot of time to find the optimal similarity function for each of the 12 attributes. Therefore, the rest of the attributes, which are of the "symbol" type (not including age and hours_per_week which are integers and explanation and prediction which are strings) in myCBR are modelled in the simplest form, just as in figure c), where they're only considered similar when there's an exact match. The *weight* and *explanation* attribute are deactivated in the retrieval stage as they are not supported or relevant in the myCBR framework for the retrieval-stage.

5.4 Running Example

To illustrate how the system works, we have created a project (adult_final.prj in the projects folder) which has been populated with five random cases from the validation set. It's depicted in figure 5.8, from a) to e). Then we have one query case, f), which is the target case we want to explain.

Name	Value
CapitalGain	None
CapitalLoss	None
Sex	Male
HoursPerWeek	40
Race	White
Age	35
Education	Associates
Explanation	0
MaritalStatus	Never-Married
Occupation	Blue-Collar
Country	United-States
Workclass	State-gov
Relationship	Not-in-family
Prediction	0
Weight	[0.0177, -0.0683, 0.0094, -0.0982

(A) Case 1

Name	Value
CapitalGain	None
CapitalLoss	None
Sex	Male
HoursPerWeek	50
Race	White
Age	38
Education	High School grad
Explanation	1
MaritalStatus	Married
Occupation	White-Collar
Country	United-States
Workclass	Self-emp-inc
Relationship	Husband
Prediction	1
Weight	[0.0933, 0.1454, -0.2919, 0.0776,

(B) Case 2

Name	Value
CapitalGain	None
CapitalLoss	None
Sex	Male
HoursPerWeek	43
Race	White
Age	45
Education	Masters
Explanation	2
MaritalStatus	Married
Occupation	Professional
Country	United-States
Workclass	Private
Relationship	Husband
Prediction	1
Weight	[0.2246, 0.0232, 0.1101, 0.1672,

(C) Case 3

Name	Value
CapitalGain	None
CapitalLoss	None
Sex	Female
HoursPerWeek	40
Race	White
Age	19
Education	High School grad
Explanation	3
MaritalStatus	Never-Married
Occupation	Sales
Country	United-States
Workclass	Private
Relationship	Not-in-family
Prediction	0
Weight	[-0.0552, -0.0121, -0.0508, -0.03

(D) Case 4

Name	Value
CapitalGain	None
CapitalLoss	None
Sex	Male
HoursPerWeek	40
Race	White
Age	35
Education	High School grad
Explanation	4
MaritalStatus	Married
Occupation	Blue-Collar
Country	United-States
Workclass	Local-gov
Relationship	Husband
Prediction	0
Weight	[0.1143, -0.0641, 0.0509, 0.1357,

(E) Case 5


```
{
    "Age":43,
    "CapitalGain":"None",
    "CapitalLoss":"None",
    "Country":"United-States",
    "Education":"Bachelors",
    "Explanation":"0",
    "HoursPerWeek":40,
    "MaritalStatus":"Never-Married",
    "Occupation":"Service",
    "Prediction":"0",
    "Race":"White",
    "Relationship":"Own-child",
    "Sex":"Male",
    "Weight":"[0.1293, 0.0244, -0.0168, -0.1195,
              -0.1036, -0.1204, -0.0147, 0.1077,
              -0.1014, 0.0503, -0.0317, 0.0361]",
    "Workclass":"Local-gov"
}
```


(F) Query case

FIGURE 5.8: Overview of the five cases in the case-base and the query case we want an explanation for.

For this query case, Anchor is automatically generated to show how the Anchor compares to the Anchor generated for the most retrieved case. In practice, the Anchor for a new case could be set to only be generated when it's under a certain threshold for a similarity score. For this case, the similarity is 76.17 %.

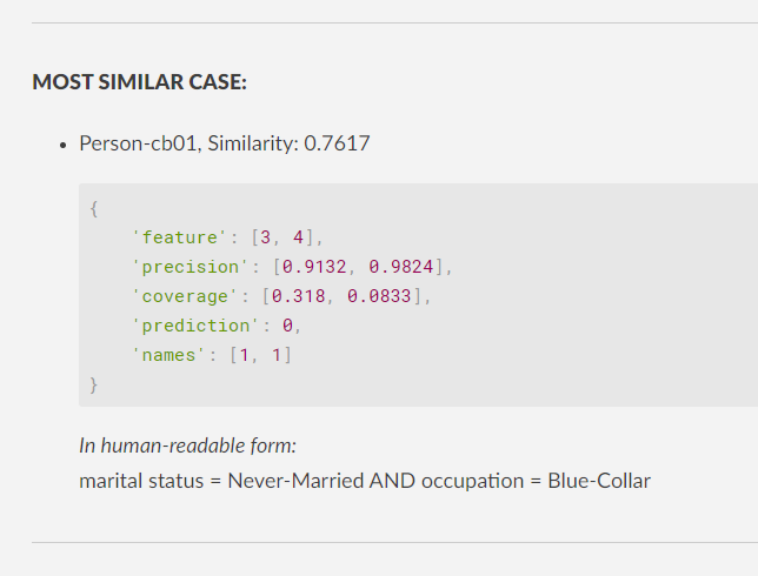


FIGURE 5.9: Most similar explanation

This is the explanation outputted from the system from the case it deemed to be the most similar. The explanation from the retrieved case is not an exact match, but the partial anchor of only *marital status = Never Married* grad holds as an easily interpretable explanation to why the prediction was set to class 0. The coverage and precision show, that with this partial anchor, the precision is 91 % correct, and is able to hit 31% of the validation dataset.

Name	Value
CapitalGain	None
CapitalLoss	None
Sex	Male
HoursPerWeek	40
Race	White
Age	35
Education	Associates
Explanation	0
MaritalStatus	Never-Married
Occupation	Blue-Collar
Country	United-States
Workclass	State-gov
Relationship	Not-in-family
Prediction	0
Weight	[0.0177, -0.0683, 0.0094, -0.098

FIGURE 5.10: Most similar case is case 1.

Additionally, the user is also presented the case that was used to explain.

Chapter 6

Discussion

In this chapter we have a general discussions on topics that hasn't been covered in earlier discussions. In the next secion, we present the issue of reproducibility and our approach to achieve this with the thesis.

6.1 General Discussion

One question that comes up regarding our system is why use the CBR system at all? Why not use the model induction methods by themselves? One answer to this is that we can't know whether or not if it is time-reasonable to predict the behavior of the model from every instance that is to be explained. We can assume that the time needed might be too long, and above what can be considered within a reasonable time frame. In our implementation it takes about one second to generate the Anchor for one given instance, we know that this number will likely increase as we increase the number of features. If we had even more complex methods as well, this could be an even more drastic change. Retrieval against a case-base would in this case be much faster. One possibility is that the time needed to calculate *some* of the model induction methods doesn't grow outside of its usefulness. In this case, we could get *some* explanation-knowledge to use alongside the context and the case-base system. This would allow for more precise explanations toward a particular instance, as we have done with using the attribution weighting from Integrated Gradients.

By using previous explanations in the explanation generation phase along with the contextual information given by the user, we should be able to give better explanations that fit the user needs. These explanations would also be more interpretable with the inclusion of feedback, to guide the explanation towards the goal of the user.

General Architecture

There is a problem related to the type of explanation that is given in a system like this. Looking at figure 4.2 with the explanation from the black-box to factory prediction: the explainer is isolated from the predictor. This results in explanations that seem to fit the prediction, but confirming whether or not explanation has a high degree of fidelity related to the black-box' own prediction, is very difficult. An isolated system would in some instances fail to produce an explanation that really captures the inner workings of the model. If it could, it should be able to give the prediction itself as stated by Rudin [79]. No model is always correct, and as such should not always be trusted in every instance. In this instance, the explanation could give a false sense of trust, where the resulting prediction could be very damaging.

Verifying the different model-induction techniques are also a problem, if the methods themselves show conflicting explanations between one another, the model-induction technique might be weak.

How many cases need to be stored in the case-base and how to deal with conflicting anchors is also a matter of interest. In terms of multiple potential anchors to explain a given instance, it would probably be a good idea to present the user with the most applicable cases and present multiple explanations if needed.

6.2 Reproducibility

Reproducibility is important for the trustworthiness of a paper. Following the increased research in AI, we have seen an increasing number of researchers addressing the issue of *reproducibility*. Unfortunately, just as "explainability" is hard to define, there is no one agreed upon definition of reproducibility. Nevertheless, Gundersen et al. [41], have a definition which we agree with: *"Reproducibility in empirical AI research is the ability of an independent research team to produce the same results using the same AI method based on the documentation made by the original research team."*

According to Gundersen et al., research points to open-data and open-code is an important step towards reproducibility. In our thesis, we have taken steps to preserve the ability to reproduce our results. Firstly, the UCI adult dataset we're using is open and can be looked up by anyone on their website [1]. Secondly, our code is open source and can be found in a public repository on GitHub [28]. Thirdly, our code is well commented in addition to this thesis which functions as added documentation to the architecture a whole. Lastly, all our code is written such that the same seed is sent anytime there is any randomization involved to be reproducible at a later stage.

[1] `https://archive.ics.uci.edu/ml/datasets/Adult`

Chapter 7

Conclusion and Future Work

In this chapter, we will begin by summing up what has been done in 7.1, listing our contributions in 7.2, presenting future work in 7.3 and some final words about the thesis in 7.4.

7.1 Conclusion

In this thesis, we have presented why XAI has become a hot topic in the industry, among ordinary citizens and regulators on issues related to trust and privacy. We've performed a literature review on state of the art in XAI, reviewing, the latest conferences in the field such as IJCAI 2017, IJCAI 2018 and ICCBR 2018. We have performed a similar literature review for case-based reasoning and discussed how it has been used to generate explanations. We have presented a general architecture, implemented parts of it and presented a specific running example. Finally, we will present the contribution of this thesis and what we think can be done in the future to improve the system.

In the introduction, we made the reader aware that the implemented system is a proof of concept. Working on this thesis we have learned that this thesis has been greater in scope than what we first anticipated. Our solution is by no means a finished product and there are many things which can be improved upon. We have mentioned in the general architecture and in the future workings, what can be tweaked to create a better explainer. When we started this thesis, we set out to answer three research questions. In the remainder of this section, we will summarize how we have tackled these questions in this thesis:

Research Question 1: *What is the current status of eXplainable Artificial Intelligence (XAI) and eXplainable Artificial Intelligence (XAI) related to Case-Based Reasoning (CBR)?*

In the first research question, we were interested in finding state of the art in XAI. XAI has seen an increased interest in research with conferences, such as IJCAI, solely dedicated to the topic

of XAI. Chapter 3 is a direct attempt at tackling the first research question. In short, we have found that most research papers can be divided into one of three categories: model induction, interpretable models or deep explanation. Although our focus has been on model induction, as this has been most relevant for our thesis, we have also described state of the art in the other categories.

Research Question 2: *How can model-agnostic methods be combined with Case-Based Reasoning (CBR) to gain knowledge of an underlying Deep Neural Network (DNN)?*

Chapter 4, and more specifically chapter 4.3, seeks to answer this. Here we presented a general architecture which could combine various model-induction methods with CBR to explain the DNN. We believe that by combining model-induction methods directly on a DNN to find patterns, along with the experience natured CBR system, we can gain direct knowledge about the DNN itself to fully interpret and justify the predictions.

Research Question 3: *Can our proposed system provide a step towards achieving user-understandable explanations?*

Through our work with both the specialization project and the master's thesis it has become clear that it is hard to define what constitutes "a good explanation" with respect to the user. Also, it's challenging to increase the interpretability as this is highly subjective. Among others factors, user types, data complexity, and more, all come into play when defining this. We believe that with the inclusion of user-context and user-feedback (as described in chapter 4.3) from an explanation, the system could be able to adjust the explanations to achieve the different goals as defined by the user itself, and thus give more understandable explanations.

7.2 Contribution

The main contribution of this thesis is:

1. A summary of our review on state of the art from the specialization project. Additionally, an extension of the research with a focus on model induction techniques in combination with case-based methods.
2. A presentation of a general architecture for achieving model-agnostic explainable AI on a black-box algorithm (DNN) utilizing the CBR methodology.
3. Commented implementation code, a trained DNN which has been open-sourced, various myCBR projects and results from the open dataset used.

7.3 Future work

There are still questions that need to be answered related to improvements and a full implementation of the general architecture proposed.

Evaluation of selected model induction methods
The model-induction techniques used may behave poorly under certain circumstances. A broader evaluation of our selective methods should be considered for further analysis. It would be valuable to figure out the limitation of these methods, and how "far" they apply. How to evaluate these methods are in and of itself a great challenge, as there isn't one defined metric that can be used. With the current pace of research in XAI, we think that better alternative methods will appear. Any method that can give more insight than the previous is useful and a step in the right direction.

This question also applies to how well the model performed, as the method used could be very reliant on whether the model performed well enough to be used in the first place. E.g. when testing the Anchor implementation, we noticed that if the model was not trained (only random initiation of weights) before using Anchor, we couldn't get any results from Anchor. Figuring out how the method relate to the performance of the model itself is very important. As many of the induction techniques' usefulness may be directly connected to the performance of the DNN, we might get completely different explanations from say, a DNN with 60% accuracy, versus a DNN with 90%.

Combining Induction Techniques
We think that by combining multiple methods into one, we should be able to gain insight into a larger part of the underlying knowledge hidden inside the black-box. Which induction techniques to combine and how these can be used in tandem is a research point which needs to be investigated further. In our example, in listing 4.1, we have shown how the explanation-base may look with Anchor, LIME, LORE, etc. However, the details on how these methods would work together is something that needs to be taken a closer look at. We want to amplify the explanatory coverage of each individual method, by utilizing the individual strengths of each method to create more useful explanations.

Feedback
Feedback from the user is a key property of an explainable system, regardless of whether or not the explanation was sufficient or even correct. With feedback from the user, the explanatory reasoner in the CBR system, can be altered to capture patterns that were not only wrong from the explanation generation itself, but also the DNN models prediction. How to include this type of feedback, and how to use it is a problem not fully considered in this thesis. As feedback could have many goals and variations. Handling them all is not an easy task, and is something that needs to be looked on.

Knowledge representation

Knowledge representation is an essential part of any knowledge/reasoner system, and as such, a broader knowledge representation should be utilized. Simple rules as sets of predicates cannot capture all relevant rules for our tabular problem case, or any other black-box system for that matter. Simple AND/OR rules are not sufficient for a lot of representations, especially relational information or information related to a group. The simpler the rule, the easier it is to work with and present to the user, but the less coverage of the true complexity on the domain is it able to capture. Finding a middle ground, or combining different representations to work from should be figured out.

Evaluation on users

Any continuation of this project would be served with conducting user tests throughout the development process. As it is hard to define what can be said to be a good explanation, continuous users tests with users who are actually going to utilize the system is highly recommended. Conducting a user test should be considered a priority to develop this system further. This will also show if the system actually works in a real-world setting.

Use attribution to find similar cases

Currently, the weights are not used in the final implemented retrieval stage when looking for similar cases. It would be very valuable to figure out whether the attribution weights do align with the similarity of the amalgamation function itself. Another test is whether or not the feature importance weight, which is commonly found in the amalgamation function, can be substituted with these attribution weights, as this expert knowledge can be hard to obtain in some domains. In a car domain, the feature related to the paint of the car is probably not as important as the engine used.

7.4 Final Thoughts

We have found this project to be interesting in numerous ways, but especially from a research perspective. Although much work still needs to be done before explainable AI reaches human level capability, it's a relatively new field which is becoming more important. We can see that although the concept of explainability has been discussed ever since the dawn of AI, it's only in recent years it has gained any traction with full conferences dedicated to the topic. Furthermore, the conferences and results in the paper's keep improving in certain areas. Hopefully, the community sees the advantages that the CBR methodology can bring to this table.

Bibliography

[1] Agnar Aamodt. "Knowledge-intensive case-based reasoning in creek". In: *European Conference on Case-Based Reasoning*. Springer. 2004, pp. 1–15.

[2] Agnar Aamodt and Enric Plaza. "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In: *AI communications* 7.1 (1994), pp. 39–59.

[3] David Alvarez Melis and Tommi Jaakkola. "Towards Robust Interpretability with Self-Explaining Neural Networks". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 7775–7784. URL: http://papers.nips.cc/paper/8003-towards-robust-interpretability-with-self-explaining-neural-networks.pdf.

[4] David Alvarez-Melis and Tommi S. Jaakkola. "A causal framework for explaining the predictions of black-box sequence-to-sequence models". In: *CoRR* abs/1707.01943 (2017). arXiv: 1707.01943. URL: http://arxiv.org/abs/1707.01943.

[5] David Alvarez-Melis and Tommi S. Jaakkola. "On the Robustness of Interpretability Methods". In: *CoRR* abs/1806.08049 (2018). arXiv: 1806.08049. URL: http://arxiv.org/abs/1806.08049.

[6] Marco Ancona et al. "A unified view of gradient-based attribution methods for Deep Neural Networks". In: *CoRR* abs/1711.06104 (2017). arXiv: 1711.06104. URL: http://arxiv.org/abs/1711.06104.

[7] Dana Angluin. "Queries and Concept Learning". In: *Machine Learning* 2.4 (Apr. 1988), pp. 319–342. ISSN: 1573-0565. DOI: 10.1023/A:1022821128753. URL: https://doi.org/10.1023/A:1022821128753.

[8] M. Gethsiyal Augasta and T. Kathirvalavakumar. "Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems". In: *Neural Processing Letters* 35.2 (Apr. 2012), pp. 131–150. ISSN: 1573-773X. DOI: 10.1007/s11063-011-9207-8. URL: https://doi.org/10.1007/s11063-011-9207-8.

[9] Kerstin Bach and Klaus-Dieter Althoff. "Developing Case-Based Reasoning Applications Using myCBR 3". In: *Case-Based Reasoning Research and Development*. Ed. by Belén Díaz Agudo and Ian Watson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 17–31. ISBN: 978-3-642-32986-9.

[10] Amir Beck and Marc Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM J. Img. Sci.* 2.1 (Mar. 2009), pp. 183–202. ISSN: 1936-4954. DOI: 10.1137/080716542. URL: http://dx.doi.org/10.1137/080716542.

[11] Mustafa Bilgic and Raymond Mooney. "Explaining Recommendations: Satisfaction vs. Promotion". In: (Jan. 2005).

[12] Alexander Binder et al. "Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers". In: *CoRR* abs/1604.00825 (2016). arXiv: 1604.00825. URL: http://arxiv.org/abs/1604.00825.

[13] Alexander Binder et al. "Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers". In: *CoRR* abs/1604.00825 (2016). arXiv: 1604.00825. URL: http://arxiv.org/abs/1604.00825.

[14] Adam Bloniarz et al. "Supervised Neighborhoods for Distributed Nonparametric Regression". In: *AISTATS*. 2016.

[15] Rich Caruana et al. "Case-based explanation of non-case-based learning methods." In: *Proceedings of the AMIA Symposium*. American Medical Informatics Association. 1999, p. 212.

[16] Chaofan Chen et al. "This looks like that: deep learning for interpretable image recognition". In: *CoRR* abs/1806.10574 (2018). arXiv: 1806.10574. URL: http://arxiv.org/abs/1806.10574.

[17] François Chollet et al. *Keras*. https://keras.io. 2015.

[18] Alan Cooper. "The Inmates are Running the Asylum". In: *Software-Ergonomie '99: Design von Informationswelten*. Ed. by Udo Arend, Edmund Eberleh, and Knut Pitschke. Wiesbaden: Vieweg+Teubner Verlag, 1999, pp. 17–17. ISBN: 978-3-322-99786-9. DOI: 10.1007/978-3-322-99786-9_1. URL: https://doi.org/10.1007/978-3-322-99786-9_1.

[19] Susan Craw and Agnar Aamodt. "Case Based Reasoning as a Model for Cognitive Artificial Intelligence". In: *Case-Based Reasoning Research and Development*. Ed. by Michael T. Cox, Peter Funk, and Shahina Begum. Cham: Springer International Publishing, 2018, pp. 62–77. ISBN: 978-3-030-01081-2.

[20] Pádraig Cunningham, Dónal Doyle, and John Loughrey. "An evaluation of the usefulness of case-based explanation". In: *International Conference on Case-Based Reasoning*. Springer. 2003, pp. 122–130.

[21] DARPA. "Explainable Artificial Intelligence (XAI)". In: (2016). URL: https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf.

[22] Patrick Doherty David W. Aha Trevor Darrell and Daniele Magazzeni, eds. *IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI-18)*. Vol. 27/23. Stockholm, Sweden, July 2018.

[23] Amit Dhurandhar et al. "Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 592–603. URL: http://papers.nips.cc/paper/7340-explanations-based-on-the-missing-towards-contrastive-explanations-with-pertinent-negatives.pdf.

[24] Derek Doran, Sarah Schulz, and Tarek R. Besold. "What Does Explainable AI Really Mean? A New Conceptualization of Perspectives". In: *CoRR* abs/1710.00794 (2017). arXiv: 1710.00794. URL: http://arxiv.org/abs/1710.00794.

[25] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

[26] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[27] Sondre Engen and Piraveen Perinparajan. "Explainable Artificial Intelligence: Using Case-Based Reasoning to Explain Deep Neural Networks". Computer Science, Specialization Project. Dec. 2018.

[28] Sondre Engen and Piraveen Perinparajan. *GitHub Project Code*. https://github.com/piravp/XAI-CBR. June 2017. DOI: 10.5281/zenodo.3247158.

[29] Neng Fan, Qipeng P. Zheng, and Panos M. Pardalos. "Robust optimization of graph partitioning involving interval uncertainty". In: *Theoretical Computer Science* 447 (2012). Combinational Algorithms and Applications (COCOA 2010), pp. 53 –61. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2011.10.015. URL: http://www.sciencedirect.com/science/article/pii/S0304397511008851.

[30] World Economic Forum. *The Global Risks Report 2017*. http://www3.weforum.org/docs/GRR17_Report_web.pdf. (Accessed on 12/10/2018).

[31] Manoel Vitor Macedo França, Artur S. D'Avila Garcez, and Gerson Zaverucha. "Relational Knowledge Extraction from Neural Networks". In: *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches - Volume 1583*. COCO'15. Montreal, Canada: CEUR-WS.org, 2015, pp. 146–154. URL: http://dl.acm.org/citation.cfm?id=2996831.2996849.

[32] Alex A. Freitas. "Comprehensible Classification Models: A Position Paper". In: *SIGKDD Explor. Newsl.* 15.1 (Mar. 2014), pp. 1–10. ISSN: 1931-0145. DOI: 10.1145/2594473.2594475. URL: http://doi.acm.org/10.1145/2594473.2594475.

[33] Nicholas Frosst and Geoffrey E. Hinton. "Distilling a Neural Network Into a Soft Decision Tree". In: *CoRR* abs/1711.09784 (2017). arXiv: 1711.09784. URL: http://arxiv.org/abs/1711.09784.

[34] Andrew Gelman, Xiao-Li Meng, and Hal Stern. "POSTERIOR PREDICTIVE ASSESSMENT OF MODEL FITNESS VIA REALIZED DISCREPANCIES". In: *Statistica Sinica* 6.4 (1996), pp. 733–760. ISSN: 10170405, 19968507. URL: http://www.jstor.org/stable/24306036.

[35] Amirata Ghorbani, Abubakar Abid, and James Zou. "Interpretation of neural networks is fragile". In: *arXiv preprint arXiv:1710.10547* (2017).

[36] Hector Gómez-Gauchía, Belén Díaz-Agudo, and Pedro A. González-Calero. "A Case Study of Structure Processing to Generate a Case Base". In: *Advances in Case-Based Reasoning*. Ed. by Peter Funk and Pedro A. González Calero. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 587–600. ISBN: 978-3-540-28631-8.

[37] B. Goodman and S. Flaxman. "European Union regulations on algorithmic decision-making and a "right to explanation"". In: *ArXiv e-prints* (June 2016). arXiv: 1606.08813 [stat.ML].

[38] Mingyang Gu. "Supporting Generalized Cases in Conversational CBR". In: *MICAI 2005: Advances in Artificial Intelligence*. Ed. by Alexander Gelbukh, Álvaro de Albornoz, and Hugo Terashima-Marín. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 544–553. ISBN: 978-3-540-31653-4.

[39] Riccardo Guidotti et al. "A Survey Of Methods For Explaining Black Box Models". In: *CoRR* abs/1802.01933 (2018). arXiv: 1802.01933. URL: http://arxiv.org/abs/1802.01933.

[40] Riccardo Guidotti et al. "Local Rule-Based Explanations of Black Box Decision Systems". In: *CoRR* abs/1805.10820 (2018). arXiv: 1805.10820. URL: http://arxiv.org/abs/1805.10820.

[41] Odd Erik Gundersen and Sigbjørn Kjensmo. "State of the art: Reproducibility in artificial intelligence". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[42] Trevor J. Hastie. *Statistical Models in S. A Guide for Making Black Box Models Explainable*. https://www.taylorfrancis.com/books/9781351414234, 1991, 249—307.

[43] Andreas Henelius et al. "A peek into the black box: exploring classifiers by randomization". In: *Data Mining and Knowledge Discovery* 28.5 (Sept. 2014), pp. 1503–1529. ISSN: 1573-756X. DOI: 10.1007/s10618-014-0368-8. URL: https://doi.org/10.1007/s10618-014-0368-8.

[44] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. "Explaining Collaborative Filtering Recommendations". In: *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. CSCW '00. Philadelphia, Pennsylvania, USA: ACM, 2000, pp. 241–250. ISBN: 1-58113-222-0. DOI: 10.1145/358916.358995. URL: http://doi.acm.org/10.1145/358916.358995.

[45] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Comput.* 18.7 (July 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. URL: http://dx.doi.org/10.1162/neco.2006.18.7.1527.

[46] *IJCAI Workshop on Explainable Artificial Intelligence (XAI-17)*. Vol. 26. IJCAI. Melbourne, Australia, Aug. 2017.

[47] AI Index. *AI Index 2017 Annual Report*. 2017. URL: http://aiindex.org/2017-report.pdf.

[48] Jalil Kazemitabar et al. "Variable Importance Using Decision Trees". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 426–435. URL: http://papers.nips.cc/paper/6646-variable-importance-using-decision-trees.pdf.

[49] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. "Examples are not enough, learn to criticize! Criticism for Interpretability". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 2280–2288. URL: http://papers.nips.cc/paper/6300-examples-are-not-enough-learn-to-criticize-criticism-for-interpretability.pdf.

[50] Been Kim, Cynthia Rudin, and Julie A Shah. "The Bayesian Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 1952–1960. URL: http://papers.nips.cc/paper/5313-the-bayesian-case-model-a-generative-approach-for-case-based-reasoning-and-prototype-classification.pdf.

[51] Viktoriya Krakovna and Finale Doshi-Velez. "Increasing the interpretability of recurrent neural networks using hidden Markov models". In: *arXiv preprint arXiv:1606.05320* (2016).

[52] Sanjay Krishnan and Eugene Wu. "PALM: Machine Learning Explanations For Iterative Debugging". In: *Proceedings of the 2Nd Workshop on Human-In-the-Loop Data Analytics*. HILDA'17. New York, NY, USA: ACM, 2017, 4:1–4:6. ISBN: 978-1-4503-5029-7. DOI: 10.1145/3077257.3077271. URL: http://doi.acm.org/10.1145/3077257.3077271.

[53] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. "Interpretable decision sets: A joint framework for description and prediction". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1675–1684.

[54] David Leake and Matthew Whitehead. "Case Provenance: The Value of Remembering Case Sources". In: *Case-Based Reasoning Research and Development*. Ed. by Rosina O. Weber and Michael M. Richter. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 194–208. ISBN: 978-3-540-74141-1.

[55] David B Leake. "Abduction, experience, and goals: A model of everyday abductive explanation". In: *Journal of Experimental & Theoretical Artificial Intelligence* 7.4 (1995), pp. 407–428.

[56] Tao Lei, Regina Barzilay, and Tommi Jaakkola. "Rationalizing Neural Predictions". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016, pp. 107–117. DOI: 10.18653/v1/D16-1011. URL: http://aclweb.org/anthology/D16-1011.

[57] Oscar Li et al. "Deep Learning for Case-based Reasoning through Prototypes: A Neural Network that Explains its Predictions". In: *CoRR* abs/1710.04806 (2017). arXiv: 1710.04806. URL: http://arxiv.org/abs/1710.04806.

[58] Zachary Chase Lipton. "The Mythos of Model Interpretability". In: *CoRR* abs/1606.03490 (2016). arXiv: 1606.03490. URL: http://arxiv.org/abs/1606.03490.

[59] Yin Lou et al. "Accurate Intelligible Models with Pairwise Interactions". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. Chicago, Illinois, USA: ACM, 2013, pp. 623–631. ISBN: 978-1-4503-2174-7. DOI: 10.1145/2487575.2487579. URL: http://doi.acm.org/10.1145/2487575.2487579.

[60] S. Lundberg and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions". In: *ArXiv e-prints* (May 2017). arXiv: 1705.07874 [cs.AI].

[61] Dmitry M. Malioutov et al. "Learning Interpretable Classification Rules with Boolean Compressed Sensing". In: *Transparent Data Mining for Big and Small Data*. Ed. by Tania Cerquitelli, Daniele Quercia, and Frank Pasquale. Cham: Springer International Publishing, 2017, pp. 95–121. ISBN: 978-3-319-54024-5. DOI: 10.1007/978-3-319-54024-5_5. URL: https://doi.org/10.1007/978-3-319-54024-5_5.

[62] D McSherry. "Explanation in Case-Based Reasoning: an Evidential Approach In Lees, B.(ed.) Proceedings of the 8th UK Workshop on Case-Based Reasoning 47–55". In: *Cambridge. Google Scholar* (2003).

[63] Natalie Meurer. *A Simple Guide to Entropy-Based Discretization*. 2015. URL: https://natmeurer.com/a-simple-guide-to-entropy-based-discretization/ (visited on 05/13/2019).

[64] Tim Miller, Piers Howe, and Liz Sonenberg. "Explainable AI: Beware of Inmates Running the Asylum". In: *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*. Springer. 2017, pp. 36–42.

[65] Mirjam Minor, ed. *ICCBR Workshop on Case-Based Reasoning for the Explanation of Intelligent Systems (2018)*. Vol. 26. ICCBR. Stockholm, Sweden, July 2018.

[66] Brent Mittelstadt, Luciano Floridi, and Sandra Wachter. "Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation". In: *International Data Privacy Law* 7.2 (June 2017), pp. 76–99. ISSN: 2044-3994. DOI: 10.1093/idpl/ipx005. eprint: http://oup.prod.sis.lan/idpl/article-pdf/7/2/76/17932196/ipx005.pdf. URL: https://dx.doi.org/10.1093/idpl/ipx005.

[67] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/, 2019.

[68] Ali Mousavi, Gautam Dasarathy, and Richard G Baraniuk. "Deepcodec: Adaptive sensing and recovery via deep convolutional neural networks". In: *arXiv preprint arXiv:1707.03386* (2017).

[69] Conor Nugent and Pádraig Cunningham. "A case-based explanation system for black-box systems". In: *Artificial Intelligence Review* 24.2 (2005), pp. 163–178.

[70] Conor Nugent and Padraig Cunningham. "A Case-Based Explanation System for Black-Box Systems". In: *Artif. Intell. Rev.* 24 (Oct. 2005), pp. 163–178. DOI: 10.1007/s10462-005-4609-5.

[71] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. "Model Agnostic Supervised Local Explanations". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 2515–2524. URL: http://papers.nips.cc/paper/7518-model-agnostic-supervised-local-explanations.pdf.

[72] G. Ras, M. van Gerven, and P. Haselager. "Explanation Methods in Deep Learning: Users, Values, Concerns and Challenges". In: *ArXiv e-prints* (Mar. 2018). arXiv: 1803.07517 [cs.AI].

[73] Gabrielle Ras, Pim Haselager, and Marcel van Gerven. "Explanation Methods in Deep Learning: Users, Values, Concerns and Challenges". In: *arXiv preprint arXiv:1803.07517* (2018).

[74] Juan A. Recio-García, Pedro A. González-Calero, and Belén Díaz-Agudo. "jcolibri2: A framework for building Case-based reasoning systems". In: *Science of Computer Programming* 79 (2014). Experimental Software and Toolkits (EST 4): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT-3 2010), pp. 126 –145. ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2012.04.002. URL: http://www.sciencedirect.com/science/article/pii/S0167642312000664.

[75] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-precision model-agnostic explanations". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018. URL: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982.

[76] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Nothing else matters: model-agnostic explanations by identifying prediction invariance". In: *arXiv preprint arXiv:1611.05817* (2016).

[77] Ariella Richardson and Avi Rosenfeld. "A Survey of Interpretability and Explainability in Human-Agent Systems". In: *XAI 2018* (), p. 137.

[78] Alvin E. Roth. "Introduction to the Shapley value". In: *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Ed. by Alvin E.Editor Roth. Cambridge University Press, 1988, 1–28. DOI: 10.1017/CBO9780511528446.002.

[79] Cynthia Rudin. "Please Stop Explaining Black Box Models for High Stakes Decisions". In: *arXiv e-prints*, arXiv:1811.10154 (Nov. 2018), arXiv:1811.10154. arXiv: 1811.10154 [stat.ML].

[80] Md. Kamruzzaman Sarker et al. "Explaining Trained Neural Networks with Semantic Web Technologies: First Steps". In: *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017*. Ed. by Tarek R. Besold, Artur S. d'Avila Garcez, and Isaac Noble. Vol. 2003. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-2003/NeSy17_paper4.pdf.

[81] M. Sato and H. Tsukimoto. "Rule extraction from neural networks via decision tree induction". In: *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*. Vol. 3. July 2001, 1870–1875 vol.3. DOI: 10.1109/IJCNN.2001.938448.

[82] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences". In: *CoRR* abs/1704.02685 (2017). arXiv: 1704.02685. URL: http://arxiv.org/abs/1704.02685.

[83] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), p. 484.

[84] Barry Smyth and Mark T. Keane. "Remembering to Forget: A Competence-preserving Case Deletion Policy for Case-based Reasoning Systems". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., 1995, pp. 377–382. ISBN: 1-55860-363-8, 978-1-558-60363-9. URL: http://dl.acm.org/citation.cfm?id=1625855.1625905.

[85] Armin Stahl. "Learning Similarity Measures: A Formal View Based on a Generalized CBR Model". In: *Case-Based Reasoning Research and Development*. Ed. by Héctor Muñoz-Ávila and Francesco Ricci. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 507–521. ISBN: 978-3-540-31855-2.

[86] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3319–3328.

[87] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. "MoviExplain: A recommender system with explanations". In: *RecSys'09 - Proceedings of the 3rd ACM Conference on Recommender Systems* (Jan. 2009), pp. 317–320. DOI: 10.1145/1639714.1639777.

[88] Frode Sørmo, Jorg Cassens, and Agnar Aamodt. "Explanation in Case-Based Reasoning—Perspectives and Goals". In: *Artif. Intell. Rev.* 24.2 (Oct. 2005), pp. 109–143. ISSN: 0269-2821. DOI: 10.1007/s10462-005-4607-7. URL: http://dx.doi.org/10.1007/s10462-005-4607-7.

[89] TensorFlow. *A Neural Network Playground*. (Accessed on 12/05/2018). URL: http://playground.tensorflow.org/.

[90] Richard Tomsett et al. "Interpretable to Whom? A Role-based Model for Analyzing Interpretable Machine Learning Systems". In: (June 2018).

[91] S. N. Tran and A. S. d'Avila Garcez. "Deep Logic Networks: Inserting and Extracting Knowledge From Deep Belief Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.2 (Feb. 2018), pp. 246–258. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2016.2603784.

[92] Michael Tsang et al. "Neural Interaction Transparency (NIT): Disentangling Learned Interactions for Improved Interpretability". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 5804–5813. URL: http://papers.nips.cc/paper/7822-neural-interaction-transparency-nit-disentangling-learned-interactions-for-improved-interpretability.pdf.

[93] M. Tulio Ribeiro, S. Singh, and C. Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *ArXiv e-prints* (Feb. 2016). arXiv: 1602.04938.

[94] Ryan Turner. "A Model Explanation System: Latest Updates and Extensions". In: *arXiv e-prints*, arXiv:1606.09517 (June 2016), arXiv:1606.09517. arXiv: 1606.09517 [stat.ML].

[95] Mark W. Craven and Jude W. Shavlik. "Extracting Tree-Structured Representations of Trained Networks". In: *Adv Neural Inform Process Syst* 8 (Jan. 1999).

[96] Chih-Kuan Yeh et al. "How Sensitive are Sensitivity-Based Explanations?" In: *CoRR* abs/1901.09392 (2019). arXiv: 1901.09392. URL: http://arxiv.org/abs/1901.09392.

[97] Nianyin Zeng et al. "Deep Belief Networks for Quantitative Analysis of a Gold Immunochromatographic Strip". In: *Cognitive Computation* 8.4 (Aug. 2016), pp. 684–692. ISSN: 1866-9964. DOI: 10.1007/s12559-016-9404-x. URL: https://doi.org/10.1007/s12559-016-9404-x.

[98] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. "DeepRED – Rule Extraction from Deep Neural Networks". In: *Discovery Science*. Ed. by Toon Calders, Michelangelo Ceci, and Donato Malerba. Cham: Springer International Publishing, 2016, pp. 457–473. ISBN: 978-3-319-46307-0.

Sondre Engen & Praveen Perinparajan

Extending Model-Induction Methods by Case-Based Reasoning for Explaining Deep Neural Networks

NTNU
Norwegian University of
Science and Technology