Johannes Kvamme and Pål-Edward Larsen

Achieving Trustable Explanations Through Multi-Task Learning Neural Networks

Master's thesis in Informatics
Supervisor: Helge Langseth
June 2021

Master's thesis

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

NTNU
Kunnskap for en bedre verden

Johannes Kvamme and Pål-Edward Larsen

Achieving Trustable Explanations Through Multi-Task Learning Neural Networks

Master's thesis in Informatics
Supervisor: Helge Langseth
June 2021

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

NTNU

Kunnskap for en bedre verden

Abstract

Artificial intelligence is becoming more prominent in high-risk domains, such as criminal justice and health care, and as a result, legislature calls for insight into AI systems. This insight requires explanations that both grounds the decisions and allows us to learn from opaque box systems. The field of explainable artificial intelligence is gaining traction as a result, which aims to build trust, safety, and liability into artificial intelligence systems.

Previous literature shows several methods for generating explanations for artificial intelligence systems, but several questions remain. One of them is how we can trust these explanations. This thesis explores the current state-of-the-art of explainable artificial intelligence methods and designs an architecture based on multi-task learning, enabling pre-existing neural networks to add trustable explanations as a native part of the neural network. We argue for using explanations based on principles from social sciences in our architecture.

We present findings indicating that the architecture incorporates the positive qualities of a multi-task learner while providing explanations. We show that counterfactual explanations by domain experts can be used to amplify data to let multi-task learners excel on sparse data. Our novel loss function integrates the numerical sign difference between the gradient of the explanation and the gradient of the primary task. Through this loss, the architecture assures that all shared information is utilized similarly. As a result, one can gain increased trust in the explanations from the artificial intelligence system.

Sammendrag

Kunstig intelligens blir stadig mer tilstedeværende i høy-risiko domener slik som rettsvesen og medisin. Som en følge krever lovgivende makter innsikt i kunstig intelligens-systemer, samt forklaringer som både kan begrunne valg og åpne for læring fra ugjennomsiktige systemer. På bakgrunn av dette vokser fagfeltet *forklarbar kunstig intelligens*, som søker måter å bygge tillit, trygghet og ansvarlighet inn i kunstig intelligens-systemer.

Tidligere forsking presenterer flere metoder for å generere forklaringer for kunstig intelligens-systener, men flere spørsmål er fremdeles ubesvart. Et av disse er hvordan man kan stole på forklaringene man får. I denne masteroppgaven utforsker vi moderne forskning på metoder for forklarbar kunstig intelligens, og designer en arkitektur basert på fleroppgavelæring. Arkitekturen åpner for å legge til tillitsverdige forklaringer som en innebygd del av det kunstige nevrale nettverket. Vi argumenterer for å bruke forklaringer basert på prinsipper fra samfunnsvitenskap i vår arkitektur.

Vi presenterer funn som indikerer at arkitekturen beholder de positive kvalitetene ved fleroppgavelæring samtidig som den oppgir forklaringer. Vi viser at kontrastive forklaringer laget av domeneeksperter kan brukes til å utvide data slik at fleroppgavenettverk kan utmerke seg på små datamengder. Gjennom vår originale tapsfunksjon, som integrerer fortegnsforskjell mellom gradientene til forklaringen og hovedoppgaven, kan arkitekturen garantere at all delt informasjon blir brukt på tilnærmet lik måte. Som et resultat av dette kan man øke tilliten til forklaringene fra kunstig intelligens-systemet.

Preface

This thesis has been carried out at the Department of Computer Science at the Norwegian University of Science and Technology from September 2020 to June 2021.

We want to thank our supervisor, Professor Helge Langseth at NTNU, for invaluable support, help, and feedback during the writing of our master thesis. Without Helge's feedback, this thesis would be a lesser work. We thank you for all the hours you have spent helping us weekly.

We also want to thank Ph.D. student Yanzhe Bekkemoen for the help in discussing and developing the Friendly-Enigma architecture, EXAIGON group at NTNU for great discussions, Professor Tim Miller of the University of Melbourne for taking his time to answer our questions, and our good friends at the office for bouncing of ideas throughout the year of researching. Finally, we want to thank Astrid Tonstad and Benedicte Helen Myrvoll for all their support.

Johannes Kvamme and Pål-Edward Larsen

Trondheim, May 28, 2021

Contents

List of Figures

List of Tables

List of acronyms

AI	Artificial Intelligence
XAI	eXplainable Artificial Intelligence
MTL	Multi-Task Learner
STL	Single-Task Learner
GDPR	General Data Protection Regulation
EU	European Union
AI HLEG	High-Level Expert Group on Artificial Intelligence
AAAI	Association for the Advancement of Artificial Intelligence
ASV	Asymmetric Shapley Values
LIME	Local Interpretable Model-agnostic Explanations
ReLU	Rectified Linear Unit
MNIST	Modified National Institute of Standards and Technology
SEDC	Heuristic Best-First Search Algorithm for Finding Evidence Counterfactuals
SHAP	SHapley Additive exPlanations
VAE	Variational Autoencoder

C-CHVAE Counterfactual Conditional Heterogeneous Autoencoder

ELBO Evidence Lower Bound

TED Teaching Explainable Decisions

AUROC Area Under the Receiver Operating Characteristic (curve)

MTL-R Multi-Task Learner: Regular

MTL-FE Multi-Task Learner: Friendly-Enigma

HELOC Home-Equity Line Of Credit

GMSC Give Me Some Credit

Chapter 1

Introduction

In this chapter we will present the motivation for our research and recent advancements in explainable artificial intelligence. Many governments are creating and pushing for transparency in their artificial intelligence legislation, which is a large motivator for institutions to further research explainability.

Section 1.1 will introduce these advancements and legislation. In Section 1.2 the goal of this thesis and research questions that contributes to the goal are presented. The contributions are summarized in Section 1.3. The structure of the rest of the thesis is presented in Section 1.4.

1.1 Background and Motivation

Artificial intelligence (AI) is becoming more and more prominent in our lives, notably the (in)famous opaque-box AI neural networks. Neural networks have proved their proficiency and efficiency in many areas in recent years. AI systems are used for many tasks, some that remove tedious and mundane work from humans, others for recommending movies to watch on Netflix, and some are used in criminal justice, like finding potentially dangerous individuals [Završnik, 2020]. Another example is Amazon's same-day delivery system that by mistake routinely excluded black neighborhoods in the USA [Letzter, 2016]. According to Zolas et al. [2020], 24.8% of businesses with at least 250 employees have incorporated AI systems in some way. However, only 2.8% of all businesses use AI systems as a whole. In their article, Gerbert et al. [2017] says that 85% of executives believe that AI systems will give their businesses an advantage. Computing power is

becoming cheaper and more available, AI systems are better and easier to use, and there is a will from the business executives to use AI. Thus, there is little reason to believe that the usage of AI systems will slow down but only keep growing.

As AI is introduced into fields where its decisions will affect other humans or the environment, so-called high-risk decisions, it is crucial to trust these systems, trust that their decisions are sound, and understand their decision-making process. Some high-risk fields in which AI systems are being introduced to, or already exist in, are self-driving cars, personalized medicine, banking loans, and the criminal justice system [Guidotti et al., 2018]. A big part of trusting opaque-box AI is to understand why they made their decisions. For humans, this understanding often comes in the form of an explanation. In AI, this is the field of eXplainable AI (XAI), creating AI systems capable of creating explanations that humans can understand.

XAI has grown more popular in recent years. Many methods of explanation have been tested and tailored for different types of machine learning methods. Guidotti et al. [2018] argues for the need for explainable models to be more prominent than earlier, as the European Union's General Data Protection Regulation (GDPR) granting individuals the right to a "meaningful explanation of the logic involved when automated decision making takes place." Trust, safety, and liability are also mentioned as reasons behind the need for XAI, now that Big Data is more available and trained models are used for new purposes. Recently, Google launched their platform Vertex AI which incorporates Explainable AI [Lardinois, 2021] to try to respond to the growing popularity of AI and the following need for XAI.

Today, many modern AI systems and implementations are based on neural networks due to their superiority and success in many fields. Neural networks are incredibly complex, to the point where it is often impossible to understand the reasoning behind the network's decision, which is why they are named opaque boxes. Comparatively, decision trees are interpretable and understandable; it is possible to follow a branch path of the tree to see how a given input became a specific output. These attributes make decision trees trustworthy to any user, as they can easily understand the decision tree's reasoning by following the decision path. However, neural networks are still in use since they are applicable to many domains where decision trees cannot be used, e.g., audio and images. To further advance neural networks and their many descendants, it is crucial to understand them better. As a human brain's neurons heavily inspire neural networks, a better understanding of neural networks can also bring a better understanding of our brains and how they work.

In response to a growing opaque-box society, many nations and governments have begun putting laws into action to set restrictions on AI systems. As mentioned, the most notable is the General Data Protection Regulation implemented by the European Union (EU) on the 25th of May, 2018. GDPR is intended to give individuals within the EU protection of their data. GDPR article 15, section 1 [European Parliament and Council of the European Union, 2018] addresses the rights to the information an individual can demand. Subsection 1-h specifies these rights with regards to AI systems, which says:

> (1) The data subject shall have the right to obtain ... the following information:
>
> (h): the existence of automated decision-making, including profiling, referred to in Article 22(1) and (4) and, at least in those cases, meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing for the data subject.

An attachment to this article is Recital 71, where paragraph 4 states

> In any case, such processing should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached after such assessment and to challenge the decision.

The recital is meant as an attachment for giving insights into how the article was meant to be interpreted, but it is not legally binding. This article and recital impose heavy responsibilities on companies limiting opaque-box models like neural networks used in critical human decision systems within the EU. A typical example is the bank loan example where Jane, a woman who wants to get a loan to purchase a house, is denied the application. Jane tries to get a loan from the bank Bank AI but is denied the loan. Jane asks Bank AI why the loan was rejected. Since Bank AI has implemented a neural network to make the decision, Bank AI cannot tell Jane the exact reasoning behind the rejection due to Bank AI not understanding why their neural network made the decision. Bank AI is then legally obliged to give Jane meaningful information about the logic involved and urged to present explanations for the given decision. Failure to comply with an article can result in severe fines. From its implementation until September 2020, the GDPR fines have accumulated to an incredible sum of € 491,063,290 [CMS, 2019]. Due to these new regulations, there is a significant motivation within the EU to develop explainable AI.

The High-Level Expert Group on Artificial Intelligence (AI HLEG) was created and assembled in June 2018. This group's objective is to support the implementation of the European Strategy on Artificial Intelligence. AI HLEG focuses mainly on the human-centric approach to AI. They have created a list of key requirements for trustworthy AI systems and a set of recommendations to achieve trustworthy AI with the focus of protecting humans.

Focus on explanations is not limited to the European Union and AI HLEG. Very recently, the American Federal Trade Commission published a blog post regarding truth, fairness, and equity in AI, and they stress the need for transparency [Jillson, 2021]. As this thesis and authors originate from Norway, there are also guidelines created by the Norwegian government, a national strategy for artificial intelligence [Ministry of Local Government and Modernisation, 2020]. This strategy is supposed to be a framework for both the private and public sectors. Furthermore, it emphasizes the importance of EU's strategy and states that the Norwegian use and development of AI should be grounded in the principles of EU's strategy.

There are currently advocates for stopping further development of opaque-box models [Rudin, 2019], e.g., neural networks or tree ensembles. The argument is that we cannot risk faulty explanations. If a model is created that explains the predictor, there is no reason for the existence of the predictor as the explanator has all the same capabilities. However, opaque-box models continue to prove very efficient and give good results with little data analysis and customization. This then raises the question of explaining the opaque box.

1.2 Goal and Research Questions

The usage of neural networks will not diminish unless some new innovative solution will outperform these networks. Neural networks will learn to fit the data, though the data might be biased. It is thus crucial to get insight to help debug and understand the systems that are used. To increase the transparency of neural networks, we want to explore the possibility of adding explanations in these opaque-box machine learning algorithms. Therefore, the goal of this thesis is to:

Goal Create adoptable and inherently trustable explanations as a native part of a neural network

When we create explanations for a model, we want to learn from its knowledge and trust its decisions. If we trust its decisions, it is vital that we also trust the

explanations themselves. As requirements for explainable AI are increasing, it is important that all AI systems adhere to these requirements. As such, explanation systems have to be easily adoptable. To reach this goal, we define the following research questions:

Research question 1 What is the current state-of-the-art of explainable AI?

Research question 2 How does the task of explaining couple with the task of classification or prediction?

If the tasks are coupled, the network's parameters should be able to represent the prediction and the explanation at the same time. Thus explanations are achieved as a native part of the neural network.

Research question 3 Can the explanations for neural networks with explanation capability be generated through existing methods?

Manually creating explanations for a dataset can be time-consuming. If explanations are to be adoptable, they have to be easily accessible. Thus the question is whether they can be automatically generated.

Research question 4 How can you trust the explanation capability of the network?

One purpose of explanations is to build trust towards a system. If the explanations themselves are not trustable, they fail this purpose. Thus, we will look into how to achieve trustable explanation capabilities.

Research question 5 Can the task of prediction be learned from sparser data by adding a task of explanation?

Acquiring data for some domains can be difficult, which makes sparse data common in the real world. If explanations can be added to a system to increase the performance on sparse data, they become increasingly adoptable.

1.3 Contributions

Based on the work in this thesis towards the research goal, we present several contributions to the fields of explainable AI built upon multi-task learning. These contributions further expand state-of-the-art research in the fields through our singular system with native, trustworthy explanations. Our findings and contributions are detailed in Section 6.2, but are summarized as:

1. A State-of-the-art Taxonomy
2. Multi-task Learning for Trustable Explanations in AI Systems
3. Trustworthy explanations based on a novel sign-difference loss
4. Explanations as a strengthened signal for sparse and noisy data outperforming traditional neural networks

1.4 Thesis Structure

Our thesis is structured as follows: Chapter 2 presents the background theory on explainable AI, multi-task learning, and the taxonomy used for the rest of the thesis. Together this forms the foundation to understand the domains of this research. Chapter 3 contains an overview of the state-of-the-art methods of which the research of this thesis will extend. Chapter 4 will introduce the Friendly-Enigma architecture, the multi-task learner deep neural network with trustworthy native explanations. In Chapter 5, the architecture and research questions are tested with respect to the overall goal to see to which degree the main research question is feasible. Chapter 6 will discuss the overall questions issued by the experiments, summarize our contributions and discuss future work. Finally, Chapter 7 will conclude this thesis.

Chapter 2

Background Theory and Taxonomy

In this chapter, we will present the taxonomy used in this thesis and the background theory necessary to get an understanding of explainable AI and the prominent field of multi-task learning, which our work will build on. Section 2.1 presents our taxonomy on explainable AI based on previous research and social sciences. Our taxonomy tree can be seen in Figure 2.1. In Section 2.2, we present the historical background of explainable AI, an introduction to the state of the art research and methods of explainable AI further presented in Chapter 3, and the theory behind multi-task learning.

2.1 Taxonomy

To understand the domain of eXplainable Artificial Intelligence, one must first establish a common understanding of related terms. The field XAI is one of many within the umbrella term Artificial Intelligence. XAI itself is a sub-field of another field within AI, trustworthy AI. What defines trustworthiness within AI is up for discussion. However, in this report, we will follow the definitions by the High-Level Expert Group on Artificial Intelligence, other papers, and conferences. AI HLEG has defined key requirements for achieving this as a part of their report [High-Level Expert Group on AI, 2019]. One of these requirements is transparency, under which explainability is filed.

Two key domain terms which are defined in various ways in explainable AI are explainability and interpretability. The Expert Group defines explainability as "the ability to explain both the technical processes of the AI system and the reasoning behind the decisions or predictions that the AI system makes.". However, they do not define how to determine whether this reasoning and process have been explained.

In the last few years, there has been a tutorial or workshop on the topic of XAI at the AAAI (Triple AI) conferences. At this conference, the speakers Gade et al. [2020] presented their talk on XAI, where they define Trustable AI as a combination of valid AI, privacy-preserving AI, responsible AI, and explainable AI. Other researches that try to define these terms are Barredo Arrieta et al. [2020]. They include privacy and security as a part of Responsible AI, their umbrella term similar to Trustworthy AI, and place the audience as a core part of explainability.

Doshi-Velez and Kim [2017] defines interpretability as "the ability to explain or to present in understandable terms to a human.". They state that explanations are hard to define, albeit not delving more into the topic, but note that other communities are trying to define this. Guidotti et al. [2018] narrows Doshi-Velez and Kim's definition of interpretability by stating that concepts in a given explanation are self-explanatory, and need no further explanation. This removes the aspect of a continued process of explaining and narrows interpretability to be "the ability to present knowledge and reason in understandable terms to a human.".

Another definition of these key terms is defined by Rudin [2019]. Rudin proposes a call to arms to stop using explainable machine learning and start creating interpretable machine learning instead. Rudin says that a second model to explain another model (post-hoc) *must* be wrong to some degree, or else it would be an exact copy of the original model. One can interpret her definition of explainable as "the ability to be explained by any external agent.". Summarized, Rudin's claim for her call to arms is that interpretable models are understandable by design and believes that using a second model to explain another model is inherently flawed. If a post-hoc model is often wrong, applying the model to a high-risk domain might have dire consequences, e.g., using one to incarcerate a criminal may end up jailing one innocent person out of five. It is, therefore, ethically not suitable to use these post-hoc models on highly critical decisions. Rudin's definition of *explainable* and *interpretable* does not fit the definitions of Guidotti et al. [2018] and requirements by Miller [2019]. Thus, we define interpretability similar to Guidotti et al. [2018], as the "ability to present in understandable terms to a human.". There are several ways to present something understandable, either by

using a transparent model that humans understand or presenting an explanation that fits the prediction or classification that is understandable.

Another important term, and corresponding antonym, are local and global interpretability. We use the term from Doshi-Velez and Kim [2017]:

> Global interpretability implies knowing what patterns are present in general (such as key features governing galaxy formation), while local interpretability implies knowing the reasons for a specific decision (such as why a particular loan application was rejected). The former may be important for when scientific understanding or bias detection is the goal; the latter when one needs a justification for a specific decision.

The other key term in XAI is explainability. We define explainability based on Miller [2019], further discussed in Section 2.1.2, as "capable of participating in a social process between two or more participants in which the system transfers knowledge to the explainee.". This knowledge can be both global or local knowledge. This definition is more strict than others in the XAI community. It allows the explainee (the user or learner) to question the decision of the explainer (the AI system) and receive a new response.

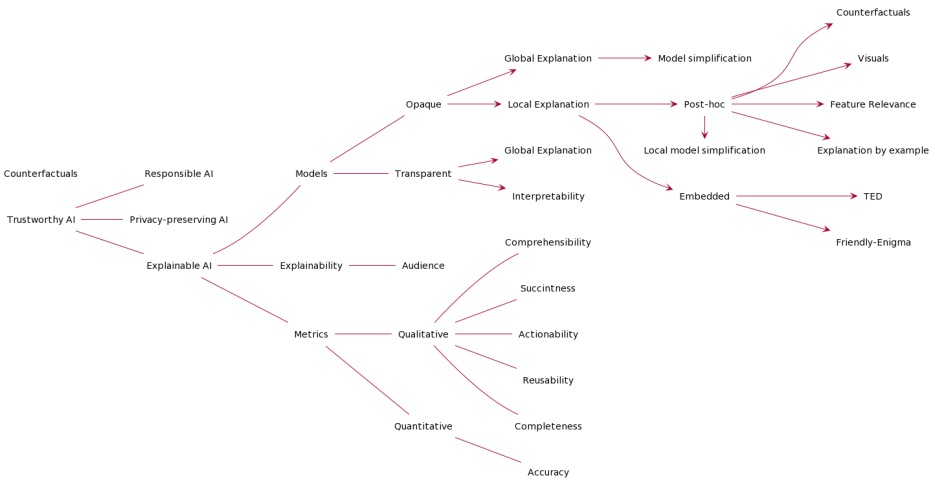

Figure 2.1: Taxonomy tree

2.1.1 Explanation Methods

We use the definitions of explanation methods from Barredo Arrieta et al. [2020]. These are summarized as:

- **Text explanations** which a model generates to explain the model's rationale. It is done by creating an explanation in a formal language comprehensible by humans, e.g., English.
- **Visual explanation** visualizes the model's behavior or reasoning. Visual explanation is a very prevalent method of explaining, especially within convolutional neural networks. Some of the most common methods are using saliency masks or heatmaps.
- **Local explanations** generates explanations by segmenting the model into different subspaces which are easier to interpret.
- **Explanations by example** are giving similar prediction inputs or prototypes built from the current classification such that the receiver can try to connect the similarities themselves.
- **Explanations by simplification** is the technique of building an interpretable, simpler model built from the output of the original model.
- **Feature relevance explanation** is explaining through generating an importance value, or a score, for each feature. This value reflects how much the different features impacted the final decision.

2.1.2 Explanations in Social Science

The XAI field is impaired by the lack of a sound definition of what an explanation is. There has been a lot of discourse about the subject, dating as far back as Aristotle. Many AI scientists try to develop explainable AI models without knowing how the models should explain or what a sufficient explanation is. Guidotti et al. [2018] provides a survey of opaque-box explanation methods and divides opaque-box explanation into several subcategories. These are the opaque-box model explanation category, opaque-box outcome category, opaque-box inspection category, and transparent box design category. These are the categories that the authors divide the different communities' perspectives on explaining machine learning models into. They conclude that one of the most critical problems is a lack of an agreement on what an explanation is.

Professor Tim Miller at the University of Melbourne addresses the issue of a

standard definition of an explanation from Guidotti et al. [2018] and grounds explanation in both cognitive, social, and philosophical science. He also connects how each part of an explanation can, and maybe should, be used within XAI software. Miller argues that an explanation is a combination of two processes and a product.

- The first process, the cognitive process, is the process of inferring an explanation (explanans) given a certain question to an event (explanandum). This process is to infer the most crucial attributes that cause an event, e.g., even though the big bang is crucial for why the building is burning, it is not included in an explanation.

- The explanation, the product, is the result of the cognitive explanation process.

- The second process, the social process, is the process of the explainer trying to transfer their knowledge to an explainee or a group of explainees. One of several possible goals within this process is that the explainer has a good enough understanding to reason about the attributes of the explanandum correctly. If the explainer cannot identify the correct reason, the explainer will misrepresent the event, and the explainee may believe it is the truth.

Miller evaluates current works similarly to Guidotti et al. [2018] and states that the works base their definition of explanation on the authors' intuitions. Four findings are then proposed to be necessary to XAI, which most works do not include. These findings are as follows:

Explanations are contrastive - When people ask for an explanation to event P, they implicitly mean "Why did P happen instead of Q?". The answer to this question is a contrastive explanation.

Explanations are biasedly selected - Humans pick a few causes for an event as the explanation is based on certain cognitive biases.

Probabilities probably do not matter - Although likelihood is important in an explanation and their probabilities do matter, actually referring to probabilities is less effective in explaining an event than referring to the causes of the event.

Explanations are social - Explanations are based on how the different parties perceive each other and what each party expects the other party to believe. Explanations are a part of a conversation or interaction.

These are summarized in the explanation flow as:

> While an event may have many causes, often the explainee cares only about a small subset (relevant to the context), the explainer selects a subset of this subset (based on several different criteria), and explainer and explainee may interact and argue about this explanation.

Miller continues by proposing this explanation flow as imperative for truly explainable AI systems. Guidotti et al. [2018] focuses on interpretability. They use a definition from Doshi-Velez and Kim [2017] that states interpretability as "the ability to explain or to provide the meaning in understandable terms to a human.". They continue by pointing out that "this definition assumes implicitly that concepts expressed in the understandable terms composing an explanation are self-contained and do not need further explanations." This directly opposes Miller's finding of explanations being social and part of interaction or conversation.

2.1.3 Contrastive Explanations

According to Miller, contrastive explanations are among the most important findings within philosophical and cognitive sciences. When a person, an explainer, explains an event P, they do not explain everything between the heavens and earth that has happened to reach P. The explainer will explain event P in relation to some other event, familiar or closely related event Q. Q is a fictional event that did not occur but might have occurred in a different event. This relational method of explaining events may be either intended, where Q is provided explicitly by the explainer, or implicitly, where Q is not provided by the explainer but is innate in the explanation given by the explainer. A question where Q is implicit is the question "Why is that person running?" where the Q is "instead of walking".

Lipton [1990] introduced the terminology fact and foil to be the events P and Q, respectively. The fact was the event that occurred, and the foil was the fictional, related event. Cause C is the reason for event P occurring, while a counterfactual is the reason for the foil Q of occurring instead of fact P. The counterfactual case can also be seen as the reason for event P not occurring. From this Section and onward, we will adopt this definition of counterfactual. An example of this is when a person is running to reach the bus. A question that might occur is "Why did that person run instead of walk?". In this question, event P (the fact) is that the person was running to the bus, and event Q (the foil) is that the person was walking to the bus. In this case, we say that the fact's cause C is that "The bus arrived earlier than expected.". The foil's cause, the counterfactual, is then

"The bus arrived on time.". The contrastive explanation to the question "Why did that person run instead of walk?" would be "The person would have walked if the bus arrived on time.".

According to Miller [2019], most authors in the field say that why-questions are always asking for a contrastive explanation, either directly or implicitly. When the foil is included implicitly, people are good at inducing what the foil is meant to be. To many XAI researchers, this is a key argument for why XAI models should be creating contrastive explanations, as it fits the mental model for humans to learn why an event did (or did not) occur. For people to learn, the explainee must understand the foil. A contrastive explanation might be insufficient if the explainee misunderstood the foil.

2.2 Background Theory

There has been much research that focus on comparing explanation methods [Guidotti et al., 2018; Barredo Arrieta et al., 2020]. This is a difficult task as many papers use different metrics for measuring success. No metrics have been agreed upon as good metrics to use in explainability, and as such, which metrics to use is left up to each paper. Two speakers from Accenture, Lecue and Wan [2018], presented six metrics that were to be used for explanations in XAI. The metrics are supposed to help define what is not and what is a valid explanation. The metrics are:

- **Comprehensibility** - How much effort is needed for a human to interpret it?
- **Succinctness** - How concise is it?
- **Actionable** - How actionable is the explanation? What can we do with it?
- **Reusability** - Could it be interpreted and reused by another AI system?
- **Accuracy** - How accurate is the explanation?
- **Completeness** - Does the "explanation" explain the decision completely or only partially?

However, these metrics are meant not for optimization and development but verification of explanations. There are yet to be developed metrics meant for measuring and comparing the explanation capability between models and methods. Without qualitative research involving people to evaluate the explanations in

terms of Lecue and Wan [2018]'s metrics, it is hard to compare these state-of-the-art methods against each other. Also, note that the metric **Accuracy** is reused here and should not be confused with the existing metric used to measure the prediction or classification capability of AI- and machine learning systems.

As Lecue and Wan states, explanations are qualitative by nature, and some things are harder to explain. Many AI models are very creative in the way that they go for solutions that a human may disagree with upfront. Like the famous move 37 in the second game of Go between the AI, AlphaGo, and the 18-time world champion, Lee Sedol [BBC, 2016]. The move was calculated to be a 1 in 10 000 chance that a human professional would make, a very obscure move. Move 37 was later deemed the catalyst for the second game, ending in a win for AlphaGo.

To know AlphaGo's reasoning for why move 37 was the selected move would be invaluable for the rest of the Go community. This desire for an explanation is often shared between users of AI systems. Decisions such as move 37 can not be blindly trusted in highly critical situations if their outcomes will affect human lives. If a medical AI system believes that a patient has an illness, and the system proposes a treatment with possibly dangerous side effects, the AI's treatment will not be pursued if the human doctors disagree with the decision. The possibility exists that this medical AI system found its 'move 37', but due to the gravity of the domain, it cannot be trusted. This is where explainable AI systems are essential and showcase why they are necessary.

AI systems must therefore not be limited to what a human can understand or explain. A vital part of the explanation process, as Miller states, is that the explainer needs to have a sufficient understanding of the topic at hand. Teaching Explainable Decisions (TED) is a framework created by IBM [Hind et al., 2019], an embedded method that produces both the prediction and the explanation as a tuple. TED has gained some traction and has been used to prove explainability within AI [Codella et al., 2019]. TED requires a domain expert to explain alongside the training examples, which is then used to train a model. As move 37 was a move no professional Go player would likely make, it should suffice to say that the move would not be adequately explained by the expert because it would often be viewed as the wrong move. Human understanding should not be a limitation to what a machine can understand, and in TED's case, a human will impose their understanding of the world onto the machine. In many cases, a machine can find and see the intricacies of a domain that a human may not, as their view may be limited due to their understanding of the subject's inner workings. It is therefore crucial that the machine is given the tools to explain its worldview.

2.2.1 Multi-Task Learning

In this thesis, we will build upon ideas from multi-task learning. Multi-task learning is a popular approach to neural networks that allows for learning multiple tasks simultaneously. The purpose of machine learning is to create an AI system that can excel at a specific task. The tasks they can be created for are endless, e.g., image recognition, predicting the weather, and reading and speaking a language. In many tasks, it is desirable that the AI system can predict or classify more than one thing. A classic example is to classify an object in an image, e.g., a cat or a dog, and where it is, e.g., below the table. One possibility is to create two models; one for learning what the object is and a second for learning where it is. Each of these models is a single-task learner. A single-task learner (STL) is a neural network that only uses a single loss function to learn. The other method is to use multi-task learning, a neural network with a loss consisting of multiple loss functions with a weighted sum. Multi-task learning builds upon the idea of transfer learning. Sequential transfer learning for single-task learners was proposed in 1991 by Pratt et al., which tested if weights from a task could be used as bias for a connected task.

Senior Principal Researcher Dr. Rich Caruana [1997] proposed and popularized MTL in neural networks. He introduced multiple clear motivations on why MTLs should be used instead of STLs. An essential fact about learning is that people use the knowledge they have previously learned and apply them to related tasks. If someone has only encountered a door with a door handle (vertical push to open), they can use this knowledge when they encounter a door with a knob (horizontal twist to open). This is the inductive bias Caruana [1997] talks about, which is a clear benefit of using MTLs. The inductive transfer can be used to improve the generalized accuracy of the model's prediction, the speed of learning from the model (less data), and the intelligibility of the model. Caruana argues that most real-world problems are multi-task problems and should not be treated and solved by an STL as the performance gained from the MTL will be lost by multiple STLs trained to solve the multi-task problems.

What makes MTLs work better than their STL counterparts is how they share their learned parameters between layers. As mentioned, multi-task learning is popular in deep neural networks, neural networks with more than one hidden layer. To achieve multi-task learning in deep neural networks, there exist two main methods for creating MTLs, *hard parameter sharing* or *soft parameter sharing*. These two main methods and the positive attributes are presented onward as individual sections.

Faster Learning by Data Amplification

Faster learning in this sense means that the MTLs learn the same, or better, from fewer training samples than STLs do. The data amplification is a virtual amplification of the dataset by the noise that occurs when training. That noise may improve the learning of a neural network is a heavily researched subject [Patel and Kosko, 2009; Schaefferkoetter et al., 2020; Audhkhasi et al., 2013].

MTLs' faster learning is contributed to multiple signals and sources of noise, one for each head [Caruana, 1997]. The goal of each loss function is that they will learn a good representation of their given task. Given that the data shares a hidden layer representation, the MTL will be able to utilize this to better average the layer features. This is how an MTL will learn faster than STL, and in essence, amplify the dataset.

Increased Accuracy by Eavesdropping

Caruana [1997] showed that MTLs increase accuracy by eavesdropping. This is explained with a dataset where a hidden layer feature F exists that will help a neural network predict on task T and a second task T', and that F is easy to learn when training a net on task T but is hard to learn on task T'. If there are two STLs, one learning T and the other learning T', the STL learning T will likely learn F, but the other will likely not as F is hard to learn through T', ending in a worse prediction. If an MTL is learning both T and T', there is a higher chance that the shared layer F will be learned. Then task T' may eavesdrop on what T learned by proxy through F and learn better.

A real-world example of this could be two teachers teaching math. The curriculum F is similar for both teachers, T and T'. Teacher T teaches math through problems (good), and T' teaches math with powerpoints (bad). A student learning from T will learn the curriculum F well, while a student learning from T' is less likely to learn the curriculum. A student able to attend both classes is more likely to learn the curriculum F than the other students.

Improved Generalization by Bias

Caruana showed that MTLs learn hidden layer representations that are shared between tasks, allowing the net to better generalize on tasks and reduce overfitting. This is shown by having two tasks, T and T', which have two local minima A & B, and A & C respectively, where A is the same, shared local minima. Each task would perform well if the net entered either A or B for task T and A or C for task T'. Caruana showed that STLs trained on T and T' are equally as likely to fall into either one of the local minima. However, when an MTL is trained on

both T and T', it is highly likely to fall into A, showing that the MTL will learn hidden layers that several tasks represent.

Caruana ran a second experiment, where task T was updated to have a solid inclination to lead to minima B instead of A, the shared minima, while T' would still have no preference between the minima A and B. When MTLs are trained on the new task, it showed that the MTL trained on T would choose B, as expected, but the other trained on T' would have a high chance to fall into C than A, showing that MTL tasks do *not* prefer to use hidden layers that other tasks do *not* prefer either. When the MTL learns the common representation between tasks, it has learned a better method to generalize the tasks, and the chance to overfit on a specific task is reduced.

Learning Features that Matter

If a task T contains much noise and there exists a second task T', which share a common hidden layer F with T, an STL only learning T may have problems learning what features of the data is essential to the task and what is just noise and irrelevant to learn F. Caruana showed that an MTL will be more capable of selecting the features from T and T' that are both important and more likely to ignore irrelevant features in both T and T', making it more straightforward to learn F.

Hard Parameter Sharing

Hard parameter sharing has been the most common approach to MTLs and has been so for the past 20 years [Ruder, 2017] when introduced by Caruana. Hard parameter sharing is usually implemented by sharing a set of hidden layers of a deep neural network between all the tasks. At the same time, each head of the MTL has its output layer specific to its respective task. This can be seen in Figure 2.2.

As previously mentioned, using MTLs will reduce the risk of overfitting on the learned tasks. Baxter [1997] showed that the more related tasks learned by the MTL simultaneously, the smaller the risk for overfitting is. This is due to the bias that exists in all the heads, which will be learned by the shared hidden layers of the MTL. The more heads the MTL has, the more general the representation of the task has to become akin to Occam's Razor that says simpler theories are preferred to the more complex theories.

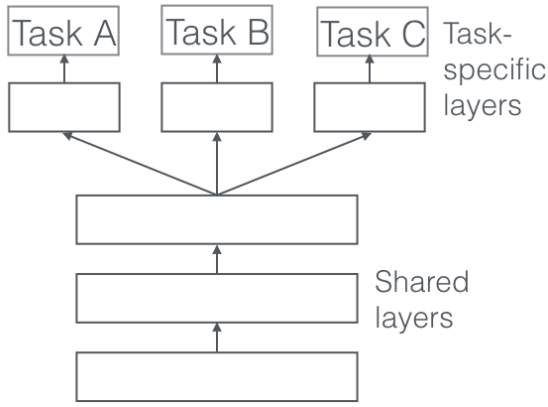

Figure 2.2: Hard parameter sharing for multi-task learning in deep neural networks [Ruder, 2017]

Soft Parameter Sharing

Soft parameter sharing is not as straightforward as hard parameter sharing. This is most likely why hard parameter sharing is still the most commonly used method. In soft parameter sharing, each head has its own model, as shown in Figure 2.3, compared to hard parameter sharing, where a set of hidden layers are shared. In hard parameter sharing, all the parameters are learned and stored as intrinsic values in the layers. In contrast, in soft parameter sharing, the values and distance must be explicitly shared between each head. There has been much research on how to develop several soft parameter sharing models [Ruder, 2017], where the distance between the models, which parameters to share, and how to share them are vital to improving the model.

Due to the vast and deep space of possible parameters to share between the layers, several recent soft parameter sharing models have focused on learning what parameters to share between themselves. Ruder et al. [2019] shows that learning what to learn can surpass hard parameter sharing with several percentage points.

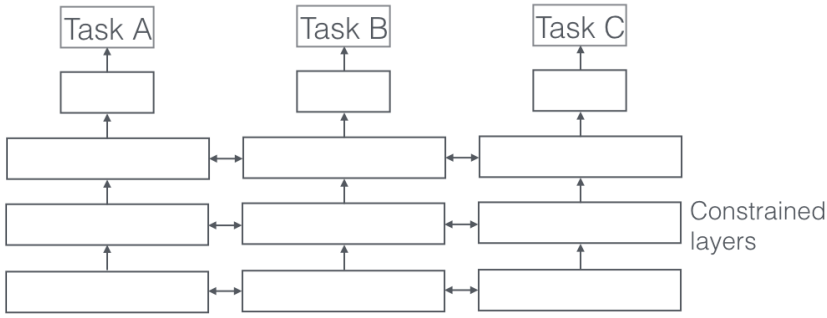

Figure 2.3: Soft parameter sharing for multi-task learning in deep neural networks [Ruder, 2017]

Chapter 3

State of the Art

This state-of-the-art section will present several of the most influential, important, and prominent models, frameworks, and methods developed within explainable AI. This section will give insight into how these methods work and their unique contributions to their respective fields.

Throughout this chapter, we denote X as the set of all training instances for a machine learning algorithm, where $x \in X$ such that x is a singular instance of X. The set of all labels that is possible to classify in an AI system is denoted L. The features of an instance will be denoted as $x = (x_1, x_2, ..., x_n)$, where n is the number of features in x.

As noted by the taxonomy in Section 2.1, the two approaches to explainable models are either using transparent models, such as shallow decision trees or explaining a complex model. Since complex models repetitively have given good results and are easily implemented, they are still popular. This then leaves the problem of explaining the complex models.

When explaining a complex model, it is common to explain an already existing AI system, which is called post-hoc explanations. Post-hoc explanations use an existing AI system and query the system to create an approximation of its inner workings, which is the basis for their explanations. The results of a post-hoc explanation are often a more coherent set of scores, heatmaps, natural language, or other more understandable representations of the systems.

The most obvious benefit of using a post-hoc method to generate explanations

is that one can use an already existing AI system without interrupting its use in any way. This way, an opaque model may become more transparent through post-hoc explanations.

The AI system that post-hoc methods create explanations for will be denoted as f throughout this chapter. Predicting or classifying x will be denoted as $f(x)$, which will either be the set of probabilities that x belongs to instance l for all instances in L or a predicted value based on x for regression problems. As x is the original representation of the instance being explained, let $x' = \text{bin}(x)$ denote the binary representation of x. For each feature value in x, if the value is 0, the representation is labeled as inactive (0). If it is non-zero, the representation is labeled active (1). $\text{bin}(x)$ is defined as Equation 3.1.

$$\text{bin}(x) = \forall i \in x \begin{cases} 1, i \neq 0 \\ 0, i = 0 \end{cases} \tag{3.1}$$

3.1 Feature Relevance

One way of explaining a complex models' output is based on the input values, i.e., attributing the outcome to the feature values to the input. This is also called "attribution" or "contribution" [Ancona et al., 2018]. Feature Relevance is one of the explanation method categories by Barredo Arrieta et al. [2020], mentioned in the taxonomy in Section 2.1.

3.1.1 Shapley Values

Shapley values define a game-theoretic method for calculating the contribution of each feature based on a given value function. The Shapley value is the feature value's marginal contribution to the value compared to the average value for the dataset. When using Shapley values in machine learning, the value function can be the probability that x belongs to a class y, $f(x, y)$ or a predicted real number for a regression model. Let then $h(x)$ be a value function on x such that $h(x) \in \mathbb{R}$. Let ϕ_j denote the contribution of feature value x_j of input x. The Shapley value of a feature value can then be written as $\phi_j(h)$, the contribution of feature value $x_j \in x$ on $h(x)$.

Let S be a partially ordered set in the set of all possible subsets without x_j, so that $S \subseteq \{x_1, \ldots, x_n\} \setminus \{x_j\}$. As the Shapley value is the average contribution of a feature value, the sum is multiplied by $\frac{1}{n!}$. Since $n!$ includes different orderings of features as different permutations, the set S is multiplied by the number of per-

mutations of length $|S|$ in the number $n!$. This is summarized as the normalizing constant k, denoted as Equation 3.2.

$$k = \frac{1}{n!} * |S|!\,(n - |S| - 1)! = \frac{|S|!\,(n - |S| - 1)!}{n!} \tag{3.2}$$

The formula for a feature values' Shapley value is the marginal contribution of adding x_j to the set S

$$\phi_j(h) = \sum_{S \subseteq \{x_1,\ldots,x_n\} \setminus \{x_j\}} k\,(h\,(S \cup \{x_j\}) - h(S)) \tag{3.3}$$

This means that a feature value's Shapley value is its marginal contribution to the result of the value function, averaged over all permutations in which the feature value can be included. This has the advantage of distributing the contributions fairly among the features such that the sum of contributions for a given feature set is equal to the difference of the prediction of that set to the prediction of the average set. This property is called the "Efficiency"-property [Molnar, 2019].

Calculating the Shapley value for all features is time-consuming, as the number of terms in the sum is 2^k, where k is the number of features. According to Faigle and Kern [1992], it is a #P-hard problem, problems that are famously hard to solve. When working with images, it is not unusual to have millions of features, as every pixel is a feature. Thus, multiple methods try to approximate the Shapley values of a model. One of these is presented in Section 3.1.4. Frye et al. [2020] also notes that Shapley values ignore causality and rely on fictitious data, as it substitutes feature values with other feature values in the data to be able to marginalize each feature. Another issue that Frye et al. states regarding Shapley values is that the explanations are provided in terms of values based on the input values x, which might not be directly interpreted. This is summarized as the four following issues of Shapley values:

1. Time-consuming
2. Ignore causality
3. Relying on fictitious data
4. Explanations are not directly interpretable

3.1.2 Asymmetric Shapley Values

Shapley values assign equal weight to all features, and their possible combinations through $\frac{|S|!(p-|S|-1)!}{p!}$. However, this fails to capture casual relations between features, as noted in issue 2 of Shapley values' issues. An example that Frye et al. [2020] introduces is "age" as a clear causal ancestor to "education". In normal Shapley values, they might have equal or similar values, while one could be interested in knowing the root cause only, i.e., the causal ancestors. To incorporate this in Shapley values, Frye et al. proposes Asymmetric Shapley Value (ASV), a relaxation of the symmetry-property of Shapley values. The symmetry property states that the Shapley value of two features are the same if both features contribute equally to all sets that they are a part of.

$$\begin{aligned} \textbf{Symmetry: } & \phi_j = \phi_k \text{ if } h(S \cup \{x_j\}) = h(S \cup \{x_k\}) \\ & \text{for all } S \subseteq \{x_1, \ldots, x_p\} \setminus \{x_j, x_k\} \end{aligned} \tag{3.4}$$

Another way Equation 3.3, the equation for the Shapley value of a feature value, can be found is by summing over all possible orderings of features in x, denoted R, instead of summing over the sets of features in x not including x_j. Although this set of orderings are bigger than the inclusion set $S \subseteq \{x_1, \ldots, x_n\} \setminus x_j$, one does not need to multiply by the number of sets of length $|S|$ in $n!$, as the sum is over all orderings which length is equal to $n!$. Let i be the set of features preceding x_j by ordering $r \in R$. For each ordering r, calculate the marginal contribution of x_j to i. The Shapley value of a feature value x_j is then denoted by Equation 3.5:

$$\phi_j(h) = \sum_{r \in R} \frac{1}{n!}(h(i \cup x_j) - h(i)) \tag{3.5}$$

Let $\Delta(R)$ be the set of probability measures on R. As Asymmetric Shapley values relaxes the symmetry-property, $\frac{1}{n!}$ is replaced by any probability measure w on the set of all permutations of features R and each $w \in \Delta(R)$ is a map $w : R \to [0, 1]$ satisfying $\sum_{r \in R} w(r) = 1$.

$$\phi_j^{(w)}(h) = \sum_{r \in R} w(r)[h(i \cup \{x_j\}) - h(i)] \tag{3.6}$$

If the distribution $\Delta(R)$ is uniform, Equation 3.6 is equivalent to Equation 3.3. This allows for defining distributions that incorporate causal knowledge in the calculations of the Shapley values instead of equal distribution. Frye et al. shows the following distribution as an example of a possible distribution with incorpo-

rated causal knowledge:

$$w_{distal}(r) \propto \begin{cases} 1 & \text{if } j \text{ precedes } i \text{ for any known ancestor } j \text{ of descendant } i \text{ under ordering } r \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The Asymmetric Shapley value for a root cause feature then indicates its effect on the result. At the same time, a descendant's ASV will be the difference it makes to the result, given that the root cause is already accounted for.

Asymmetric Shapley values are useful for scenarios with bijectively related features, where one is the deterministic causal ancestor of the other, e.g., native country being the deterministic causal ancestor of education when predicting an individuals income based on such features. Regular Shapley values with symmetry would assign these features the same Shapley value, while one could be interested in assigning all the importance to the ancestor instead. Frye et al. uses Asymmetric Shapley values to incorporate the effect of the causal ancestors of features in a dataset of census income and shows some root causes, like sex, affect other causes by a high degree. Another example they pose is unresolved discrimination in education applications. In this example, sex should only affect which department they apply for, not which departments they are accepted into, as sex cannot be altered. Here they assign a non-uniform distribution of values such that specific sensitive values like gender should be ordered after resolving values, in this case, "applied department", i.e., the Asymmetric Shapley value for gender is the contribution of gender given that department is already known. They create two datasets, one where gender is not directly affecting the outcome but passed through two features that will be accessible for the model, and one where they add a feature that will not be accessible for the model. The ASV's for these datasets then show that gender is directly influencing the outcome on the second dataset, meaning that there is unresolved discrimination in the data. This example shows that it is beneficial to be able to incorporate causality in Shapley values.

3.1.3 Local Interpretable Model-agnostic Explanations (LIME)

LIME was published by Ribeiro et al. [2016] and is quite popular in the field of XAI, with over 3800 registered citations on Semantic Scholar and 8400 stars on their Github repository [Ribeiro, 2016].

LIME is a model-agnostic post-hoc method that creates explanations by querying an AI system to create an interpretable local model that approximates the original system's logic. In this paper, LIME is categorized as Shapley values as LIME's output and interpretation fit the definition of Expectation Shapley values from

[Lundberg and Lee, 2016]. These methods are explained in Section 2.1.1. As such, it addresses the issue of attributing the model's outcome to the feature values of an input, as well as making an interpretable local model which gives insight into the original models' decisions, similar to the underlying issue of Section 3.2. In addition to being categorized as a feature relevance explanation method, LIME's interpretable local model is also categorized as a local explanation method.

LIME uses the terms local and global fidelity, which are very similar to the terms local and global explanations from Section 2.1. Global fidelity means that it is possible to explain the entire model. Local fidelity means that the explanation is meaningful for the instance that is being predicted. Global fidelity implies local fidelity, although local fidelity does not necessarily imply global fidelity. LIME will select the explanation with the highest local fidelity.

In LIME, G is the class of all possibly interpretable models. LIME assumes that for any $g \in G$, LIME can present a user with visual or textual components (to help explain the model-prediction combination). As not all $g \in G$ are necessarily as (easily) interpretable as simple decision trees, $\Omega(g)$ measures the complexity of the model g. The more interpretable a model is, the lower $\Omega(g)$ will be. The function Ω varies with g. One example of this is if g is a decision tree, $\Omega(g)$ may be the depth of the tree.

Let k denote a proximity measure between two instances x and z. k is the exponential kernel function, given a distance function d. $k(x, z)$ weights the sampled instance z compared to the original instance x that calculates the closeness of x and z. Ribeiro et al. uses the cosine distance function for textual data, and *L2* (least square errors) distance function for images. Their kernel function is defined as $k(x, z) = \exp(-d(x, z)^2/\sigma^2)$, with width σ.

$\mathcal{L}(f, g, k)$ is the fidelity function, and will calculate how *poor* the model (g) is at approximating the implemented model (f) defined by the proximity function (k). The sampled instances around x are defined as $\mathcal{Z}$. Given an instance x, for all closely related instances, the fidelity function will calculate the error of g that is weighted by k. The fidelity function is written as:

$$\mathcal{L}(f, g, k) = \sum_{z, z' \in \mathcal{Z}} = k(x, z) \cdot (f(z) - g(z'))^2 \tag{3.8}$$

LIME generates explanations by selecting the explanation that has a low com-

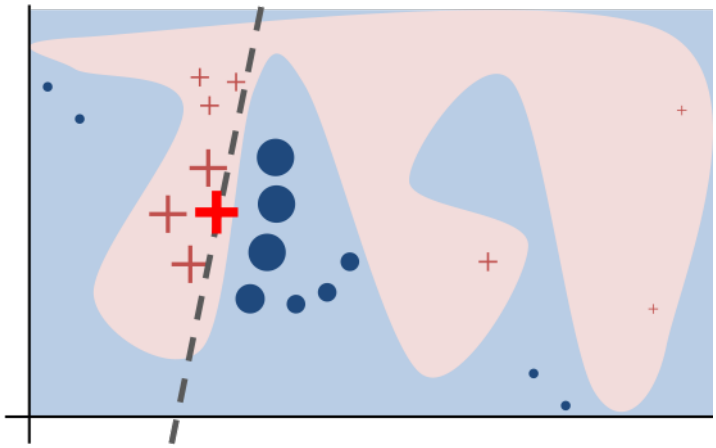

Figure 3.1: Visual example on how LIME logic works. The decision function on the model is the blue and pink background which is non-linear. The bright red cross is the instance being explained by LIME. The blue circles and other red crosses are sampled instances by LIME, and their size is their distances from the original instance. The dashed line is LIME's learned explanation [Ribeiro et al., 2016].

plexity (Ω) and low fidelity ($\mathcal{L}$). This is done by the following formula:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \ \mathcal{L}(f, g, k) + \Omega(g) \tag{3.9}$$

The approximation is generated when LIME is to explain $f(x)$. LIME will sample instances surrounding x and weigh the sampled instances up against x. LIME will then create the selected model g, which is the explanation, for instance, x.

To allow for interpretability, Ribeiro et al. uses a *bag of words* representation for text classification. They limit the number of words used by the threshold value K, a constant value. For image classification, they use *super-pixels* for interpretability, a collection of regular pixels used to segment an image into larger groups with some meaning. They use a custom algorithm called K-LASSO to select K features as the threshold value.

A visualization of LIME is shown in Figure 3.1. LIME works on tabular data, images, as well as text, making it a versatile model. On tabular data and text,

the explanations are a prediction probability of x, i.e., the probability of x belonging to a specific class, seen in Equation 3.9. Since LIME only gives a set of probabilities of what class x is a part of, it is very limited in its explanations. Despite LIME being released in 2016, it is still relevant to this day. LIME is also included in Captum; a popular PyTorch library aimed to increase model transparency and explainability.

3.1.4 DeepLift

DeepLift is a post-hoc backpropagation method explaining an existing model's output by comparing a given input to a chosen reference input. More specifically, the explanation is the difference between the input x and output $y = f(x)$ with regards to a reference input x^0 and its output $f(x^0)$, thus calculating importance scores of the features $x_i \in x$. According to Lundberg and Lee [2016], when viewing Shapley values as a conditional expectation of output given input, DeepLift values are a fast approximation to Shapley values. Thus it tries to solve the first issue of Shapley values as a method for explaining complex models. Formally, the explanations of DeepLift are written as the sum of the contributions of each features' difference to the corresponding reference feature, $\Delta x_i = x_i - x_i^0$, towards the difference in output and reference output. Each contribution is denoted $C_{\Delta x_i \Delta y}$, such that Δy is the difference $\Delta y = y - y_0$ between the output y and reference output y_0.

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta y} = \Delta y \tag{3.10}$$

This is called the summation-to-delta property. Earlier methods used either propagation or backpropagation, according to [Shrikumar et al., 2017], to create feature importance score explanations. Backpropagation methods need only one pass to create the explanation as they calculate the derivative of the loss function with respect to the weights, the gradient, to find the direction and value of which to update the weights. In comparison, propagation methods consist of input perturbation to measure features' impact on the output used to create an explanation. This is, however, computationally ineffective. Multiple backpropagation methods use the gradients of the output of the target class, with respect to the input, as the signal to propagate back through the network to see input significance.

Perturbation-based methods face the problem of saturation. Saturation is encountered when some of the inputs are used in a function that limits the values to a fixed range. When the inputs exceed this threshold, the output will not change. An example of this given by Shrikumar et al. [2017] is Equation 3.11, where $i_1 = 1$ and $i_2 = 1$, such that $y = 1$. As perturbation-based methods change

one value at a time, when either value is changed, there is no change in y, falsely giving both features an importance score of 0.

$$\begin{aligned} y &= 1 - h \\ h &= \max(0, 1 - i_1 - i_2) \end{aligned} \tag{3.11}$$

DeepLift solves this by splitting $C_{\Delta x_i \Delta t}$ into a positive and negative part, allowing the sum to be non-zero even when the change in the input with respect to the output is zero.

Some gradient methods face the problem of misleading importance scores when facing discontinuing gradients. The problem of discontinuing gradients is the biases or rectifying functions like ReLU, which can confuse gradients as the underlying function seemingly change when a value is reached, e.g., 0 for ReLU. An example of this is $y = max(0, x - 10)$. At 10 or lower, the gradient is 0, where above 10, it jumps to 1. By comparing with a reference instead, DeepLift avoids this sudden jump.

The reference input DeepLift utilizes can either be selected by trial-and-error or by utilizing domain experts. An example of such a reference input would be all zeros (all black) for a classifier that uses pixel values of black-white images. This is what [Shrikumar et al., 2017] used when testing the MNIST dataset since one can then filter out the background and only look at activations for the non-black sections. DeepLift improves on earlier methods for importance scores by utilizing the reference input to find importance scores, although some gradients are zero. However, it relies on picking good reference input in contrast to earlier methods that did not require any reference input. Another improvement is the three rules for assigning importance scores for each neuron to its output, which is then used with the chain rule to calculate the importance score for the features. These rules are the Linear rule, the Rescale rule, and the RevealCancel rule. Let x denote the previous linear layer, such that x_i are the inputs from node i in layer x to a neuron y. Let Δx^+ denote the positive parts of x and Δx^- denote the negative parts of x, so that $x = \Delta x^+ + \Delta x^-$. The linear rule applies to linear functions,

and define:

$$\begin{aligned}
\Delta y^+ &= \sum_i 1\{w_i \Delta x_i > 0\} w_i (\Delta x_i^+ + \Delta x_i^-) \\
\Delta y^- &= \sum_i 1\{w_i \Delta x_i < 0\} w_i (\Delta x_i^+ + \Delta x_i^-) \\
C_{\Delta x^+ \Delta y^+} &= 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^+ \\
C_{\Delta x^- \Delta y^+} &= 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^- \\
C_{\Delta x^+ \Delta y^-} &= 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^+ \\
C_{\Delta x^- \Delta y^-} &= 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^- \\
m_{\Delta x_i^+ \Delta y^+} = m_{\Delta x_i^- \Delta y^+} &= 1\{w_i \Delta x_i > 0\} w_i \\
m_{\Delta x_i^+ \Delta y^-} = m_{\Delta x_i^- \Delta y^-} &= 1\{w_i \Delta x_i < 0\} w_i
\end{aligned} \tag{3.12}$$

The Rescale rule applies to nonlinear transformations with a single input such as ReLU.

$$\begin{aligned}
\Delta y^+ &= \frac{\Delta y}{\Delta x} \Delta x^+ = C_{\Delta x^+ \Delta y^+} \\
\Delta y^- &= \frac{\Delta y}{\Delta x} \Delta x^- = C_{\Delta x^- \Delta y^-}
\end{aligned} \tag{3.13}$$

RevealCancel is an alternative to Rescale, which handles cases where Rescale returns misleading results. [Shrikumar et al., 2017] gives the example of $\min(i_1, i_2)$ with $i_1 = 0$ and $i_2 = 0$ where Rescale will improperly attribute all importance to either one of them. RevealCancel will attribute 0.5 to each.

$$\begin{aligned}
\Delta y^+ &= \frac{1}{2}(f(x^0 + \Delta x^+) - f(x^0)) + \frac{1}{2}(f(x^0 + \Delta x^- + \Delta x^+) - f(x^0 + \Delta x^-)) \\
\Delta y^- &= \frac{1}{2}(f(x^0 + \Delta x^-) - f(x^0)) + \frac{1}{2}(f(x^0 + \Delta x^x + \Delta x^-) - f(x^0 + \Delta x^x)) \\
m_{\Delta x^+ \Delta y^+} &= \frac{C \Delta x^+ \Delta y^+}{\Delta x^+} = \frac{\Delta y^+}{\Delta x^+}; m_{\Delta x^- \Delta y^-} = \frac{\Delta y^-}{\Delta x^-}
\end{aligned} \tag{3.14}$$

To test DeepLift against other explanation methods, [Shrikumar et al., 2017] created a task where each model should find the set of 157 pixels (approx. 20% of the image) to remove which most impacts the resulting classification to change the classification to a new target class. Each model is then compared based on the log-odds score change from the original classification to the new target classification. DeepLift showed promising results compared to previous methods when tested on this task, as shown in Figure 3.2.

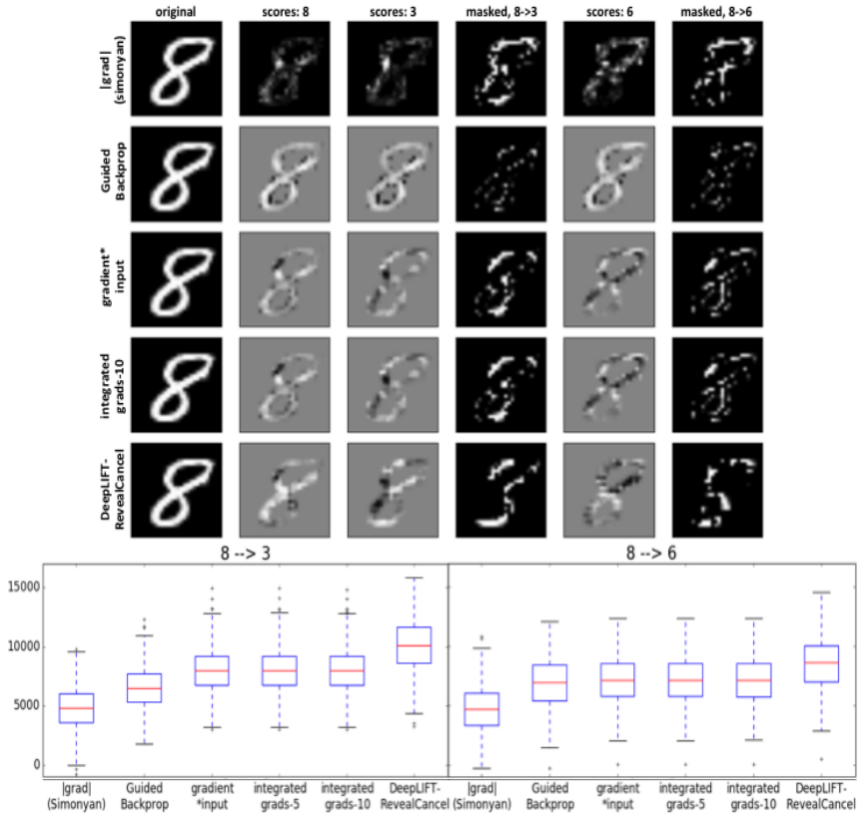

Figure 3.2: DeepLift results on the MNIST dataset compared with other methods. In the top image, one can see the white areas as the different pixel importance scores for the classification, whiter means higher. The lower part of the image is the change in log-odds score between classes for each method. Higher values in the box plots indicate better results [Shrikumar et al., 2017].

3.2 Interpretable Explanations by Counterfactuals

One of the issues that Frye et al. [2020] noted regarding Shapley values, which also apply to DeepLift, is that the explanations based on the raw input x might not be directly interpretable. To understand what the value indicates, one might need expert- or domain knowledge.

Multiple methods were developed based on the idea of contrastive explanations from Miller [2019] to address the issue of interpretable explanations. These methods are capable of generating counterfactual explanations. By the definitions of Barredo Arrieta et al. [2020], these methods are categorized as explanations by example. A counterfactual explanation is an explanation that tells what modifications need to be made to an instance's features to change its classification, e.g., an original classification of *false* to the different classification *true*. In the case of bank loans, an example of a counterfactual explanation could be that a current application gets denied a loan. However, if they reduced the loan amount by 10% or increased their yearly income by 5%, their application would be accepted. As pointed out by [Martens and Provost, 2014], a counterfactual search algorithm may have various goals. Possible goals of a search for counterfactuals may be:

Goal 1. to find the counterfactual explanation with as few feature value changes as possible,

Goal 2. to find all counterfactual explanations,

Goal 3. to find as many counterfactual explanations given a specific time limit, or

Goal 4. to find a counterfactual explanation as quickly as possible.

Choosing one (or several) of the goals is a significant efficiency-effectiveness trade-off when searching for explanations.

Ramon et al. [2020] says that the binary representation x' of x is used to search for the counterfactual explanations, as it shows which feature values are essential for its classification and which feature values can be ignored. An active (1) feature is partially responsible for its current classification, while an inactive (0) feature is not. A counterfactual explanation to $f(x)$ is e, which is a modified instance of x with regard to x'. An *ideal* counterfactual explanation is also an ideal modified instance. The ideal instance is defined by the selected goals and how the search algorithm finds counterfactuals.

3.2.1 Heuristic Best-First Search Algorithm for Finding Evidence Counterfactuals (SEDC)

The ideas behind SEDC were first proposed by Martens and Provost [2014], which was used for classifying documents but was first implemented for explanations by Ramon et al. [2020]. They implemented their SEDC algorithm to find the explanation with as few changes as possible (Goal 1, defined above) while being model-agnostic.

The method that Ramon et al. [2020] uses is to alter a subset of x's active features into inactive features until the classification has changed, where the inactive and active features are established by $\text{bin}(x)$. Let I be a set of indices that forms a subset of the active features in x. Ramon et al. [2020] defines a modified instance of x, and the explanation e, as

$$e = \forall j \in x \begin{cases} 0, j \in I \\ x_j, j \notin I \end{cases} \tag{3.15}$$

SEDC uses four conditions to look for a set of modified instances. The first one being $d(z, x)$, which measures the distance between the new instance z and the original instance x. The distance function is a cosine similarity function, after both z and x has been converted to their binary representation, as defined earlier in equation 3.1, remember $x' = \text{bin}(x)$. In short, the more features that are in x, which are also in z, will make the instances more similar.

$$d(z, x) = cosine(z', x') = \frac{z' \cdot x'}{|| z' || \cdot || x' ||} \tag{3.16}$$

The second condition is the classifier's scoring function, f. With an appropriate threshold value t, the scoring function will allow the classifier to switch a predicted score into an appropriate binary representation. For classification problems, t is roughly the same as the class imbalance in the training set. If the score is below the value of t, it makes sure the classification has flipped; if not, it will be the same label, and it will not be counterfactual.

The last two conditions will only allow active features to be transformed and ignore inactive ones to filter out the unwanted instances.

These four conditions will create a set of explanations, A.

$$A = \{z|(z = \text{argmin } d(z,x)) \wedge (f(z) < t) \wedge (\forall x_j > 0 : z_j \in \{0, x_j\}) \wedge (\forall k : x_k = 0 \implies z_k = 0)\} \tag{3.17}$$

Equation 3.17 will therefore look for a set of modified instances, A, that are

1. as close to x as possible, and
2. that the new instance's score is below the threshold t.

In the case of A containing more than one explanation that fits the conditions of equation 3.17, z^* is the explanation that decreases the most regarding the scoring function from all possible explanations in A. This equation works as an extra layer to choose the closest explanation possible.

$$z^* = \text{argmin f(z)}, z \in A \tag{3.18}$$

The explanation that SEDC returns is the minimal list of predictors that has to be set to 0 to alter the prediction and the new score of the new prediction. The minimal list may not be global, as it is restricted by the goals previously mentioned, e.g., if the goal is to find an explanation as quickly as possible (Goal 4), it will find the first explanation with the smallest change. Ramon et al. claims that SEDC is often fast and effective, compared to their other contributions, LIME-C and SHAP-C, which use the existing LIME and SHAP frameworks. LIME-C and SHAP-C incorporate the counterfactual search algorithm of SEDC. SEDC is also probably optimal for linear models, but it was shown that SEDC was good on non-linear models and often found explanations quicker than the other models.

According to Ramon et al. [2020], an ideal explanation is an explanation that flips as few active features as possible to inactive, which is also the smallest counterfactual explanation. This is the primary objective of SEDC, to find a counterfactual with as few feature flips as possible. As defined in their counterfactual Equation 3.15, SEDC will only work on data and features where feature values can be zero.

3.2.2 Counterfactual Conditional Heterogeneous Autoencoder (C-CHVAE)

Another method for creating counterfactuals is C-CHVAE. C-CHVAE is a framework by Pawelczyk et al. [2020], which has been developed to work with tabular data. C-CHVAE's main workings are based on a variational autoencoder (VAE) structure. The VAE is used as the search algorithm for counterfactuals, where it will find proximate and connected counterfactuals to the original instance that is to be explained. The counterfactuals being proximate and connected means that they are close to original high-density data from the dataset and closely relate to existing observations. This makes the counterfactuals more believable and attainable, as they are likely to occur in real-life scenarios.

VAEs have gained in popularity as generative models to create new instances based on previous training data. VAEs are complex with their theoretical basis in probabilistic models and variational inference. The two main components of a VAE are the encoder and decoder. The encoder will reduce the dimensionality of the original data by either removing features or combining a set of features into new features to reduce the number of features. The decoder will then reconstruct the instance. The encoder-decoder pair wants to keep as much information in the dimensional reduction. The encoder may be lossy to get as little reconstruction error as possible by the decoder. A VAE can be seen in Figure 3.3. VAEs introduce neural networks to optimize tuning the functions used for encoding data to a distribution of latent space and then decode the latent space into new instances. If a point between two different distributions of the latent space is decoded, it should look something like a mix of the two. For example, if the point is between images labeled as "Cat" and another as "Dog", the point when decoded, would look like a "Cat-Dog".

To generate counterfactuals, C-CHVAE uses the encoder to learn a representation of the original data to a lower-dimensional representation of the data v and add randomness δ to the data $v + \delta$ and input that into the decoder. When the δ is small, the decoder will potentially create a counterfactual. The potential counterfactual is then fed into the classifier. If the classification has been altered, it is a counterfactual. Therefore, the counterfactual should likely occur, as only minor changes have been made to the original instance.

They denote D as the dimensional feature space. They split their feature space into two: protected and free features, X_p and X_f respectively. Protected features cannot change, like assigned sex, and can therefore not be used to create a counterfactual. Therefore, a specific instance of a feature is denoted x_i^p for protected features and x_i^f for free features. In practice, protected and free features act like

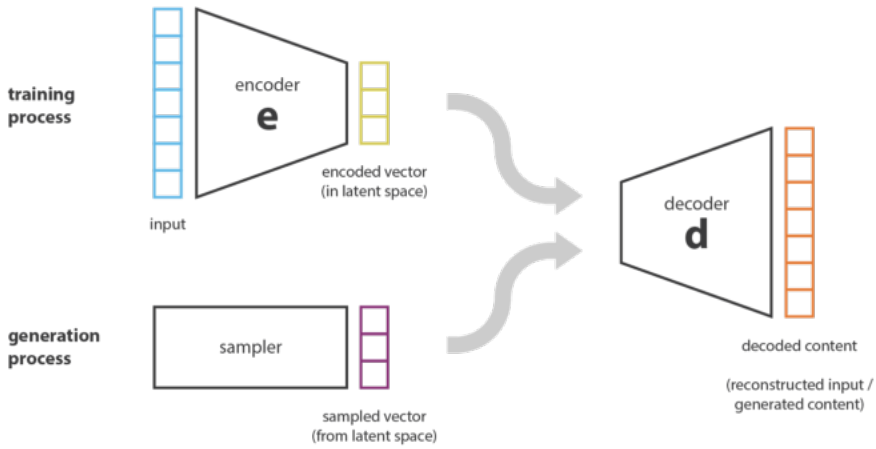

Figure 3.3: A variational autoencoder and how it samples data from the latent space [Rocca, 2021].

active and inactive features from SEDC.

C-CHVAE's VAE, like all VAEs, introduces a Gaussian prior in the encoder and decoder and tries to optimize evidence lower bound known as ELBO. The Gaussians are used to encode a distribution over the latent space, enforcing the result close to a standard distribution. There are two Gaussians for both the decoder and the encoder, $\mathcal{N}$ and Cat. $\mathcal{N}$ is the normal distribution function, and Cat is the categorical distribution. They will cluster the latent space of the data into two clusters, one for each Gaussian. The clustered data will help to look for counterfactuals among similar alternatives. Denote π as $1/L$ where L is the number of Gaussian distributions, μ to be the distributional parameters, c_i to be a vector indicating the current mixture component (a Gaussian distribution), and I to be the identity matrix.

The decoder's Gaussians are given as:

$$p(c_i|x_i^p) = Cat(\pi) \tag{3.19}$$

$$p(v_i|c_i, x_i^p) = \mathcal{N}(\mu_p(c_i), I) \tag{3.20}$$

While the encoder's Gaussians are:

$$q(c_i|x_i^p, x_i^f) = Cat(\pi(x_i^p, x_i^f)) \tag{3.21}$$

$$q(v_i|x_i^f, c_i, x_i^p) = \mathcal{N}(\mu_q(x_i^f, x_i^p, c_i), \Sigma_q(x_i^f, x_i^p, c_i)) \tag{3.22}$$

Equation 3.19 and 3.20 is the Gaussian mixture prior for the decoder, where they assume independence between c_i and x_i^p. Equation 3.21 and 3.22 is the prior for the encoder, where they assume independence between x_i^p, x_i^f as well as c_i.

C-CHVAE uses a likelihood function for each feature value, x_i, which is then summed. A log-likelihood function for each input allows for modeling data with varying values, like categorical and ordinal values, which have to be carefully selected based on the input dimensions. Log-likelihood models are used to measure the estimate of the values. After the encoder has encoded the data into its compressed representation of the latent space, the decoder will reconstruct the latent space into similar occurrences of the data. The equation 3.23 decodes the sampled input and reconstructs it into a counterfactual.

$$\begin{aligned} p(v_i, x_i^f, c_i|x_i^p) &= p(v_i, c_i|x_i^p) \prod_{d=1}^{D_f} p(x_{d,i}^f|v_i, c_i, x_i^p) \\ &= p(v_i|c_i, x_i^p) p(c_i|x_i^p) \prod_{d=1}^{D_f} p(x_{d,i}^f|v_i, c_i, x_i^p) \end{aligned} \tag{3.23}$$

The C-CHVAE encoder is much like their decoder but with different Gaussian priors, and the input is encoded as the (normal and categorical) distribution over the latent space.

$$\begin{aligned} q(v_i, x_i^f, c_i|x_i^p) &= q(v_i, c_i|x_i^p) \prod_{d=1}^{D_f} p(x_{d,i}^f|v_i, c_i, x_i^p) \\ &= q(v_i|c_i, x_i^p) q(c_i|x_i^p) \prod_{d=1}^{D_f} q(x_{d,i}^f|v_i, c_i, x_i^p) \end{aligned} \tag{3.24}$$

The counterfactual search, C, is the nearest neighbor search of the latent space.

The search algorithm requires only the already trained classifier, f, and the instance x. The search will then return the closest explanation to $C(x)$.

In contrast to SEDC, C-CHVAE creates counterfactual explanations that allow for distances in their counterfactuals, e.g., a real number like 10, representing the new value instead of only 0. The real numbered counterfactuals by C-CHVAE make it more flexible by enabling the usage of datasets where non-binary only explanations are helpful. The difference of the explanations could be "You need $10 000 more income a year to get a loan." instead of "Your income is not sufficient." as SEDC would explain.

3.3 Teaching AI to Explain its Decisions (TED)

TED is a framework created by Hind et al. [2019] at IBM. In contrast to other methods in this chapter, this framework does not build upon existing AI systems but defines a framework for building new models with incorporated explanations. As TED incorporates explanations, it is one of few embedded explanation methods as shown in the taxonomy tree in Figure 2.1. TED addresses the issue of creating interpretable explanations of complex models without creating a new representation. The framework is model agnostic to any supervised machine learning algorithm. Although Hind et al. states that the explanations have to be created by a domain expert to guarantee that they match the complexity and mental model of the domain, the framework itself is agnostic to how the explanations are generated. The explanations may be anything, ranging from numbers to strings and images to videos. This freedom of what an explanation may be allows for explanations tailored for different knowledge groups, e.g., customers vs. experts.

The TED framework is inspired by the same process a junior employee goes through when learning the ropes from a senior employee. The senior employee shows them right from wrong in certain instances, explaining why their reasoning is correct. After a while, the junior will be able to reason by themselves without the senior employee. This is the same intuition on how TED works as a supervised learning framework.

The TED framework requires three components to learn, X, L, and E. As denoted earlier in the chapter, X is the set of instances. L is the set of labels for each instance in X, which is the required part for all supervised machine learning algorithms. The final component, E, is the set of explanations for each label in L. Many XAI methods require an explanation related to the instance of the feature, but with TED, any high-level (or low-level) concept may be used. The TED

framework will predict both the label l_i and e_i on instance x_i. Hind et al. [2019] accomplishes this by encoding l_i and e_i into le_i so that any supervised machine learning algorithm can use it. To return the explanation, one would decode the classification, le_i, provided by the classifier, back into l_i and e_i.

In a typical training example, the supervised learning algorithm will need (x, l) to learn correctly. A set of training examples may be

$$\begin{aligned}&(x_1, l_1), (x_2, l_1), (x_3, l_1), (x_4, l_1)\\&(x_5, l_2), (x_6, l_2), (x_7, l_2), (x_8, l_3)\end{aligned} \tag{3.25}$$

The TED approach is to include the explanation with the classification. When adding the TED approach to 3.25, it may look like

$$\begin{aligned}&(x_1, l_1, e_1), (x_2, l_1, e_1), (x_3, l_1, e_2), (x_4, l_1, e_2)\\&(x_5, l_2, e_3), (x_6, l_2, e_3), (x_7, l_2, e_4), (x_8, l_3, e_5)\end{aligned} \tag{3.26}$$

If the training example is in the patient-diagnosis field, a certain patient (x_1) may get a treatment (l_1), due to an explanation (e_1). Another patient (x_3) may get the same treatment (l_1), but due to a different explanation (e_2).

In example 3.26 it shows that TED increases the classification space when applied to the training examples given in 3.25 from three unique classifications into five unique classifications. The increased number of unique classifications (by adding explanations) is a property that a machine learning algorithm may utilize to increase training accuracy. This can be seen when turning the original classification spread of $l_1 = 4$, $l_2 = 3$, and $l_3 = 1$ into $l_1e_1 = 2$, $l_1e_2 = 2$, $l_2e_3 = 2$, $l_2e_4 = 1$, and $l_3e_5 = 1$ with TED.

The general idea of TED is then to generalize over explanations and classification pairs to use the intrinsic knowledge in the input to get the explanation. According to McDonnell et al. [2016], this may not burden the training time and will improve accuracy. However, as TED requires explanations for every sample in the training data, it is limited to classification tasks and not regression tasks.

The basis of TED is that the system creating a prediction is also the same system that is creating an explanation. An anecdote to show the difference between embedded explanations and post-hoc explanations: person A shadows person B, for a while, trying to understand what they think and how they reason about things. Later, when person B is asked about something and answers, it is up to

person A to explain it (post-hoc). Although this might work, it is not given that they will always get it right, or if they will get it right at all. It would be more reasonable if person B answered and explained their reasoning (embedded). This conclusion is equivalent to the one by Rudin [2019].

Chapter 4

Architecture

In this chapter, we introduce our multi-task learner architecture, which builds upon the general idea of TED from Section 3.3 and is grounded by previous work on multi-task learning from Section 2.2.1, presented in Section 4.1. In Section 4.2 we argue for using counterfactuals as explanations in the architecture. In the final Section 4.3, we present the multi-task learner architecture and how we achieve trustable explanations.

4.1 Architecture Overview

Our proposed architecture, Friendly-Enigma, is a multi-task learner deep neural network with two heads, as in Figure 4.1. The first head, the primary task, is responsible for predictions. The second head is responsible for creating an explanation to explain what was predicted by the first head. The explanation can be anything, as long as it is in a format suitable for neural networks.

While TED addresses directly interpretable explanations, it requires building a model anew to learn the combined classification (x, l). Furthermore, it is limited to classification tasks. By taking the general idea of learning parameters for both explanation and prediction or classification in the same system but dividing the tasks into different heads, keeping a previous AI system is possible. Caruana [1997] also finds that multi-task learners generalize better than single-task learners on most tasks. These are the rationale behind the Friendly-Enigma architecture.

4.2 Explanations

We recommend using counterfactuals as explanations based on Miller [2019], as why-questions almost always ask for a contrastive explanation. Furthermore, a counterfactual is of the same format as the data learned by the primary task making it easy to represent as an explanation.

There are two ways to create counterfactuals. The first option is to use a domain expert to generate counterfactuals for each sample of the data as in Hind et al. [2019]. One can then be sure that the counterfactuals are as good as the expert. The other option is to generate the counterfactuals to get synthetic data. To generate counterfactuals, SEDC or C-CHVAE can be used as presented in Section 3.2. SEDC will generate counterfactuals as the minimal set of features that can be "removed" to change the prediction or classification and the new prediction score. By "removing", it is meant that the feature value is set to 0 for a given sample. C-CHVAE views counterfactual explanations as a perturbed instance of the original instance's predictors, which achieves a different classification. For binary classifications, the original target value of 1 would become 0, and conversely, 0 would become 1.

4.3 Friendly-Enigma

Creating a more trustworthy explanation alongside a prediction is the pillar of the Friendly-Enigma architecture. We build on the idea of TED that the system that predicts should also be the one to explain. This is the reason we explore an embedded explanation method by using an MTL instead of using a post-hoc method directly.

The reasoning behind choosing a multi-task learner architecture is that we believe that there is some intrinsic knowledge in the training data that should be the basis when creating a prediction and an explanation, similar to the TED framework. The hard parameter sharing strategy was chosen as it was shown by Caruana [1997] to be better than a single task learner, and as such, sufficient to reach our goal. Both the prediction head and the explanation head may benefit by sharing the knowledge. The shared values in the core body are the base knowledge for both the two heads, while the more specialized knowledge for predicting and explaining lies in the respective heads. Each head of the MTL may have its own set of layers and respective loss functions to fit and encapsulate the problem space properly.

4.3.1 Prediction and Explanation Heads

In multi-task learners, it is common to differentiate between the primary task and the secondary tasks. Our prediction head is what will be seen as the primary task in the AI system. Although we name it the prediction head, it is not limited to prediction, i.e., regression tasks.

The explanation head is the secondary task of the multi-task learner. This is equivalent to the explanation e of the composite output le of TED. Each head has its loss function, *prediction loss* and *explanation loss* respectively, and they are used to minimize errors on the prediction task and the explanation task. As this task might be more complicated than the prediction, one might wish to have some layers before the explanation output. To ensure that these layers still rely on the same information and parameters that the prediction does, we include a value based on their sign-difference, shown in Equations 4.1 to 4.4.

4.3.2 Loss Function

In multi-task learners, there is one loss that consists of multiple loss functions, one for each head. The loss of each head is weighted by the hyperparameter λ to allow for unequal importance of the prediction task and explanation task. The default value $\lambda = 0.5$ weights the prediction and explanation loss equally.

As mentioned in Section 2.2, no quantitative metric exists that can be included in a loss to optimize for explanation capability. Thus, we enforce the explanation head to utilize the output of the last shared layer similar to the prediction head. This assures that a value that counts positively towards the prediction should also count positively towards the explained value. To make sure this is upheld in our MTL a *sign-difference value* and an *alignment weight* is added to the total loss. The alignment weight is used to set the importance of the sign-difference, in contrast to *prediction weight* that sets the importance of the combined loss of the prediction and the explanation, i.e., an alignment weight of 10 and a prediction weight of 1 makes the alignment ten times more important than the prediction weight.

The calculation of the sign-difference value uses the gradients of the prediction loss and explanation loss with respect to the last shared layer. These gradients tell us how to update the weights. Thus, if the sign of a value of the gradient vector for the prediction and the explanation is the same, then the attribution to the sign-difference value will be 0. If the signs of the values are not equal, the attribution is the mean squared error of the distance to the correct sign of the gradients of the explanation head. Note that we only count the explanation

head's values as errors since its task is to explain the prediction head while the prediction head is independent.

Each part of the gradient vector of the heads is compared against each other to evaluate which ones have different signs. This is written formally as:

$$l_{a,b} = \begin{cases} 1, ((a < 0) \wedge (b > 0)) \vee ((a > 0) \wedge (b < 0)) \\ 0, else \end{cases} \tag{4.1}$$

The values of the gradients that have different signs are gathered in the set U. Formally, U is defined as:

$$U = \{b \mid l_{a,b} > 0\} \tag{4.2}$$

The sign-difference value s is the mean squared value of U, where the error is the distance to the correct sign. As such, this value indicates *how much* the network should adjust before every value has the correct sign.

$$s = \frac{1}{n} \sum_{u \in U} u^2 \tag{4.3}$$

The sign difference s is then added as part of the total loss of the model:

$$\text{Loss} = weight_{alignment} \cdot s + weight_{prediction} \cdot (\lambda \cdot loss_{prediction} + (1 - \lambda) \cdot loss_{explanation}) \tag{4.4}$$

The default values of the prediction weight and alignment weight are an equal distribution, i.e., 0.5. It is possible to change this according to which variable is most important.

4.3.3 Using the Friendly-Enigma Architecture

There are two ways to use the Friendly-Enigma architecture. Although it builds on the idea of TED, the first way it can be used is with an existing AI system. which TED cannot. This is done through "freezing" the existing layers [Sagar, 2020] and appending an explanation head and -layers. The system can then be trained on data containing both prediction targets and explanations. This method is more akin to transfer learning, although it maintains the primary task for predictions. The other method is to instantiate a new system and to train it

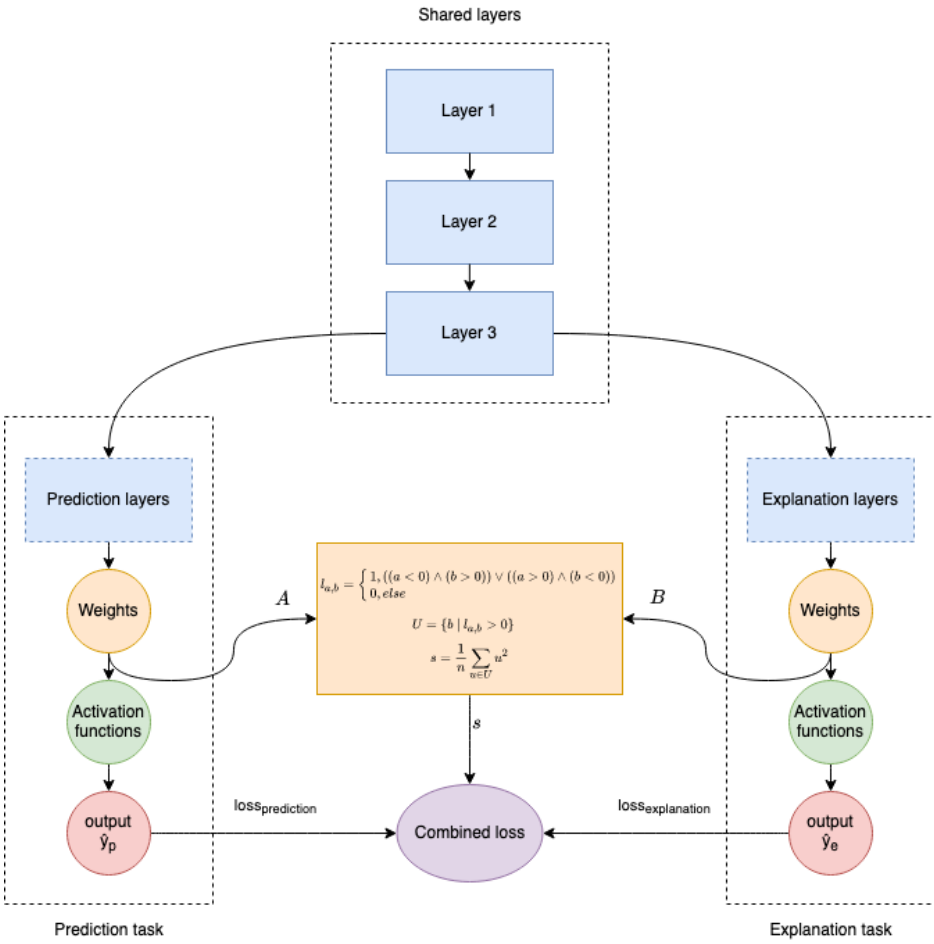

Figure 4.1: Diagram of the Friendly-Enigma architecture with sign-difference loss function

"holistically", i.e., both heads at the same time. This is the core training method of multitask learning, as explained by Caruana [1997], and is the method tested in our implementation found on GitHub [Kvamme and Larsen, 2021].

Chapter 5

Experiments and Results

In this chapter, we present experiments related to the research questions proposed in Section 1.2. We will also present the results and the discussion of those results.

We will present the datasets that have been used in the experiments and how the explanations have been created in Section 5.1. In Section 5.2, we repeat our research questions and explain our experimental plan, which is executed in Section 5.3, presenting the experiments and their results.

5.1 Datasets

We will be using three datasets

- Synthetic Data
- Synthetic Data with Manual Explanations
- Synthetic Data with Counterfactual Explanations

5.1.1 Synthetic Data

The synthetic dataset was made to test and verify the validity of the different components in the architecture. The synthetic dataset has five (5) features; four (4) predictors and one (1) target, where the target feature is a binary classification

problem. This dataset contains 50000 instances. The four features are integers of a random value between -50 and 50, where each value has an intrinsic *contribution value* between -20 and 20. If a feature is below -20, e.g., -25, its contribution value is limited to -20, and equivalently for positive values. The weighted sum of all features is between -80 and 80. A positive example in this dataset is when a row's contribution sum is above or equal to 20. All other examples are negative. This dataset has 20% positive examples and 80% negative examples.

Two different sets of explanations were made to go alongside this dataset, one created manually by us and a set of counterfactual explanations created by C-CHVAE. They are described in further detail in Section, 5.1.2 and 5.1.5.

5.1.2 Synthetic Data with Manual Explanations

As we created the dataset with a known target function, a script generating perfect explanations was made. The explanations used for this dataset are a row of 0s and 1s matching the length of the dataset's predictors. The 1 represents the feature that is the explanation of this instance. For a positive instance, the script finds the index of the feature that contributes the most to that given instance. The opposite is done for a negative instance, finding the feature that contributes the least. The explanation is the most significant index as an explanation should be selective [Miller, 2019]. If there are several equally significant indices, any one of them is chosen at random. An example for both a negative and positive instance and their explanations can be seen in Table 5.1. The manually crafted explanations have an equal majority split of 25% on the four features.

Original instance	Explanation
(target is 0) -19, 2, -25, 19	0, 0, 1, 0
(target is 1) 19, 9, 14, 48	0, 0, 0, 1
(target is 1) 35, 7, -1, 22	1, 0, 0, 0

Table 5.1: Example of an original instance from the Synthetic Data dataset and an explanation from the Synthetic Data with Manual Explanations dataset

5.1.3 Counterfactuals by C-CHVAE

To generate explanations, we use the C-CHVAE framework and implementation by Pawelczyk et al. [2020]. C-CHVAE was selected because it is model agnostic, works on tabular data, and creates counterfactuals that are connected and proximate, as presented in Section 3.2.2. Pawelczyk et al. also showed good results on real-world datasets with C-CHVAE.

The original implementation of C-CHVAE creates counterfactuals for any given dataset. The implementation was modified to create the delta between the original instance and its counterfactual as well. By the taxonomy of Section 2.1, the delta is actionable as it contains information regarding how much each value needs to be changed to get the desired classification. An example of an original training instance, the instance's counterfactual, and their delta is shown in Table 5.2.

Training instance	Counterfactual	Delta
42, 4, 10, 2	52, 4, 5, 8	10, 0, -5, 6

Table 5.2: Example of a synthetic example training instance, counterfactual from C-CHVAE, and their delta

5.1.4 Amplification Process

As Miller [2019] states, explanations are selective. Thus, a counterfactual might contain too much information for the recipient. We decided to simplify explanations and therefore simplify the counterfactual dataset. This process is named the amplification process, as it potentially increases the size of the dataset manifold. The counterfactuals were generated by overlapping the original counterfactual and each value in the delta, as shown in Table 5.2.

Each number in the delta is a feature's value to be applied to the original instance to become the counterfactual. We generate n rows of data for each counterfactual, one for each non-zero value in the delta, as shown in Table 5.3 compared to the original data from Table 5.2. If all features are in use, i.e., all counterfactuals contain non-zero feature values, the distribution of explanations are $\frac{1}{n}$

New training instance	New counterfactual	New delta
42, 4, 5, 8	52, 4, 5, 8	10, 0, 0, 0
52, 4, 10, 8	52, 4, 5, 8	0, 0, -5, 0
52, 4, 10, 2	52, 4, 5, 8	0, 0, 0, 6

Table 5.3: A set of training examples, counterfactuals, and their deltas which has been amplified from its original format.

5.1.5 Synthetic Data with Counterfactual Explanations

The Synthetic Data with Counterfactual Explanations dataset will be created, amplified, used, and inspected in Experiment 2, to see if C-CHVAE can under-

stand the intricacies of what makes a positive and negative classification in the dataset.

As mentioned in Section 3.2.2, some values can be locked if they should not be used in C-CHVAE's counterfactual generation process. The four features were set as free features, and none of them were locked. As all feature values are given random values, they should have equal importance, and it is expected that each feature will be used to create a counterfactual. If they are of equal importance, the explanations will have an equal majority class distribution of 25% as well, like the Synthetic Data with Manual Explanations dataset.

5.2 Experimental Plan

The technical research questions from Chapter 1 are repeated here for reference.

Research question 2 How does the task of explaining couple with the task of classification or prediction?

Research question 3 Can the explanations for neural networks with explanation capability be generated through existing methods?

Research question 4 How can you trust the explanation capability of the network?

Research question 5 Can the task of prediction be learned from sparser data by adding a task of explanation?

5.2.1 How Does the Task of Explaining Couple with the Task of Classification or Prediction?

The core question that has to be answered with respect to the overall goal is how the task of explaining couple with the primary task of classification or prediction. Previous research into transferring knowledge between multiple tasks has shown that when the tasks relate, even if the secondary task is artificial, the primary task will be able to generalize better [Rosenstein et al., 2005; Caruana, 1997]. Hind et al. [2019] showed that it was possible to learn a representation for the explanation and the prediction by expanding the data type to be a classification problem where the classes are the target data combined with an explanation. Thus, the outlook of whether the task couple is promising.

The first experiment we will perform, Experiment 1, is to test the architecture

on the Synthetic Data. In this experiment, two different models will be trained. The first model we will train is an STL that will try to learn to predict the target of the dataset. The task is to learn that a positive target is when the weighted sum of the features is above 20, as mentioned in Section 5.1.1.

The second model will be an MTL using our Synthetic Data with Manual Explanations dataset. This model will show to which degree MTLs can learn to both predict and explain uncomplicated data.

5.2.2 Can the Explanations for Neural Networks with Embedded Explanation Capability be Generated through Existing Methods?

As creating explanations might be time- and resource-consuming, it is beneficial to test whether it is possible to use existing methods to generate explanations, which can be used for neural network multi-task learners. In Experiment 2, the Synthetic Data dataset will be expanded with explanations generated by C-CHVAE. An MTL will be trained on this dataset.

Suppose the network is capable of generalizing beyond the majority class. In that case, it might indicate that existing methods can be used in place of human-generated explanations for neural networks with embedded explanation capability.

5.2.3 How Can You Trust the Explanation Capability of the Network?

In Section 5.2.1 it was questioned if it was feasible to build embedded explanation capabilities into neural networks; given that it is, the question then becomes if we can trust these explanations. If a domain expert builds the explanations as in Hind et al. [2019], the domain expert might also validate the explanations of the AI system by random sampling. However, there might always be some error if the network only learns from the training data. This issue is what the novel sign-difference loss from Section 4.3 addresses. Although the logic of the loss might be intuitive, it is interesting to test whether it impacts the network. In Experiment 3, a multi-task learner is trained without the sign-difference value and then compared to a multi-task learner trained with the sign-difference as part of the loss. If the models are comparable, the addition of the sign-difference loss will make the explanation more trustworthy without any loss of predictive power.

5.2.4 Can the Task of Prediction be Learned from Sparser Data by Adding a Task of Explanation?

A notable addition is whether the prediction task can be learned from sparser data through multi-task learning with explanations. Suppose the multi-task learner can generalize better, even on sparse data. In that case, it could be a justification for the extra time and resource an institution might use to add explanations to their system. It could also apply to problems in which the data is sparse, even if one is not interested in the explanation itself.

In Experiment 4 we train STLs and MTLs on sparse data. This experiment uses the Synthetic Data with Manual Explanations dataset to test if there exists a range of sparsity where the multi-task learner can perform better than a single-task learner. For all sparser or expanded datasets, the experiment also tests if the multi-task learner performs better or equally well as the single-task learner.

5.3 Experiments

All experiments use the following setup:

1. What should be tested?
2. How will the test achieve this?
3. What metrics will be used to measure accomplishment?
4. Why are these metrics used?
5. What are the expected results?
6. Present obtained results.
7. Discuss results.

All experiments will use the same parameters unless stated otherwise. These parameters are found in Table 5.4 for the STL and Table 5.5 for the MTL. As both networks should be comparable, they were initialized with equal layers. The λ weight of the MTL loss, as mentioned in Section 4.3.2, is set to the default value 0.5 so that the loss value is in the same range as the STL, to make the plots of the models' performance comparable. The activation function is chosen as LeakyReLU for all layers to mitigate the problem of dying ReLUs [Leung, 2021]. As the Synthetic Data dataset is a binary classification task, we use binary cross-

entropy loss for the prediction head, although the architecture is not limited to classification problems.

Parameter	Value
Optimizer	Adam
Learning rate	0.0004
Batch Size	1000
Epochs	200
Shared layers	(Size, activation) 256, LeakyReLU (0.01 negative slope) 256, LeakyReLU (0.01 negative slope) 256, LeakyReLU (0.01 negative slope)
Prediction layers	1, Binary Cross Entropy
Dataset	Synthetic Data

Table 5.4: Default parameters for the STL

Parameter	Value
Optimizer	Adam
Learning rate	0.0004
Batch Size	1000
Epochs	200
Shared layers	(Size, activation) 256, LeakyReLU (0.01 negative slope) 256, LeakyReLU (0.01 negative slope) 256, LeakyReLU (0.01 negative slope)
Prediction layers	1, Binary Cross Entropy
Explanation layers	4, Cross Entropy Loss with Logits
Dataset	Synthetic Data with Manual Explanations

Table 5.5: Default parameters for the MTL

5.3.1 Experiment 1

The first experiment will test the efficacy of the code and the idea behind the Friendly-Enigma architecture that a multi-task learner can have the capability to both predict and explain. This will be achieved by training two different models. First, we will use the Synthetic Data on an STL to test the viability of the dataset and how well it can learn the prediction problem of that dataset. Second, an MTL model will be trained with our manual explanations. This is to see how viable it is to both predict and explain.

The metrics used in this experiment are accuracy and Area Under the Receiver Operating Characteristic curve (AUROC). AUROC will only be used for the prediction task, while both tasks use test accuracy. Accuracy is used to see how many of the total predictions were correct in the dataset. This metric is often helpful but is inadequate when the dataset is imbalanced, e.g., 90% of the dataset favors a single target, and the algorithm only learns the majority class. As mentioned in Section 5.1.1, the majority class of the Synthetic Data is 80%, and thus any accuracy above this is considered positive. On any boolean classification task, an AUROC of 50% is the same as a random guess. An AUROC value of 50% is the minimum for this experiment. The AUROC value will be used to compare the models' prediction capabilities on the binary task. For the Synthetic Data with Manual Explanations dataset, the majority class is a 25% split between the first four features. Consequently, an accuracy above 25% is considered positive. As the synthetic task is relatively easy, we expect that both models will generalize well on the prediction problem. Based on the findings of Caruana [1997] the MTL should learn the generated explanations and prediction task simultaneously.

To initialize the first model, the STL, we use a shallow neural network as the task should be easy. The parameters that differ from default are shown in Table 5.6 and Table 5.7. The loss function used for the STL model is the loss received from the task, while the combination of binary entropy loss and categorical entropy loss is used as the loss function for the MTL model.

Parameter	**Value**
Epochs	40
Shared layers	(Size, activation) 16, LeakyReLU (0.01 negative slope) 16, LeakyReLU (0.01 negative slope) 16, LeakyReLU (0.01 negative slope)

Table 5.6: Experiment 1: Non-default parameters for the STL model

Parameter	**Value**
Epochs	40
Shared layers	(Size, activation) 128, LeakyReLU (0.01 negative slope) 128, LeakyReLU (0.01 negative slope) 128, LeakyReLU (0.01 negative slope)

Table 5.7: Experiment 1: Non-default parameters for the MTL model

Results

The results of the models can be seen in the Table 5.8 and Table 5.9 respectively. Both models performed well on the prediction task, and the MTL with manual explanations learned the explanation task well. The MTL learned to predict better than the majority class, and the AUROC value was higher than 0.5.

Metric	Result
Accuracy	0.9768
AUROC	0.9979
Precision	0.9533

Table 5.8: Experiment 1: Metric results for the STL

Metric	Result
Accuracy Prediction	0.9798
Accuracy Explanation	0.9666
AUROC Prediction	0.9983
Precision Prediction	0.9660
Precision Explanation	0.9666

Table 5.9: Experiment 1: Metric results for the MTL

The first model, the STL, quickly learned to fit the dataset with few epochs. This experiment shows that the original synthetic dataset is a simple problem for a machine learning algorithm. That the STL learned quickly was expected and set a precedent on later experiments that this dataset should not be problematic to learn. Specifically, the MTL should be able to solve the prediction problem in every experiment.

The second model, the MTL, was to learn to predict and explain our manual explanations, Synthetic Data with Manual Explanations. This model exceeded expectations and reached a 97.9% accuracy on predicting test data with an AUROC value of 99.8%. On the explanation task, the MTL reached 96.6% accuracy on the test data. The result shows that learning to predict and explain which feature contributes the most is possible.

The results show that the predictions of the MTL were marginally better than the predictions of the STL, in accord with the findings of Caruana [1997]. Whether these results hold for all datasets will be further tested in Experiment 5.

5.3.2 Experiment 2

As mentioned in Section 5.2.2, Experiment 2 will test whether existing methods can be used to generate explanations for neural network multi-task learners. C-CHVAE was selected as specified in Section 5.1.3. C-CHVAE needs a classifier to generate counterfactuals. Thus we will generate two datasets with C-CHVAE, one by using the target function of the Synthetic Data dataset as the classifier and a second by using C-CHVAE as intended with a pre-trained STL from Experiment 1 on the Synthetic Data dataset. To measure the validity of a counterfactual, they will be tested against the target function and compared to the original sample's target. An MTL will train on both datasets to learn the prediction task while also learning the task of predicting which feature is the most significant feature for the outcome of the prediction task, known as the explanation problem.

In this experiment, we will measure the accuracy of the explanation head and the AUROC of the prediction head. The model used the same parameters as the MTL from Experiment 1, specified in Table 5.7. The Synthetic Data with Counterfactual Explanations dataset will be inspected after creation to find the majority class, which will be used to evaluate the model's explanation accuracy. The AUROC will be compared against the results of Experiment 1.

Based on Pawelczyk et al. [2020], C-CHVAE should be able to generate counterfactuals with minor changes to a sample (connected, proximate explanations) as mentioned in Section 3.2.2. The explanations used in Experiment 1 represent the index of the most significant value. The explanations used in this experiment are the index of the most significant delta, where the delta represents the change needed to alter the prediction, a connected counterfactual. Although they are not the same, both sets of explanations likely use the same feature for their explanations. The explanation accuracy and prediction AUROC of the experiment should be close to the MTL from Experiment 1.

Results

C-CHVAE created 50000 counterfactuals for each classifier used, the trained STL from Experiment 1 and the target function used to generate the Synthetic Data dataset. When the counterfactuals were tested on the STL, only 17831 were legitimate counterfactuals. Testing the counterfactuals created by the target function resulted in 20375 legitimate counterfactuals. 17831 counterfactuals is a reduction of 64.33% from the original 50000. Out of these instances, 32.8% were positive instances. 20375 is a reduction of 59.25%, with a 49/51% split for positive and negative instances. As we see that not all the counterfactuals generated from C-CHVAE are actual counterfactuals, we will use *legitimate counterfactuals* for

the counterfactuals that have been verified by a classifier and use *samples* for the whole dataset produced by C-CHVAE. Both datasets have an equal 25% majority class split between each feature on the explanation task, indicating that all features are in use when generating a counterfactual, as expected from Section 5.1.5.

The results of the models for dataset generated by the target function and the STL can be seen in the Tables 5.10 and 5.11 respectively. The MTL performed well on the task of prediction, although the MTL trained on the amplified counterfactuals by C-CHVAE did not perform as well as the other models. The MTL with manual explanations learned the explanation task much better than the MTL with counterfactuals by C-CHVAE. The MTL learned to predict better than the majority class, and the AUROC value was higher than that of the average of 0.5.

Metric	**Result**
Accuracy Prediction	0.7250
Accuracy Explanation	0.5126
AUROC Prediction	0.8246
Precision Prediction	0.6846
Precision Explanation	0.5126

Table 5.10: Experiment 2: Metric result for the MTL with counterfactuals by C-CHVAE, verified by the target function

Metric	**Result**
Accuracy Prediction	0.7734
Accuracy Explanation	0.4928
AUROC Prediction	0.8228
Precision Prediction	0.7848
Precision Explanation	0.4928

Table 5.11: Experiment 2: Metric result for the MTL with counterfactuals by C-CHVAE, verified by the STL

As mentioned, only close to a third of the dataset were legitimate counterfactuals. This is contrary to what was expected. The expectation was that all samples were legitimate counterfactuals with variation in connectedness and proximity. However, as seen in the box plots in Figures 5.1 and 5.2 multiple values were severely out of range for feature X0 and X1 for the STL classifier and feature X1 for the target function. These features had a standard deviation between 92 to 98, while the original data had a standard deviation of 28. As presented

in Section 5.1.1, the instances from the original dataset contain feature values between -50 and 50, although values above 20 or below -20 hold no real value to the weighted sum. As such, features X0 and X1 were neither proximate nor connected. Since the counterfactuals are not proximate and connected, it might be that they fail to capture the minimal changes needed to change the prediction and thus deviate from the original patterns. The difference between the MTL trained on this dataset and the MTL trained in Experiment 1 could perhaps be attributed to this difference.

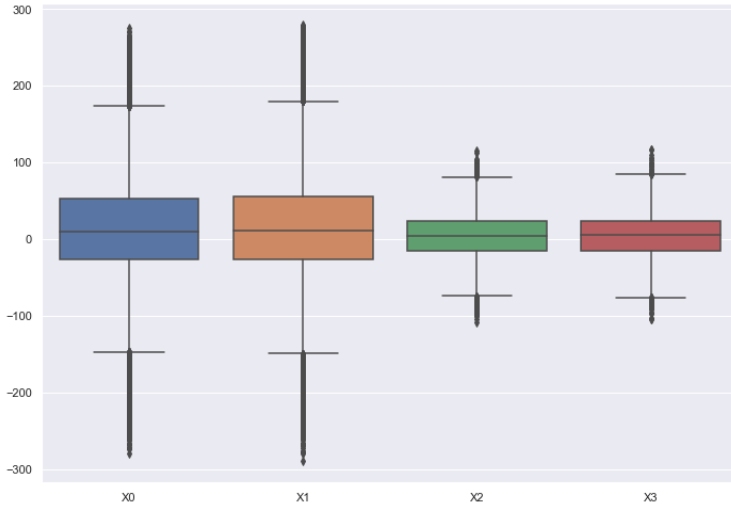

Figure 5.1: Box plot of counterfactuals generated by C-CHVAE using our trained classifier

Both models in this experiment learned to explain better than the majority class (51.26% and 49.28% respectively), and both learned to predict better than their respective prediction majority split with an AUROC above 50%. This experiment shows that both MTLs were able to learn to both predict and explain, albeit not as well as the STL or MTL from Experiment 1. We attribute the difference between the models from this experiment and the models from Experiment 1 to the counterfactuals created by C-CHVAE.

The results of this experiment indicate that it might be possible to use existing methods to generate explanations for neural network multi-task learners, at the cost of predictive power, as seen in comparison to the STL of Experiment 1. However, in comparison to Experiment 1, the explanations in this experiment were unsatisfactory. We expected better results given that the results of C-

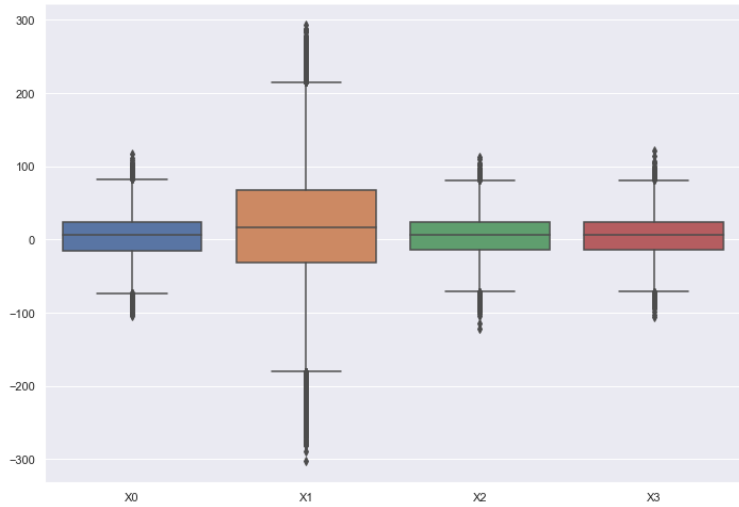

Figure 5.2: Box plot of counterfactuals generated by C-CHVAE using the target function

CHVAE presented by Pawelczyk et al. [2020] were good. As such, we will try to use the same datasets as they used, in Experiment 7.

5.3.3 Experiment 3

The Friendly-Enigma architecture incorporates a novel sign-difference loss to ensure trustable explanations. Experiment 3 will test whether the sign-difference value can be added as part of the loss to ensure trustable explanations without any negative effect on the prediction task. A model using the Friendly-Enigma architecture is initialized with the same parameters as the MTL from Experiment 1. The major difference between the new model from this experiment and the model from Experiment 1 is the sign-difference loss, as detailed in Section 4.3.2. The model from Experiment 1 will be named MTL-R (MTL-Regular), and the new model from this experiment will be named MTL-FE (MTL-Friendly-Enigma) respectively to differentiate between the two. The new model will be trained on the same dataset as MTL-R from Experiment 1.

To compare the models, we will measure the AUROC of the prediction task and accuracy of the explanation task. As both models operate on the same dataset, these metrics should be sufficient to measure their relative success.

For this experiment, as well as experiments 4, 5, and 6, we will run multiple tests to measure the statistical significance of the AUROC values. We start with the null hypothesis, H_0, that there is no difference in performance metric AUROC of the prediction head between the neural networks MTL-R and MTL-FE, i.e., the metric results are part of the same distribution. The alternative hypothesis, H_a, is that there is a difference between the two networks on the metric AUROC. These null- and alternative hypotheses are repeated for the other experiments as well. As the data does not have a normal distribution, we will perform the Wilcoxon signed-rank test. The Wilcoxon signed-rank test compares two groups of repeated measurements of paired or related data to check whether their mean ranks differ, i.e., the distribution of the samples differ [Corder, 2009]. As we use the same training and test data, the results will be related. The Wilcoxon signed-rank test gives a p-value to test whether the AUROC's of the models are significant. In other words, it tells us the probability that the difference was found through randomness and actually stems from the same group. For the p-value, we select a confidence level of 95%. A p-value of 0.05 or below will match the confidence level. Matching the confidence level means that the H_0 hypothesis will be rejected, and H_a will be used instead. Possible outcomes of the explanation task are that the MTL-FE metrics decrease due to restriction that is not free to adjust the weights in its most desirable manner for the explanation task isolated; it is restricted to the prediction. If the MTL-FE metrics decrease, it will favor the H_a hypothesis. Another outcome would be that the metrics are equivalent to the MTL-R. It could indicate that sign-difference does not negatively impact the explanation capability, given that the two tasks are coupled as tested in Experiment 1, which favors the H_0 hypothesis.

Results

The results of this experiment can be seen in Table 5.12, where the MTL-R results from Table 5.9 are repeated for clarity. As seen in the results, MTL-FE scored above the minimum expectation for each head, their respective majority classes for their task. Comparing the results of MTL-R with the results of MTL-FE shows that both models score approximately equal on all metrics.

From Table 5.12, we see no apparent difference between the two models trained on the same dataset. By the Wilcoxon signed-rank test, the p-value is 0.0488, and as such, we reject the null hypothesis. The significant difference indicates that the models are different. Although we reject the null hypothesis, the differences between the models are small; the results were good on both models. As the difference was minimal, it seems that there is no negative effect of MTL-FE's sign-difference loss, and the trustable explanations can be used with no decrease in performance. The limitations forced on the MTL-FE are that the gradients from

Metric	MTL-R	MTL-FE
Accuracy Prediction	0.9798	0.9802
Accuracy Explanation	0.9666	0.9649
AUROC Prediction	0.9983	0.9984
Precision Prediction	0.9660	0.9613
Precision Explanation	0.9666	0.9649

Table 5.12: Experiment 3: Metric results for MTL-R and MTL-FE

the last shared layer should count in the same direction (positive or negative), making sure that both tasks will use the same underlying parameters for each feature. As mentioned in Section 4.3.2, they should therefore be more trustworthy and consistent explanations. The good results might indicate that the underlying task of the dataset is too easy to learn. Experiment 6 will repeat this experiment with added noise to the dataset, obfuscating the underlying task to see if there will be a more apparent difference between the models.

5.3.4 Experiment 4

Experiment 4 will test whether a multi-task learner will perform better than a single-task learner on sparse data, as mentioned in Section 5.2.4. In Experiment 3, we used a confidence level of 5%. As the chance for making a type 1 error is the confidence level α for a single experiment, the chance of erroneously rejecting the null hypothesis for at least one out of N experiments is $\alpha_{total} = 1 - P = 1 - (1 - \alpha)^N$. Thus, the use of wrong p-values makes it likely that some tests will find significance where no significance exists, as stated by [Salzberg, 1997]. Although this experiment tests if the MTL is equal to the STL or not (a two-sided test), multiple tests on multiple models are performed to check their overall equivalency. As such, the chance of getting at least one type 1 error increases. We will thus adjust the α to the lower bound of $1 - (1 - \alpha)^N \leq 0.05$.

For all datasets, both models start on a sparse dataset of 374 samples (1% of the original data), selected randomly from the training data. 25% of the original data will be selected for testing purposes. The dataset will be increased to 5% and then increase by 5% until 95% of the training data is tested. 100% of the dataset will not be used as it would be equivalent to the results of Experiment 1 where we test the whole dataset. For every sparse dataset, ten models of each learner type will be instantiated to average the result. The AUROC metric is used as it is viable when comparing different models [Nathan, 2020]. Caruana [1997] states that multi-task learners can generalize better than single-task learners. Thus, we believe that the multi-task learner will perform better for all the datasets of any

size than the single-task learner.

Results

The results of the MTLs and STLs can be seen in Table 5.14 and Table 5.13, respectively. We stopped the experiment early as we encountered diminishing returns on all metrics.

Metric	374 samples (1%)	1876 samples (5%)	3749 samples (10%)	5624 samples (15%)
Accuracy Prediction	0.9487	0.9742	0.9772	0.9788
Precision Prediction	0.9310	0.9639	0.9600	0.9620
AUROC Prediction	0.9891	0.9967	0.9973	0.9980

Table 5.13: Experiment 4: Metric results for the STL on the Synthetic Data dataset, averaged for each percentage

Metric	374 samples (1%)	1876 samples (5%)	3749 samples (10%)	5624 samples (15%)
Accuracy Prediction	0.9268	0.9611	0.9775	0.9850
Precision Prediction	0.8860	0.9362	0.9611	0.9845
AUROC Prediction	0.9786	0.9940	0.9976	0.9991

Table 5.14: Experiment 4: Metric results for the MTL on the Synthetic Data dataset, averaged for each percentage

Data sample sizes	**1%**	**5%**	**10%**	**15%**
P-values	0.0019	0.0019	0.0019	0.0019

Table 5.15: Experiment 4: P-values for the AUROC metric by the Wilcoxon signed-rank test

The results in Tables 5.14 and 5.13 does not seem to indicate anything in particular. On 1% and 5% of the dataset, the single-task learner outperformed the multi-task learner. That the STL outperformed the MTL is contrary to the expected improvement on sparse data. However, all results from the multi-task learner on any larger dataset than 5% seem to generalize better than the single-task learner, which is in accord with Caruana's statements on multi-task learners performance. It is worth noting that both models' performance is excellent, likely because the task to be learned is easy. We adjust the confidence level to the lower bound $1 - (1 - \alpha)^4 \leq 0.05$ giving us $\alpha = 0.012$. By the p-values of the Wilcoxon ranked-sign test, listed in Table 5.15, we reject the null hypotheses for all tests. Although the AUROC difference was significant for all dataset sizes, the only percentage point difference with interest was seen on the 1% dataset. One possible reason for this difference could be that the MTL has to fit two tasks, which could be considered more challenging. The combined loss of the MTL is the

average of the two losses, as mentioned in Section 5.3. As the maximum value of the categorical cross-entropy loss is based on the number of classes that can be classified, the loss might weigh the explanation task favorably and upstage the prediction task. For the other dataset sizes, the difference was 0.27, 0.03, and 0.11 percentage points, respectively. This is an insignificant difference for the classification task of the Synthetic Data dataset, as all these models achieved an AUROC of 99%. Thus, it is unclear whether the results are different enough to say anything about the capability differences on varying dataset sizes, even if they are statistically significant.

A possible reason for the contradiction on the smaller datasets could be that the task is too easy to learn without overfitting, as even 1% of the data seems to be enough to learn how to generalize. This is tested in Experiment 5. In theory, the MTL has more data to learn from, given that the explanations are based on the same data. As such, it was expected that the MTL would perform better than the STL on sparse data.

5.3.5 Experiment 5

The results of Experiment 4 were insufficient to indicate anything related to the hypotheses regarding the MTL's performance on sparse data compared to the STL. Thus, it is interesting to see if the results were caused by the dataset perfectly representing the underlying function. To test whether the underlying function of the Synthetic Data dataset is easy to learn without overfitting, subsets of the data were made. 10%, 20%, 30%, 40%, and 50% of the 1%, 5%, and 10% datasets were converted to noise by setting the target to the opposite of the original. Both models were then tested on these new datasets. The metrics and parameters used for this experiment are the same as in Experiment 4 as the experiments are closely related. In general, when a dataset has noise, a model might overfit when trying to learn the noisy representation. Thus, we expected the single-task learner to overfit the training data and perform worse on the test data, while we expect the MTL to not overfit, based on the background theory on Section 2.2.1. A contradictory hypothesis is that the multi-task learner will overfit too. The explanations will also be noise for noisy samples, reducing any benefit the MTL has from the extended data signals.

Results

On sparser data with noise, both models perform well as seen in Figures 5.4 and 5.5. For most noise, the MTL scores higher than the STL.

Both models can generalize well even when the data contains increasing amounts

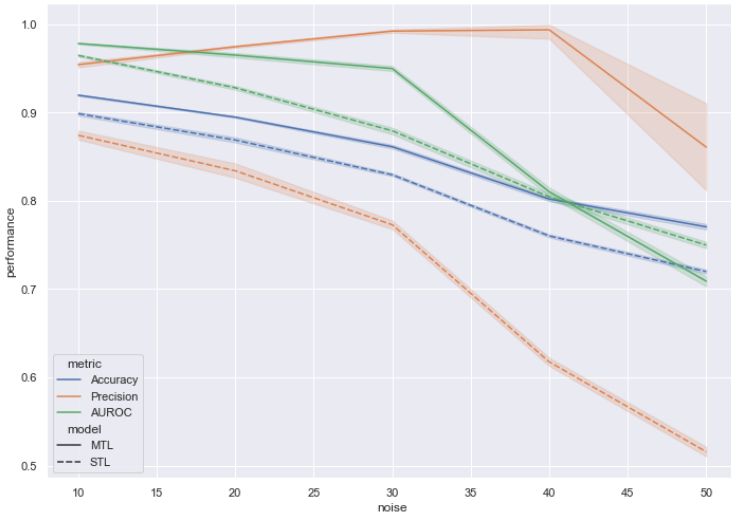

Figure 5.3: Performance measurements of MTLs and STLs on different noise levels in 1% of the Synthetic Data dataset with uncertainty regions representing the 95% confidence interval

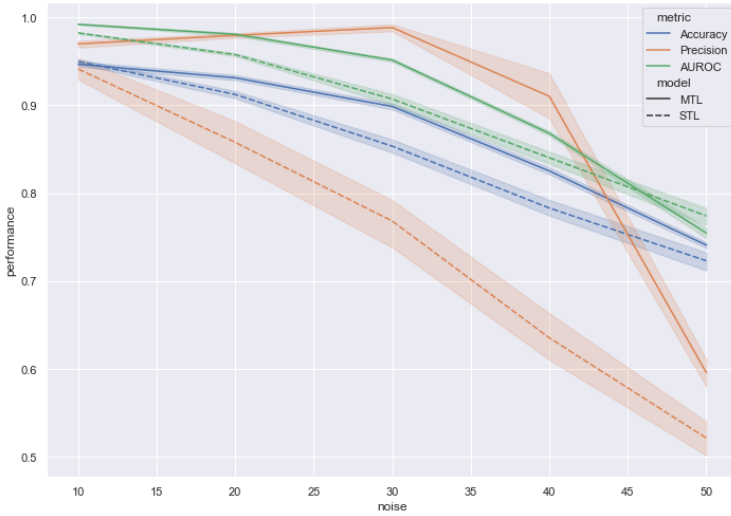

Figure 5.4: Performance measurements of MTLs and STLs on different noise levels in 5% of the Synthetic Data dataset with uncertainty regions representing the 95% confidence interval

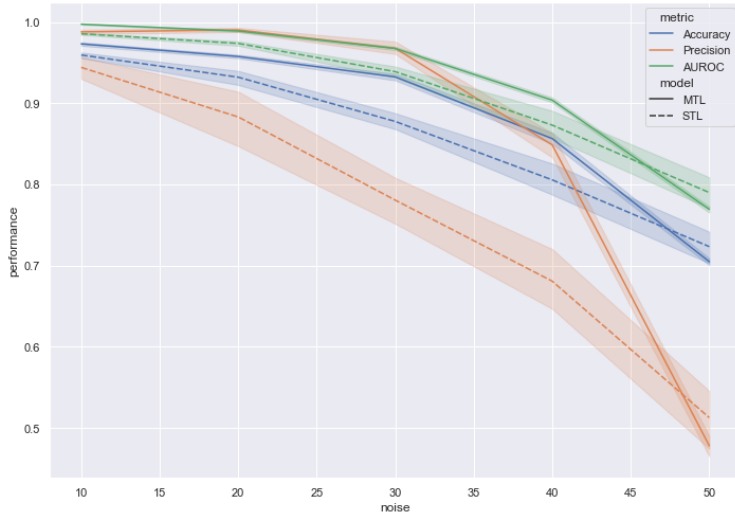

Figure 5.5: Performance measurements of MTLs and STLs on different noise levels in 10% of the Synthetic Data dataset with uncertainty regions representing the 95% confidence interval

Percentage noise	**10%**	**20%**	**30%**	**40%**	**50%**
P-values 1%	0.0019	0.0019	0.0019	0.1308	0.0019
P-values 5%	0.0019	0.0019	0.0019	0.0019	0.0097
P-values 10%	0.0019	0.0019	0.0019	0.0273	0.0839

Table 5.16: Experiment 5: P-values for the AUROC metric by the Wilcoxon signed-rank test

of noise. Despite the noise, the MTL scores better than the STL on all metrics. To account for the multiplicity effect, we set the confidence interval to the lower bound of $1 - (1 - \alpha)^{15} \leq 0.05$, giving us $\alpha = 0.0034$, as we have a total of 15 tests.

By the Wilcoxon signed-rank test, we find the p-values given in Table 5.16 for each experiment. We fail to reject the null hypothesis for the 5% and 10% datasets with 50% noise and the 1% and 10% dataset with 40% noise. The MTL scores better than the STL, indicating that the first hypothesis of Experiment 4 that the MTL will outperform the STL on sparse data might hold. It is interesting to note that for all datasets with 50% noise, the MTL performs slightly worse than the STL. However, given the statistical insignificance of the 5% and 10% datasets with 50% noise, it might be attributed to randomness, and the models might still be equivalent, as stated by the null hypothesis. The other possible reason, as the difference between the models was statistically significant on the 1% datasets with 50% noise, is that the explanation task upstages the prediction capability. This could be due to the hyperparameter λ, weighting the two losses equally. It is possible that the MTL could perform better by tuning λ. As such, the results for the MTL are conservative. If the findings would later prove to be significant given more testing, a hypothesis is that the MTL's worse performance could be attributed to the loss function. Furthermore, when 50% or more of the dataset is noise, the MTL's benefit from the extra signals might be equalized by the extra noise, and as such, the MTL and STL might perform equally.

5.3.6 Experiment 6

The results of Experiment 3 did not indicate any definitive results whether there is any downside by adding the sign-difference loss. As indicated by Experiment 5, adding noise to the dataset made the task harder to learn. This can be seen in the differences between the STL and MTL as shown in Figures 5.3, 5.4, and 5.5. As a result, we want to test adding noise to the entire dataset, to further test the hypotheses of Experiment 3. This experiment will be executed much like Experiment 5, but testing the MTL-R and MTL-FE instead. We increase the noise in intervals of 10 percentage points. The metrics used for this experiment are the same as in Experiment 3 as the experiments are comparable. As mentioned, the hypotheses from Experiment 3 are the same for this experiment as well.

Results

We stopped the experiment at 40%, as it showed close to a negligible difference between MTL-R and MTL-FE on any percent of noisy data, for all metrics as seen in Figure 5.6.

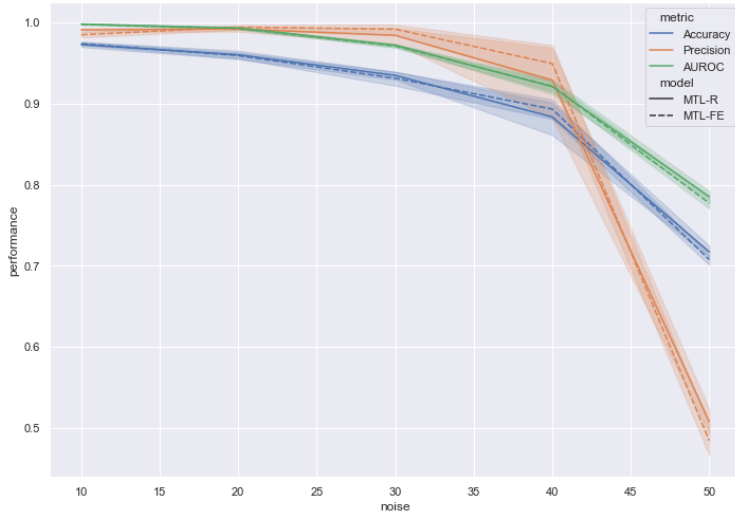

Figure 5.6: Performance measurements of MTL-R and MTL-FE on different noise levels of the Synthetic Data dataset with uncertainty regions representing the 95% confidence interval

Values	**10%**	**20%**	**30%**	**40%**	**50%**
P-values	0.0195	0.4921	0.4921	0.8457	0.5566

Table 5.17: Experiment 6: P-values for the AUROC metric by the Wilcoxon signed-rank test

The results of Experiment 6 reflect the results of Experiment 3 and the findings regarding generalization of Experiment 5. We adjust the confidence level to the lower bound of $1-(1-\alpha)^N \leq 0.05$ giving us $\alpha = 0.0102$. For all sets of different noise, we fail to reject the null hypothesis. Thus, the null hypothesis stating that the models are equivalent is still yet to be disproved. As mentioned, this indicates that the sign-difference loss can be used to enforce trustworthy explanations in an MTL based on the reasoning of Section 5.2.3 without any negative impact on the prediction performance. This validates using the Friendly-Enigma architecture to add consistent explanations without loss in the AUROC performance.

5.3.7 Experiment 7

The results of Experiment 2 did not meet the expectations and were quite contradictory to the findings of Pawelczyk et al. [2020]. As such, it is interesting to test C-CHVAE on the same datasets used in their experiments, "Give Me Some Credit" (GMSC) [Kaggle, 2011] and "Home-Equity Line Of Credit" (HELOC) [FICO, 2018]. As mentioned in Experiment 2, C-CHVAE needs a model to use for the counterfactual search. Their results were found with an SVM with a linear kernel or a *L2* logistic regression model, but they state that their system is model agnostic. As such, we will train two neural networks (STLs), one for each dataset. These classifiers will be measured against their respective competitions' best result, 73% accuracy for HELOC and 86.95% AUROC for GMSC. The C-CHVAE framework will be initialized with the same parameters from Pawelczyk et al. [2020].

A second set of two STLs will be trained on the generated samples by C-CHVAE and be compared to the original STLs. The samples will be amplified by the amplification method of Section 5.1.4. Two MTL's will then be trained on 75% of these amplified datasets. 15% of the dataset will be used for testing, and 10% will be used for validation. Last, their predictive power will be tested against the original datasets to see whether the counterfactual information is transferable to the original data.

Given that the deciding factor of the datasets is harder to pinpoint in comparison to our Synthetic Data, we will compare the difference of the mean of values of the samples to the mean of the values of the original instances to measure proximity. A low difference between means indicates that only minor changes are needed for each value to become a counterfactual. The STLs trained on the samples will be measured against all metrics for the STLs on the original datasets. Each MTL will be measured in terms of AUROC on the prediction task and accuracy for the explanation task. Furthermore, the MTL used for the HELOC dataset will also be measured by accuracy on the prediction task to compare against the

benchmark results.

Based on Pawelczyk et al. [2020], we would expect the samples to be proximate and representative counterfactuals of the original datasets. Given that they are, we expect the STLs trained on the samples to perform comparably to the STLs trained on the original datasets. Furthermore, we expect the results of previous experiments, Experiment 1 and Experiment 5, that the MTLs outperformed the STLs, to be repeated for these datasets as well. A counterfactual should be a legitimate instance, just *opposite* of the training data, and as such, it should be representative of the original data. However, by the results of Experiment 2, where C-CHVAE explanations did not prove sufficient as explanations, the more likely results are that the samples are not proximate or otherwise counterfactuals representative of the original data. If so, there is a possibility that they deviate more from the original intrinsic patterns of the original datasets.

Results

Metric	GMSC Original	HELOC Original	GMSC C-CHVAE	HELOC C-CHVAE
Accuracy Prediction	0.9381	0.7020	0.9317	0.5709
AUROC Prediction	0.8217	0.7798	0.4785	0.6834
Precision Prediction	0.5809	0.6881	0.9346	0.5464

Table 5.18: Experiment 7: Metric results for the STL on the samples made by C-CHVAE on the GMSC and HELOC datasets

The STL for the GMSC dataset achieved an AUROC of 0.82, seen in Table 5.18. This is close to the accuracy benchmark of 0.86 achieved by the GMSC competition winners.

The original distribution of the GMSC dataset can be seen in Appendix A, Table 1. The distribution of data for the C-CHVAE-generated GMSC samples are listed in Appendix A, Table 2. The means for most values of the GMSC samples reflect the original values, where *DebtRatio* and *MonthlyIncome* were the features with the largest value ranges and, as such, the largest absolute mean, which is similar to the original feature values. One note is that *RevolingUtilizationOfUnsecuredLines* has a large mean and standard deviation difference between the original and C-CHVAE datasets. It can also be noted that *RevolingUtilizationOfUnsecuredLines* and *DebtRatio* gets a minimum of -1 in the C-CHVAE dataset, which is nonexistent in the original dataset. Despite this, the samples that were generated for the GMSC dataset were proximate.

The performance of the STL on the GMSC C-CHVAE samples was not very impressive compared to the original values, as seen in Table 5.18. The AUROC value of less than 0.5 indicated that it was equal to random guess or guessing

the majority class. Based on these results, the samples were verified against the original STL, which C-CHVAE used for the counterfactual search. When verifying the samples of GMSC with the original STL, only 6% of the samples were legitimate counterfactuals. As seen in Appendix A, Table 3, almost all legitimate counterfactuals were generated for the positive target value *SeriousDlqin2yrs*. The explanations were inspected, and it showed that the majority class was 14%, split between 4 features. When the MTL was tested on the reserved test data (15% of the legitimate amplified counterfactual dataset), it scored high (above 95%) on all prediction metrics, which can be seen in Table 5.19. For the explanation accuracy, it showed that it performed better than the majority class. An explanation accuracy better than the majority class is similar to that of Experiment 2. The achieved accuracy is low for high-risk decision domains where making mistakes are fatal. This can likely be attributed to the fact that 99.5% of the targets were positive labels. However, when the model trained on the GMSC C-CHVAE samples tested on the original data, the MTL's AUROC showed that it performed slightly better than guessing randomly.

Metric	**Legitimate Amplified GMSC testset**	**Original GMSC testset**
Accuracy Prediction	0.9931	0.0656
Accuracy Explanation	0.2724	N/A
AUROC Prediction	0.9809	0.5435
Precision Prediction	0.9931	0.0656

Table 5.19: Experiment 7: Metric results for the MTL on the legitimate amplified GMSC dataset

The STL for the HELOC dataset achieved an AUROC of 0.77 and an accuracy of 0.70, seen in Table 5.18. This is close to the accuracy benchmark of 0.73 achieved by the HELOC competition winners.

The original distribution of the HELOC dataset can be seen in Appendix A, Table 4. The distribution of data for the C-CHVAE-generated HELOC samples are listed in Appendix A, Table 5. The difference of the means for most values of the HELOC samples was low, indicating that they were, in fact, proximate. The only notable difference between the original dataset and the dataset by C-CHVAE is that C-CHVAE's dataset contains several negative minimum values in contrast to the original.

Similar to the STL tested on the C-CHVAE-generated GMSC, the STL trained on the C-CHVAE-generated HELOC samples performed worse than the original, albeit better than guessing randomly. When verifying the samples of HELOC with the original STL, 40% of the samples were legitimate counterfactuals. 90% of the target feature *RiskPerformance* in the legitimate HELOC counterfactuals

were the positive value *Good*. This skew is similar to the one of GMSC (99.5%), though not as drastic. When tested on the reserved test data (15% of the legitimate amplified counterfactual dataset), it performed worse than the STL trained on the original HELOC data. The AUROC indicates that it is no better than guessing randomly. The test results on the original HELOC data are again similar to the MTL on GMSC, albeit a bit better with an AUROC of 0.63, as seen in Table 5.20.

Metric	Legitimate Amplified HELOC testset	Original HELOC testset
Accuracy Prediction	0.8664	0.5101
Accuracy Explanation	0.1152	N/A
AUROC Prediction	0.5364	0.6323
Precision Prediction	0.8949	0.4985

Table 5.20: Experiment 7: Metric results for the MTL on the legitimate amplified HELOC dataset

The expectation that the samples were proximate was met for both datasets, as seen in the results. However, they were not representative of the original dataset. For the GMSC samples, the STL was unable to learn anything. This is also seen when the MTL trained on the legitimate counterfactuals are tested on the original dataset. Given that the C-CHVAE was run with the same parameters of Pawelczyk et al. [2020], this was surprising. On the other hand, it reflects the results of Experiment 2, where the C-CHVAE-generated samples for the Synthetic Data dataset were many were invalid, and the MTL performed worse than the MTL and STL of Experiment 1.

The legitimate counterfactuals for the HELOC dataset had a greater spread of target values. As such, they were more representative of the original data than the legitimate GMSC counterfactuals. The slightly better AUROC could perhaps be attributed to this distribution. However, the results were not good compared to the STLs trained on their original datasets.

In conclusion of this experiment, the parameters used by Pawelczyk et al. [2020] indeed ensured that the generated samples were proximate. However, the distribution of target values was not representative of the original datasets. Many samples generated by the implementation of C-CHVAE by Pawelczyk et al. [2020] were not legitimate counterfactuals even though this is the main task of C-CHVAE. As such, neither the STLs nor the MTLs could use the C-CHVAE generated samples instead of the original dataset. This verifies that the results of Experiment 2 are attributed to the C-CHVAE implementation. It also shows that without good explanations in the training data, the MTL will be able to learn neither the predictive task nor the explanation task, while Experiment 1 shows

that it is possible to train an MTL with expert-generated explanations.

Chapter 6

Evaluation and Contribution

In this chapter, we will evaluate the results presented in Chapter 5 in light of the research questions.

In Section 6.1 we will present our discussion of the experiments. In Section 6.2 we will present our contributions to the field. Section 6.3 will present what we believe will be valuable to directions for future research. Finally, in Section 6.4 will be our remarks on larger topics in XAI.

6.1 Discussion

Most of the experiments showed positive results, although the Synthetic Data dataset seemed to be a bit too easy, it was thus hard to conclude anything. In the later experiments, Experiment 5 and Experiment 6, we added noise to the dataset to simulate more realistic data. These experiments showed more evident results than on the original dataset, indicating better results for neural network multi-task learners with explanation heads on more realistic scenarios. Furthermore, with the added noise, the multi-task learner outperformed the single-task learner, in accord with the research of Caruana [1997]. However, this advantage recedes at 50%, which is intuitive as the multi-task learners' benefit of extra signals from each task is diminished by the equal amount of extra noise.

We also wanted to apply the Friendly-Enigma architecture to real-world datasets. One dataset of interest was the one used by Hind et al. [2019] where they had explanations generated by domain experts. However, we found no open datasets with explanations. Thus arose the need for an explanation engine, a method for generating explanations for existing data. Through the research of state-of-the-art explanation methods, we found multiple alternatives. Based on the requirements for explanations by Miller as well as promising results on open datasets, we selected C-CHVAE as our explanation engine. Although we managed to reproduce their proximate counterfactuals, the method was unable to generate legitimate counterfactuals fully representing the original data, as seen in Experiment 7. This validated the findings of Experiment 2, where we tried to use C-CHVAE on our Synthetic Data dataset. As such, C-CHVAE was not applicable as an explanation method for existing datasets. Future work should look into other methods, as listed in Section 6.3.

When testing the multi-task learner against the single-task learner on a lot of data, no real benefit appeared, as seen in Experiment 1. The same result occurred in Experiment 4 when the training data was sparse. The real benefit of the multi-task learner over the single-task learner appeared when the data was sparse and noisy, which real-world datasets are likely to be, proved by Experiment 5. When using the knowledge of the domain experts to explain sparse data, a multi-task learner can shine as an explanation method. The multi-task learner will also do well on explanations generated by an external method, although the requirement for a better explanation engine is called for.

The Friendly-Enigma architecture includes a sign-difference value as part of the loss, as seen in Figure 4.1. The purpose of this value is to ensure that the explanations are consistent with the predictions, i.e., it is not allowed to use the information for explanations in a different manner than for the prediction. This consistency makes the explanations trustable. When testing whether this had any negative impact on the multi-task learner's prediction skill, we found no impact in Experiment 3. However, due to the hypothesis in the discussion of the results of Experiment 1 regarding the Synthetic Data dataset being too easy, we tested the architecture on noisy data in Experiment 5. The test on noisy data of Experiment 5 proved that the noise made the task harder to learn and showed different results than Experiment 4 and Experiment 1. Thus, we repeated Experiment 4 with noisy data in Experiment 6 to see whether the noise yielded different results than Experiment 4. Even with noisy data, we failed to reject the hypothesis that there was no difference between the models for most amounts of noise. However, the metric differences were negligible as they were all less than 0.5 percentage points. As the performance of both MTLs with and

without the sign-difference loss was good, the sign-difference loss can be used to ensure trustable explanations. Since both models performed well, it could be interesting to check whether an MTL will always use the shared information in a similar manner even without being forced to do so.

For Experiment 4 through Experiment 7, we initialized all STLs and MTLs with a large net. Both models were treated equally to compare their performance easily. However, it could be that the STL would perform better on a smaller net than the performance we found. As both networks trained for 200 epochs with these large layers on the relatively easy Synthetic Data dataset, it is not unlikely that the STL overfits. Thus, it could be discussed whether early stopping should have been implemented for both models to compare the models at their best. If the STL overfits while the MTL either continues to improve or is prevented from overfitting from the increased signal, the MTL will have an advantage on a higher number of epochs which might not be prominent on fewer epochs. On the other hand, better generalization, which implies less overfitting, is one of the positive advantages of an MTL argued for by Caruana [1997]. Furthermore, Salzberg [1997] states that a hyperparameter search often optimizes for a narrow task on a specific dataset and should not impede decent results. As such, our findings would likely stand even with early stopping for both models.

In our experiments, we used the Wilcoxon signed-rank test to test the statistical significance of the AUROC values. Another statistical method to compare models that can be used is McNemar's test [Salzberg, 1997]. This method is suitable for comparing two model instances. We want to compare the underlying architecture and not the trained model. As such, we need to repeat the experiments and remove the chance that specific parameters or randomness could impact the results. As the models used the same training and test data, we used the Wilcoxon signed-rank test instead. To summarize, measuring statistical significance in machine learning and artificial intelligence is complicated [Salzberg, 1997], and we will therefore advise using caution if drawing any grand conclusion from our statistical tests.

6.2 Contributions

The main contribution of this thesis is the research into using multi-task learners as a singular system to predict and explain in the field of explainable AI, where the norm is to use a second post-hoc method to explain. The research questions were formed to achieve the research goal: "Create adoptable and inherently trustable explanations as a native part of a neural network". Each contribution is listed as a subsection to present the contributions clearly.

6.2.1 Research Questions

The following research questions have been answered

1. What is the Current State-of-the-art of Explainable AI?

The current state-of-the-art is presented in Chapter 3.

2. How Does the Task of Explaining Couple with the Task of Classification or Prediction?

As Experiment 1 shows, a multi-task learner can learn to predict and explain to a very high degree. This entails that the two tasks couple, as stated by Caruana [1997]. When domain experts create the explanations, the experiment shows that the multi-task learner's explanation capability is comparable to its prediction capability. Multi-task learners perform better when they have similar tasks, indicating that a prediction and its counterfactual have similar intrinsic knowledge. This was unclear in Experiment 1 but is heavily indicative in Experiment 5. In Experiment 5, we added a large amount of noise, which showed that the multi-task learner outperformed the single-task learner. As such, creating explanations as a native part of a neural network looks feasible through our method.

3. Can the Explanations for Neural Networks with Embedded Explanation Capability be Generated through Existing Methods?

Many post-hoc methods exist to create explanations, as explored in Chapter 3, and some of these create counterfactual explanations. C-CHVAE was the selected framework as it is the only known framework that can create counterfactuals for continuous values instead of binary values where SEDC is a possible framework. If explanations are to be commonplace in neural networks, they have to be accessible. Using domain experts is not an option for multiple projects, and so this research question builds upon normalizing explanations in neural networks.

Experiment 2 shows that C-CHVAE can replace a domain expert to some degree. Approximately one-third of the counterfactuals were legitimate, and the rest had to be discarded. When the legitimate counterfactuals were used, the multi-task learners trained could explain correctly 50% of the time, where the majority class was 25%, a significant increase of a random guess. Whether this is acceptable or not is up for discussion for the domain but is not applicable to domains making high-risk decisions, e.g., criminal justice.

4. How can You Trust the Explanation Capability of the Network?

The sign-difference loss function was developed to assure the reliability of a multi-task learner's prediction and explanation capabilities. Experiment 3 and Experiment 6 tested the sign-difference, with and without noisy data respectively. Both experiments showed negligible differences between the two models trained in each experiment. The low differences support the hypothesis that both tasks are related and that the sign difference does not negatively impact the explanation capability. As such, it can be added without any loss of predictive power.

5. Can the Task of Prediction be Learned from Sparser Data by Adding a Task of Explanation?

Acquiring data for some domains can be difficult, which makes sparse data common in the real world. Experiment 5 showed indications that good explanations can help generalize even on noisy, sparse data. With noisy data, the benefits of using MTLs can be several percentage points increase in the accuracy metric. This is in accord with earlier findings by Caruana [1997] and Rosenstein et al. [2005]. If the explanations are easily accessible, as discussed in the contributions related to research question 1, the second head appended to the STL could be explanations. As such, explanations become increasingly adoptable through excellence in sparse data.

6.2.2 A State-of-the-art Taxonomy

In Section 2.1, we presented our taxonomy which extends previous taxonomies. Many existing taxonomies are focused on the methods they examine, while we present a new taxonomy with more focus on incorporating the definitions from social sciences as presented by Miller [2019].

6.2.3 Multi-task Learning for Trustable Explanations in AI Systems

As presented through the research questions of Section 6.2.1, one of our novel contributions is multi-task learning as a tool for trustable explanations in deep neural networks. By researching this combination, one might utilize the generalization skill of multi-task learners while also adding explanations to existing neural networks. Our architecture builds upon the ideas of TED [Hind et al., 2019], to use the network to explain alongside predicting, to avoid using any external method to approximate the neural network or try to understand what the output means after it has been created. However, there is no guarantee that having predictions and explanations in the same system makes the system more

trustable. Our architecture achieves this by the sign-difference loss. Through our novel sign-difference loss, we provide consistency of explanations by ensuring that the network is trained to utilize the shared layers in a similar manner. The benefit of using this method is that it is possible to freeze an existing neural network model before appending an explanation head. This makes it possible to keep the work that has already been achieved while adding explanation capabilities. As such, our work improves the ideas of TED to create trustable explanations.

6.3 Future Work

Multiple interesting topics, ideas, and issues still stand after the work presented in this thesis. These are presented here for future work.

6.3.1 Embedded Explanations without Post-hoc Methods

In our thesis, we first generate explanations through the existing method C-CHVAE. Along with Shapley and SEDC, this method uses the original data and original input as a baseline for the explanations. The alternative to generating explanations is to use domain experts. An issue that occurs when using domain experts to create explanations is that they are limited to their knowledge and how they acquired that knowledge. As mentioned in Section 2.2, the AI system AlphaGo made its famous move, move 37, which almost no professional would make. It is safe to say that when trying to explain a move like this in a dataset, the domain expert would likely give the wrong explanation. Using domain experts to create explanations limits the benefit that is gained by deep neural networks with their ability to learn new and interesting patterns where humans see none. It is therefore crucial that further work on machine learning algorithms to create explanations is done. If not, the explanations will never be any better than the explanations created by people. It is interesting to see whether one could learn these directly from the data instead of through the existing methods or domain experts.

6.3.2 Soft Parameter Sharing

As shown in Section 2.2.1, Ruder et al. [2019] states that through soft parameter sharing and using methods for learning which information to share, i.e., what to learn, multi-task learners with soft parameter sharing can perform better than those with hard parameter sharing. It would be interesting to research neural network multi-task learners with explanations, such as the Friendly-Enigma architecture, with soft parameter sharing to enhance the possible findings from Experiment 5, where we tested our architecture.

6.3.3 Actionable Explanations

In Section 5.1.4, we explain how the explanations in our experiments only give information about which feature is the most relevant to change the prediction. Further work might look at using regression learning to see whether one can learn an approximation of the value one needs to apply to change the prediction. If this is feasible, it is also interesting to look at selective explanations [Miller, 2019] and discuss the cost-benefit trade-off of multiple features with slight changes compared to a single or few features with more extensive changes.

6.3.4 Test Other Existing Methods for Generating Explanations

Experiment 2 tests generated explanations through C-CHVAE. It would be interesting to compare C-CHVAE to other methods, such as Shapley, SEDC, or others with respect to performance metrics (accuracy, precision, etc.) achieved by a model learning from a dataset generated by these methods. A new dataset would be required to compare these methods. One possible dataset where SEDC could be tested is the "20 Newsgroup" dataset [Ramon and Martens, 2020], where the prediction head can predict whether a document is "Medical" while the explanation head can provide explanations in terms of which features should be removed to change the classification.

6.3.5 Audience

The idea of the Friendly-Enigma architecture can also be extended to create explanations that match its audience. To some users, e.g., debuggers of the AI systems would potentially like to know the importance of each feature, i.e., their Shapley value, regular users might not. Adding a new head for a new type of explanation to tailor a new user builds upon ideas of Miller [2019], where users with different levels of experience require different types of explanations, e.g., beginners vs. experts.

6.3.6 Qualitative Metrics

As the quality of explanations is hard to measure through numbered metrics, it would be interesting to try these explanations on real users to evaluate their quality. Typical questions that need to be answered would be; "do you understand what went wrong?", "do you understand how you can solve the issue?" and "how feasible is it to solve the issue?". Possible metrics one could use are the ones listed in Section 2.2

6.3.7 Social Process

The taxonomy of Section 2.1 defines interpretability as the "ability to present in understandable terms to a human.". It also defines explainability as "capable of participating in a social process between two or more participants in which the model transfers knowledge to the explainee.". By these definitions, Friendly-Enigma architecture is an interpretable AI system. Further research on expanding the framework into a system capable of participating in the social process, e.g., through querying and multiple explanation formats, could fit the definition of explainable AI systems. In the bank loan example, someone tries to apply for a loan and gets rejected. If the explanation created is "you need 10 000 more income each year", but you can only get a raise of 1 000, this should be included in the query. The AI system can then adapt to the new information instead of creating the explanation, "you need 9 000 more income." if there are other possible outcomes. As mentioned in Section 2.2, AI systems are capable of finding information that we have yet to encounter ourselves. If the explanations are created by domain experts, they may not cover this scenario. If they are generated by existing methods, we might still not trust them when presented with the explanation, as they might be improbable. This further enhances the need for explanations as a social process, as there might exist solutions to which we would not be able to interpret a given explanation or a piece of information without questioning it and querying for other possible reasons before accepting the result.

6.4 Further Remarks

Some significant questions remain after this thesis is concluded. Although we have experimented with some of them in Chapter 5, others need further discussion and research. Drawing conclusions regarding these questions are out of scope for this thesis, but we present some of our views based on our research.

6.4.1 Generated Explanations

Hind et al. [2019] pivoted from most research on explanations referenced in this thesis by not addressing how to create the explanations but rather how to incorporate them into the opaque boxes to embed explanations as a native part of the system. They then state that by using domain experts to generate explanations, one can assure certain favorable features. However, this is time-consuming and often expensive, which would gate explanations from being widespread and accessible. Known methods for generating explanations or interpretable information were explored to alleviate the slow process of creating explanations through

domain experts.

An issue regarding the generated explanations is to know which method to use. Both the domain a method is suited for, and the form of the generated explanations differ. We explored SEDC and C-CHVAE, where SEDC generates counterfactuals based on the presence or absence of data, i.e., boolean values. C-CHVAE uses a variational auto-encoder, which is another opaque box, to generate counterfactuals. C-CHVAE is suited for continuous values, although whether it would learn on boolean features is yet to be tested. As C-CHVAE showed good results on both the GMSC and HELOC datasets, we modeled the Synthetic Data dataset's value types (continuous) after these datasets for our experiments. Another method for generating explanations is by using Shapley values. Shapley values calculate a "contribution value" for each feature value through coalitional game theory. The values are then applicable to most domains, but the explanation is not generalizable: it is locked to the specific feature values that were to be explained.

Barredo Arrieta et al. [2020] states that different audiences have different goals and needs for their explanations, which further contributes to the difficulty of generating explanations. SEDC, C-CHVAE, and Shapley generate interpretable information in the domain of an existing model. Thus, they all require the target audience to be knowledgeable in the domain and further complicate generating explanations. In Experiment 2, we tried to use C-CHVAE to generate explanations as counterfactuals that could make sense for the end-user applying for a loan, e.g., "You did not get a loan, but if you had these values instead, you would.". These counterfactuals might not explain what an engineer working on the network wants to know when trying to find whether or not there is a fault in the system but might be suitable for other audiences.

All methods we looked at had one thing in common: they all presented information in understandable terms for humans and were not capable of any social process. The methods followed the input-output principle of the AI systems; when given an input, an output is generated. By the definition of Section 2.1, these methods work towards interpretable AI. The shared issue further enhances the need for different methods for different audiences, as the methods only present one type of information. Suppose a form of information addresses all audiences' needs, which fits all domains and data formats. In that case, interpretable AI could fully replace explainable AI as Rudin [2019] proposes. However, as the definition of explainability is based on human interaction from social sciences, it is still likely that one would need to look into methods or frameworks, using one or more methods to facilitate the social process of explanation.

6.4.2 Is it OK to be Wrong?

Constructing an explanation as a human being is not always an easy task. Often it can be a challenging thing to achieve proficiently. The ability to create legitimate explanations requires much knowledge in several areas. The explainee needs to know a decent amount about the topic that is going to be explained. If knowledge about the topic does not reach very far or is inadequate, the explanation will likely end up being wrong. The audience will receive faulty explanations, which results in faulty knowledge about the topic, reducing the overall correct knowledge of the topic. The explainee also needs to know what knowledge the audience possesses on the topic. If the audience's knowledge is limited, abstractions are often used to convey the knowledge into a structure they can absorb and learn from [Miller, 2019]. The more knowledge the audience has, the fewer abstractions are necessary to explain. Therefore, it is also beneficial that the explainee has a decent knowledge on how to best transfer the knowledge, i.e., being pedagogical enough to teach the topic or know enough abstractions that work well between the topic and the audience.

If an AI system is supposed to create good or decent explanations, the system should have a decent knowledge of the following three topics. The first topic is what the AI system is predicting or classifying. This is the system's main job, and therefore crucial that the AI system has a good knowledge base on the subject to execute its purpose. The second topic is the knowledge about how proficient the user of the AI system is on the subject. If the users are experts, fewer abstractions and more pure data are often a viable and sought-after solution [Liao et al., 2020], which is a more straightforward explanation to generate. If the users are less experienced, more abstractions and less pure data are needed in the explanations, which is hard to create. The third topic is abstracting and generalize knowledge to users. The AI system needs to know how to translate each generated explanation to its audience so that varying audiences will understand the explanation.

The question that remains is, is it OK to be wrong? People err; it is a natural facet of being a human. Still, others can be empathetic towards the fact that people can make wrong decisions. Even experts in their respective fields make wrong decisions, less so than people who are not experts, but mistakes still occur. They may explain their reasoning well, but their base knowledge of the task might be flawed, meaning they might have good knowledge of the second and third subjects. However, their knowledge on the first subject is not sufficient and ends in some wrong decisions. Although the wrong decisions may be costly, it is often understandable, and the mistakes are seen with some level of empathy. Can this empathy be transferred to the AI system as well? Often, AI systems perform

better than many experts in a specific field. Given that they are explaining their decisions, it should be argued that the level of mistakes should be understood and accepted, the same way they are with other human experts.

Chapter 7

Conclusion

In a world where AI systems become ever prominent, a need for explanations without blind trust is necessary. Domains with high-risk decisions that affect lives should not tolerate faults leading to wrong decisions. The European High-level Expert Group on AI already proposes explanations as a requirement for all human-centric AI systems. Likewise, the Norwegian government has started a similar initiative to solidify the urgency of understanding these systems. In his critique towards the field, Miller says that too little focus on the social aspects of the AI systems has been incorporated.

Our architecture, Friendly-Enigma, shows that neural network multi-task learners can use the benefits gained from neural networks while also making the system more transparent by adding explanations. The novel sign-difference loss function we developed helps induce trustworthiness between the tasks of a multi-head neural network. The loss function makes sure that shared information between each head is utilized similarly.

In this thesis, we have shown that neural network multi-task learners can learn to both predict and explain well, given proper explanations in the training data. These explanations may come either from an expert or an external system. A question that still remains is whether the external systems that exist today are good enough. Nevertheless, with proper explanations, a neural network using the Friendly-Enigma architecture will generalize better on worse data than a regular neural network. This benefit comes from the multiple signals from each task, while Friendly-Enigma will perform just as well as a multi-task learner without

the sign-difference loss function.

Our experiments have shown promising results by using counterfactual explanations alongside neural network multi-task learners. These systems are a singular unit for classifying and explaining in contrast to the popular method of using post-hoc explanations, a secondary system to learn the first system's approximation. Counterfactual explanations are actionable and should be further researched and tested on real users of AI systems.

Bibliography

Ancona, M., Ceolini, E., Oztireli, C., and Gross, M. (2018). Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv: 1711.06104.

Audhkhasi, K., Osoba, O., and Kosko, B. (2013). Noise benefits in backpropagation and deep bidirectional pre-training. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Baxter, J. (1997). A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28:7–39.

BBC (2016). Artificial intelligence: Go master Lee Se-dol wins against AlphaGo program.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

CMS (2019). Statistics: Fines imposed over time. `"https://www.enforcementtracker.com/?insights"`. Accessed 05.10.2020.

Codella, N. C. F., Hind, M., Ramamurthy, K. N., Campbell, M., Dhurandhar, A., Varshney, K. R., Wei, D., and Mojsilovic, A. (2019). Teaching AI to explain its decisions using embeddings and multi-task learning. arXiv: 1906.02299.

Corder, G. W. (2009). Nonparametric statistics for non-statisticians: a step-by-step approach.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv: 1702.08608.

European Parliament and Council of the European Union (2018). GDPR: Right of access by the data subject. "https://gdpr-info.eu/art-15-gdpr/". Accessed 12.10.2020.

Faigle, U. and Kern, W. (1992). The Shapley value for cooperative games under precedence constraints. *International Journal of Game Theory*, 21(3):249–266.

FICO (2018). Explainable machine learning challenge. "https://community.fico.com/s/explainable-machine-learning-challenge". Accessed 11.03.2021.

Frye, C., Rowat, C., and Feige, I. (2020). Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability. arXiv: 1910.06358.

Gade, K., Geyik, S., Kenthapadi, K., Mithal, V., and Taly, A. (2020). Explainable AI in industry: Practical challenges and lessons learned. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, WWW '20, page 303–304, New York, NY, USA. Association for Computing Machinery.

Gerbert, P., Reeves, M., Steinhäuser, S., and Ruwolt, P. (2017). Is your business ready for artificial intelligence? Accessed 06.10.2020.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5).

High-Level Expert Group on AI (2019). Building trust in human-centric AI. "https://ec.europa.eu/futurium/en/ai-alliance-consultation/". Accessed 30.09.2020.

Hind, M., Wei, D., Campbell, M., Codella, N. C. F., Dhurandhar, A., Mojsilović, A., Natesan Ramamurthy, K., and Varshney, K. R. (2019). TED: Teaching AI to explain its decisions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 123–129, New York, NY, USA. Association for Computing Machinery.

Jillson, E. (2021). Aiming for truth, fairness, and equity in your company's use of AI. https://www.ftc.gov/news-events/blogs/business-blog/2021/04/aiming-truth-fairness-equity-your-companys-use-ai. Accessed 23.05.2021.

Kaggle (2011). Give me some credit. "https://www.kaggle.com/c/GiveMeSomeCredit/overview/description". Accessed 03.03.2021.

Kvamme, J. and Larsen, P. (2021). Friendly Enigma. `https://github.com/Paalar/friendly-enigma`.

Lardinois, F. (2021). Google Cloud launches Vertex AI, a new managed machine learning platform. `https://techcrunch.com/2021/05/18/google-cloud-launches-vertex-a-new-managed-machine-learning-platform/?guccounter=2`. Accessed 23.05.2021.

Lecue, F. and Wan, D. (2018). Understanding machines: Explainable AI. `"https://www.accenture.com/_acnmedia/pdf-85/accenture-understanding-machines-explainable-ai.pdf"`. Accessed 05.10.2020.

Letzter, R. (2016). Amazon just showed us that 'unbiased' algorithms can be inadvertently racist. `https://www.businessinsider.com/how-algorithms-can-be-racist-2016-4?r=US&IR=T`. Accessed 24.05.2021.

Leung, K. (2021). The dying relu problem, clearly explained. `https://towardsdatascience.com/the-dying-relu-problem-clearly-explained-42d0c54e0d24`. Accessed 09.05.2021.

Liao, Q. V., Gruen, D., and Miller, S. (2020). Questioning the AI: Informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM.

Lipton, P. (1990). Contrastive explanation. *Royal Institute of Philosophy Supplement*, 27:247–266.

Lundberg, S. and Lee, S. (2016). An unexpected unity among methods for interpreting model predictions. In *NIPS 2016 Workshop on Interpretable Machine Learning in Complex Systems*, volume abs/1611.07478.

Martens, D. and Provost, F. (2014). Explaining data-driven document classifications. *MIS Quarterly*, 38(1):73–100.

McDonnell, T., Lease, M., Kutlu, M., and Elsayed, T. (2016). Why is that relevant? Collecting annotator rationales for relevance judgments. In *HCOMP*.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Ministry of Local Government and Modernisation (2020). The national strategy for artificial intelligence. `https://www.regjeringen.no/en/dokumenter/nasjonal-strategi-for-kunstig-intelligens/id2685594/`. Accessed 24.10.2020.

Molnar, C. (2019). Interpretable machine learning. https://christophm.github.io/interpretable-ml-book/. Accessed 10.12.2020.

Nathan, L. (2020). How to compare ROC curves for model selection. https://biasedml.com/roc-comparison/. Accessed 28.05.2021.

Patel, A. and Kosko, B. (2009). Error-probability noise benefits in threshold neural signal detection. *Neural Networks*, 22(5):697 – 706. Advances in Neural Networks Research: IJCNN2009.

Pawelczyk, M., Broelemann, K., and Kasneci, G. (2020). Learning model-agnostic counterfactual explanations for tabular data. *Proceedings of The Web Conference 2020*.

Pratt, L. Y., Mostow, J., and Kamm, C. A. (1991). Direct transfer of learned information among neural networks. In *AAAI'91*, page 584–589. AAAI Press.

Ramon, Y. and Martens, D. (2020). Evidence counterfactuals for explaining predictive models on big data. https://www.kdnuggets.com/2020/05/evidence-counterfactuals-predictive-models-big-data.html. Accessed 25.05.2021.

Ramon, Y., Martens, D., Provost, F., and Evgeniou, T. (2020). A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C. *Advances in Data Analysis and Classification*, 14(4):801–819.

Ribeiro, M. (2016). LIME. https://github.com/marcotcr/lime. Accessed 15.09.2020.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Rocca, J. (2021). Understanding variational autoencoders (vaes). https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. Accessed 12.04.2021.

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., and Dietterich, T. G. (2005). To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. arXiv: 1706.05098.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent multi-task architecture learning. In *Proceedings of 33nd AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 4822–4829. AAAI Press. 33rd AAAI Conference on Artificial Intelligence - AAAI 2019 ; Conference date: 27-01-2019 Through 01-02-2019.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Sagar, R. (2020). What does freezing a layer mean and how does it help in fine tuning neural networks. Accessed 22.05.2021.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–328.

Schaefferkoetter, J., Yan, J., Ortega, C., Sertic, A., Lechtman, E., Eshet, Y., Metser, U., and Veit-Haibach, P. (2020). Convolutional neural networks for improving image quality with noisy PET data. *EJNMMI Research*, 10:105–115.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3145–3153. JMLR.org.

Završnik, A. (2020). Criminal justice, artificial intelligence systems, and human rights. *ERA Forum*, 20:567–583.

Zolas, N., Kroff, Z., Brynjolfsson, E., McElheran, K., Beede, D. N., Buffington, C., Goldschlag, N., Foster, L., and Dinlersoz, E. (2020). Advanced technologies adoption and use by U.S. firms: Evidence from the annual business survey. Working Paper 28290, National Bureau of Economic Research.

Appendix A

Experiment 7 Results

Feature	Mean	Standard deviation	Min	25%	50%	75%	Max
SeriousDlqin2yrs	0.0668	0.2497	0.0	0.0	0.0	0.0	1.0
RevolvingUtilizationOfUnsecuredLines	6.0484	249.7553	0.0	0.0298	0.1541	0.5590	50708.0
age	52.2952	14.7718	0.0	41.0	52.0	63.0	109.0
NumberOfTime30-59DaysPastDueNotWorse	0.4210	4.1927	0.0	0.0	0.0	0.0	98.0
DebtRatio	353.0050	2037.8185	0.0	0.1750	0.3665	0.8682	329664.0
MonthlyIncome	6418.4549	12890.3955	0.0	3903.0	5400.0	7400.0	3008750.0
NumberOfOpenCreditLinesAndLoans	8.4527	5.1459	0.0	5.0	8.0	11.0	58.0
NumberOfTimes90DaysLate	0.2659	4.1693	0.0	0.0	0.0	0.0	98.0
NumberRealEstateLoansOrLines	1.0182	1.1297	0.0	0.0	1.0	2.0	54.0
NumberOfTime60-89DaysPastDueNotWorse	0.2403	4.1551	0.0	0.0	0.0	0.0	98.0
NumberOfDependents	0.7572	1.1004	0.0	0.0	0.0	1.0	20.0

Table 1: Original data distribution for the GMSC dataset

Feature	Mean	Standard deviation	Min	25%	50%	75%	Max
SeriousDlqin2yrs	0.9331	0.2497	0.0	1.0	1.0	1.0	1.0
RevolvingUtilizationOfUnsecuredLines	0.3931	0.6803	-1.0	0.0	0.0	1.0	26.0
age	52.2952	14.7718	0.0	41.0	52.0	63.0	109.0
NumberOfTime30-59DaysPastDueNotWorse	0.2377	0.6782	0.0	0.0	0.0	0.0	21.0
DebtRatio	815.1248	39254.7124	-1.0	0.0	3.0	28.0	11394313.0
MonthlyIncome	9806.4155	16593.1648	1.0	2065.0	4919.5	11096.0	1051889.0
NumberOfOpenCreditLinesAndLoans	6.4817	3.7675	0.0	4.0	6.0	9.0	30.0
NumberOfTimes90DaysLate	0.1360	0.8428	0.0	0.0	0.0	0.0	30.0
NumberRealEstateLoansOrLines	1.1447	1.1068	0.0	0.0	1.0	2.0	9.0
NumberOfTime60-89DaysPastDueNotWorse	0.1298	0.8191	0.0	0.0	0.0	0.0	33.0
NumberOfDependents	0.7635	1.1010	0.0	0.0	0.0	1.0	20.0

Table 2: GMSC data description for all of the generated C-CHVAE samples

Feature	Mean	Standard deviation	Min	25%	50%	75%	Max
SeriousDlqin2yrs	0.9955	0.0668	0.0	1.0	1.0	1.0	1.0
RevolvingUtilizationOfUnsecuredLines	3.8039	131.7972	-16.0	-0.1075	0.1727	0.8762	8327.0
age	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NumberOfTime30-59DaysPastDueNotWorse	1.7722	10.1760	-7.0	0.0	0.0	1.0	95.0
DebtRatio	-177.7989	8354.7926	-478023.0	-15.0	-0.6335	1.3540	38780.0
MonthlyIncome	-3017.5611	15334.1210	-341663.0	-4636.0	462.0	3224.0	247760.0
NumberOfOpenCreditLinesAndLoans	2.6203	5.3729	-16.0	-1.0	2.0	6.0	46.0
NumberOfTimes90DaysLate	1.4997	9.3753	-6.0	0.0	0.0	1.0	91.0
NumberRealEstateLoansOrLines	-0.0361	1.6818	-6.0	-1.0	0.0	1.0	28.0
NumberOfTime60-89DaysPastDueNotWorse	1.2559	9.4281	-9.0	0.0	0.0	0.0	90.0
NumberOfDependents	-0.0043	0.0321	-0.2427	0.0	0.0	0.0	0.0

Table 3: GMSC Delta data description of the generated C-CHVAE counterfactuals

Feature	Mean	Standard deviation	Min	25%	50%	75%	Max
RiskPerformance	0.4796	0.4996	0.0	0.0	0.0	1.0	1.0
ExternalRiskEstimate	71.9874	10.1296	0.0	64.0	72.0	80.0	94.0
MSinceOldestTradeOpen	195.9080	101.5556	0.0	131.0	183.0	255.0	803.0
MSinceMostRecentTradeOpen	9.5884	12.9633	0.0	3.0	6.0	12.0	383.0
AverageMInFile	78.77815	34.0660	4.0	57.0	76.0	97.0	383.0
NumSatisfactoryTrades	21.1214	11.3213	0.0	13.0	20.0	28.0	79.0
NumTrades60Ever2DerogPubRec	0.5814	1.2387	0.0	0.0	0.0	1.0	19.0
NumTrades90Ever2DerogPubRec	0.3847	0.9932	0.0	0.0	0.0	0.0	19.0
PercentTradesNeverDelq	92.3599	11.7728	0.0	89.0	97.0	100.0	100.0
MSinceMostRecentDelq	11.1514	18.4475	0.0	0.0	0.0	15.0	83.0
MaxDelq2PublicRecLast12M	5.7579	1.6445	0.0	5.0	6.0	7.0	9.0
MaxDelqEver	6.3745	1.8491	2.0	6.0	6.0	8.0	8.0
NumTotalTrades	22.6354	12.9999	0.0	13.0	21.0	30.0	104.0
NumTradesOpeninLast12M	1.8638	1.8280	0.0	0.0	1.0	3.0	19.0
PercentInstallTrades	34.6186	17.9534	0.0	21.0	33.0	45.0	100.0
MSinceMostRecentInqexcl7days	1.8926	4.2915	0.0	0.0	0.0	1.0	24.0
NumInqLast6M	1.4559	2.1361	0.0	0.0	1.0	2.0	66.0
NumInqLast6Mexcl7days	1.3971	2.0961	0.0	0.0	1.0	2.0	66.0
NetFractionRevolvingBurden	34.2008	29.0128	0.0	8.0	28.0	56.0	232.0
NetFractionInstallBurden	44.7986	38.3269	0.0	0.0	52.0	80.0	471.0
NumRevolvingTradesWBalance	4.0372	3.0409	0.0	2.0	3.0	5.0	32.0
NumInstallTradesWBalance	2.2681	1.7117	0.0	1.0	2.0	3.0	23.0
NumBank2NatlTradesWHighUtilization	1.0277	1.5122	0.0	0.0	1.0	1.0	18.0
PercentTradesWBalance	66.3278	22.1971	0.0	50.0	67.0	83.0	100.0

Table 4: Original data distribution for the HELOC dataset

Feature	**Mean**	**Standard deviation**	**Min**	**25%**	**50%**	**75%**	**Max**
RiskPerformance	0.5203	0.4996	0.0	0.0	1.0	1.0	1.0
ExternalRiskEstimate	71.9874	10.1296	0.0	64.0	72.0	80.0	94.0
MSinceOldestTradeOpen	195.9080	101.5556	0.0	131.0	183.0	255.0	803.0
MSinceMostRecentTradeOpen	9.5420	12.7679	-76.0	1.0	8.0	17.0	228.0
AverageMInFile	78.7781	34.0660	4.0	57.0	76.0	97.0	383.0
NumSatisfactoryTrades	20.8347	13.7426	-62.0	12.0	19.0	28.0	132.0
NumTrades60Ever2DerogPubRec	0.5108	0.9832	-3.0	0.0	0.0	1.0	14.0
NumTrades90Ever2DerogPubRec	0.3561	0.7544	-3.0	0.0	0.0	1.0	10.0
PercentTradesNeverDelq	93.0969	10.5310	-57.0	89.0	96.0	100.0	122.0
MSinceMostRecentDelq	9.9818	16.2163	-44.0	-1.0	6.0	19.0	152.0
MaxDelq2PublicRecLast12M	5.8393	1.6382	-14.0	5.0	6.0	7.0	10.0
MaxDelqEver	6.5272	2.0345	-22.0	6.0	7.0	8.0	12.0
NumTotalTrades	22.4502	15.5666	-81.0	12.0	21.0	31.0	150.0
NumTradesOpeninLast12M	1.8524	1.9795	-15.0	1.0	2.0	3.0	40.0
PercentInstallTrades	34.4161	20.2267	-30.0	21.0	34.0	47.0	267.0
MSinceMostRecentInqexcl7days	1.7763	3.8975	-39.0	-1.0	2.0	4.0	22.0
NumInqLast6M	1.3944	2.2918	-6.0	0.0	1.0	2.0	76.0
NumInqLast6Mexcl7days	1.3574	2.2734	-5.0	0.0	1.0	2.0	67.0
NetFractionRevolvingBurden	33.8501	32.3518	-68.0	11.0	30.0	53.0	559.0
NetFractionInstallBurden	45.2961	44.1770	-379.0	17.0	45.0	74.0	589.0
NumRevolvingTradesWBalance	3.9992	3.3001	-11.0	2.0	4.0	6.0	36.0
NumInstallTradesWBalance	2.2492	1.8504	-5.0	1.0	2.0	3.0	29.0
NumBank2NatlTradesWHighUtilization	0.9579	1.3740	-3.0	0.0	1.0	2.0	15.0
PercentTradesWBalance	66.6465	25.5415	-47.0	50.0	66.0	83.0	470.0

Table 5: HELOC data description for all of the generated C-CHVAE samples

Feature	**Mean**	**Standard deviation**	**Min**	**25%**	**50%**	**75%**	**Max**
RiskPerformance	0.9098	0.2864	0.0	1.0	1.0	1.0	1.0
ExternalRiskEstimate	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MSinceOldestTradeOpen	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MSinceMostRecentTradeOpen	0.0617	15.8168	-62.0	-9.0	0.0	8.0	262.0
AverageMInFile	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NumSatisfactoryTrades	-0.4865	12.9070	-56.0	-8.0	0.0	8.0	140.0
NumTrades60Ever2DerogPubRec	0.1248	1.3129	-4.0	-1.0	0.0	1.0	9.0
NumTrades90Ever2DerogPubRec	0.0339	1.0902	-4.0	0.0	0.0	0.0	10.0
PercentTradesNeverDelq	-1.3187	13.1938	-73.0	-7.0	0.0	5.0	179.0
MSinceMostRecentDelq	-0.0023	21.7504	-129.0	-12.0	-1.0	10.0	105.0
MaxDelq2PublicRecLast12M	-0.1913	1.9211	-9.0	-1.0	0.0	1.0	22.0
MaxDelqEver	-0.2487	1.9870	-9.0	-1.0	0.0	1.0	26.0
NumTotalTrades	-0.1936	15.1811	-69.0	-10.0	0.0	9.0	124.0
NumTradesOpeninLast12M	0.0567	2.2578	-9.0	-1.0	0.0	1.0	18.0
PercentInstallTrades	0.5905	23.9721	-100.0	-15.0	0.0	16.0	101.0
MSinceMostRecentInqexcl7days	-0.3223	4.9989	-18.0	-3.0	-1.0	2.0	29.0
NumInqLast6M	0.0840	2.3982	-9.0	-1.0	0.0	1.0	13.0
NumInqLast6Mexcl7days	0.0488	2.4025	-9.0	-1.0	0.0	1.0	14.0
NetFractionRevolvingBurden	5.2698	34.3276	-124.0	-17.0	5.0	28.0	116.0
NetFractionInstallBurden	1.4502	51.7870	-188.0	-35.0	1.0	37.0	453.0
NumRevolvingTradesWBalance	0.1856	3.4807	-15.0	-2.0	0.0	2.0	19.0
NumInstallTradesWBalance	0.0593	2.0858	-9.0	-1.0	0.0	1.0	14.0
NumBank2NatlTradesWHighUtilization	0.2079	1.7306	-6.0	-1.0	0.0	1.0	10.0
PercentTradesWBalance	1.9292	27.6252	-122.0	-17.0	2.0	20.0	99.0

Table 6: HELOC Delta data description of the generated C-CHVAE counterfactuals

Johannes Kvamme and Pål-Edward Larsen

Achieving Trustable Explanations Through Multi-Task Learning Neural Networks

NTNU
Kunnskap for en bedre verden