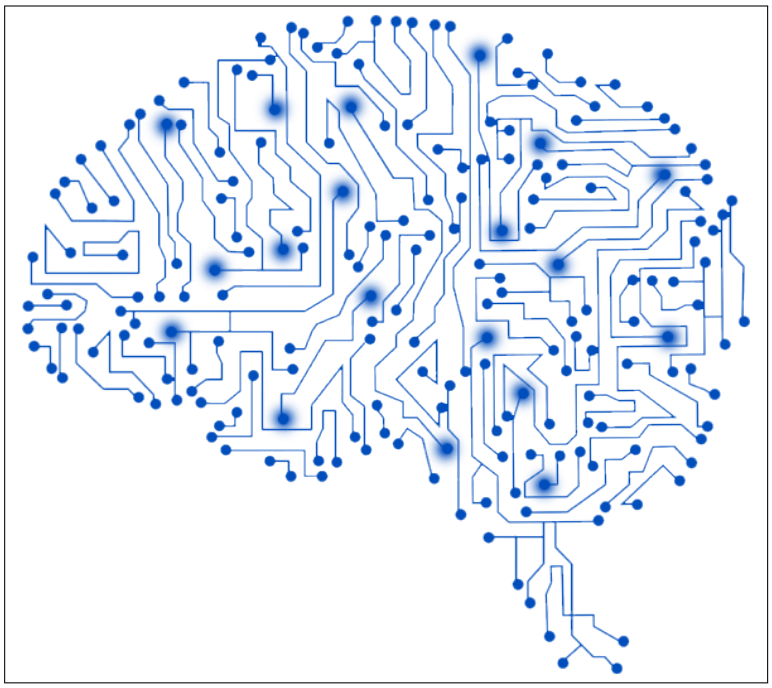
Universiteit Utrecht

Faculty of Science

STRIPAI: Determining the suitability of implementing deep learning principles in new domains

Master Thesis

Danny Knemeijer | Business Informatics



Supervisors:

Dr. Marco Spruit
Utrecht University - Faculty of Science

Dr. Matthieu Brinkhuis
Utrecht University - Faculty of Science

May 27, 2019

Abstract

Deep Learning and Reinforcement Learning are techniques that are being applied more and more. However, the combination of the two techniques sees little use outside of the gaming domain. This study aims to determine when and how these techniques can be used in new domains. This thesis asks the question how the two techniques can be applied to applications in new domains in order to improve the usability of these applications.

Based on the used methodology, a literature research on how to determine whether a domain is actually suitable for the application of Deep Learning and Reinforcement Learning is conducted. After this, a framework is shown which can be used to transform an application in such a way that it represents a game with rules, inputs, and output. Lastly, the techniques will be applied to an existing application in the prescriptive healthcare domain.

The results indicate that applying these techniques to the application in the prescriptive healthcare domain did not lead to a significant increase in the effectiveness of the application. Furthermore, theoretical results showed that there was also no significant increase in the efficiency of the application. This means that the implementation of Deep Learning and Reinforcement Learning principles did not lead to a significant increase in application usability. The results from this experiment can be used in order to better determine which domains will be suitable for these kinds of implementations.

Acknowledgements

Out of all of the chapters I had to write for this thesis, I never thought writing the acknowledgements would be the hardest chapter out of all of them. Over the last period of time, I have had the privilege to write this thesis, and to speak with a lot of people about the topic I enjoy the most.
What started out as a great idea in the first half of 2018, has since turned into the thesis you are reading today. And with my thesis defense around the corner, this chapter of my life is coming to a close. It has been a great journey, and I would like to take this opportunity to thank some of the people that have joined my journey.
First and foremost, I would like to thank my first supervisor, Dr. Marco Spruit. Together, we have had many discussions about this topic, and his expertise on both the sector of medical informatics, and data science, has helped a lot in writing this thesis. Also many thanks to my second supervisor, Dr. Matthieu Brinkhuis. He has been a great help in providing feedback along the way, and in refining the details of thesis.
As for more specific topics in this thesis, I would like to thank Ian Zhengru Shen for his additional knowledge on the STRIP Assistant, and Edwin Brinkhuis for collaborating with me on the technical aspect of this thesis. Without these two people, it would not have been possible for me to perform any kind of experiment on the STRIP Assistant. Lastly, I would also like to thank Ad Feelders, whose course Data Mining got me very interested into the field of data mining, neural networking and machine learning.
I would like to thank my friends and fellow students, both for being challenging discussion partners, as well as serving as a distraction for when I was stuck with writing this thesis. A special thank you goes out to Study Association Sticky, where I could always drop in to have a cup of coffee, and where I have done a lot of exciting things with my fellow students.
Lastly, I would like to thank my family and my girlfriend, for being there whenever I needed them the most. Without their support, I would have never pulled off this amazing achievement.

- Danny Knemeijer, May 2019

Contents

List of Figures

List of Tables

List of Acronyms

ADS Applied Data Science

AI Artificial Intelligence

ANN Artificial Neural Network

BP Back-propagation

CNN Convolutional Neural Network

CRISP-DM CRoss Industry Standard Process for Data Mining

CTC Collectionist Temporal Classification

DFF Deep Feed Forward

DL Deep Learning

DNN Deep Neural Network

DQN Deep Q-Network

DRL Deep Reinforcement Learning

DSR Design Science Research

FNN Feedforward Neural Network

GIVE Gebruik, Indicatie, Veiligheid & Effectiviteit

GPU-MPCNN GPU-based MPCNN

KDD Knowledge Discovery for Databases

KDP Knowledge Discovery Process

LSTM Long Short-Term Memory

MAI Medical Appropriateness Index

MCST Monte Carlo Search Tree

MDP Markov Decision Process

MP Max Pooling

MPCNN Max Pooling Convolutional Neural Network

NDP Neuro-Dynamic Programming

NLP Natural Language Processing

POM Prescription Optimization Method

Q-Learning *Quality*-Learning

RL Reinforcement Learning

RNN Recurrent Neural Network

SARSA State-Action-Reward-State-Action

START Screening Tool to Alert to Right Treatment

STOPP Screening Tool of Older People's Prescriptions

STRIP Systematic Tool to Reduce Inappropriate Prescribing

STRIPA STRIP Assistant

UISE Use, Indication, Safety & Effectiveness

List of Terms

***Quality*-Learning** A Markov Decision Process (MDP) algorithm that uses a quality function to describe the quality of the state the algorithm currently is in

AlphaGo Zero An algorithm developed by Google Deepmind that is capable of playing the game of Go without requiring any data from games played by humans, and by using only the rules of the game as input

Applied Data Science The knowledge discovery process in which analytical applications are designed and evaluated to improve the daily practices of domain expert

Artificial Intelligence A machine that mimics cognitive functions that humans associate with other human minds, such as 'learning' and 'problem solving'

Artificial Neural Network A computational model which is loosely inspired by the human brain as it consists of an interconnected network of simple processing units (artificial neurons) that learns from experience by modifying its connections

Back-propagation A method that is used to calculate the gradients used for the calculation of the weights used in a neural network

Convolutional Neural Network Multi-layered Neural Network specialized on recognizing visual patterns directly from image pixels (LeCun, Bottou, Bengio, & Haffner, 1998)

CRoss Industry Standard Process for Data Mining A Knowledge Discovery Process used in the (Applied) Data Science field that consists of six different stages which allows the user to streamline and standardize the Data Mining process

Data Science The extraction of actionable knowledge directly from data through a process of discovery, or hypothesis formulation and hypothesis testing

Deep Feed Forward Same as regular Feed Forward, albeit with multiple hidden layers

Deep Learning A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification

Deep Neural Network An Artificial Neural Network with multiple hidden layers

Deep Reinforcement Learning A type of machine learning algorithm in which a deep learning algorithm represents the internal agent of a reinforcement learning implementation as a way to represent its' policy function or notion of expected rewards.

Design Science Research Paradigm that seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts

Feedforward Neural Network An Artificial Neural Network in which connections between the nodes form an acyclic graph (Zell, 1994)

GPU-based MPCNN *See Max Pooling Convolutional Neural Network*

Knowledge Discovery Process Process that creates the context for developing the tools needed to control the flood of data facing organizations that depend on ever-growing databases of business, manufacturing, scientific, and personal information

Long Short-Term Memory Long Short-Term Memory networks are a variant of Recurrent Neural Networks (RNNs) that are capable of storing long-term dependencies

Markov Decision Process A model for sequential decision making when outcomes are uncertain (Puterman, 2014)

Max Pooling The procedure of taking an $N \times N$ matrix, and converting it into a smaller $M \times M$ by means dividing the matrix into smaller sections and selecting the largest value as the new value

Max Pooling Convolutional Neural Network Combination of Max Pooling and an Convolutional Neural Network, which contains both convolutional and subsampling layers

Neuro-Dynamic Programming A class of dynamic programming methods for control and sequential decision making under uncertainty

Recurrent Neural Network a type of Artificial Neural Network where connections between its components form a directed (a)cyclic graph, and keeps an unlimited history which is represented by recurrently connected components (Mikolov et al., 2010)

Reinforcement Learning A general class of algorithms in the field of machine learning that aims at allowing an agent to learn how to behave in an environment, where the only feedback consists of a scalar reward signal

State-Action-Reward-State-Action An on-policy learning algorithm used for learning Markov Decision Process policies

STRIP Assistant A web-based application that aims to assist General Practitioners (GPs) and pharmacists with pharmacotherapeutic analysis of patients' medical records (Meulendijk et al., 2015)

1 Introduction

Artificial Intelligence (AI) is defined by Russell, Norvig, Canny, Malik, and Edwards (2003) as "a machine that mimics cognitive functions that humans associate with other human minds, such as 'learning' and 'problem solving' ". A survey by Gartner of 3160 CIOs from almost 100 countries, shows that making progress with glsai initiatives is one of the top-five priorities for 2018. 21% of the CIOs are already experimenting with it, or have short-term plans for this, and 25% have medium- or long-term plans for this. As such, popular technologies from the field of AI can be found in the Gartner Hype Cycle. Deep Learning and Machine Learning are expected to hit the 'Plateau of Productivity' in 2 to 5 years. This means that these technologies will be adopted by the mainstream. Another popular AI technology, Reinforcement learning, is expected to hit this plateau in 5 to 10 years.

Deep Learning (DL) is defined by Deng and Yu (2014) as "a class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification". Reinforcement Learning (RL) is defined by van Otterlo and Wiering (2012) as "a general class of algorithms in the field of machine learning that aims at allowing an agent to learn how to behave in an environment, where the only feedback consists of a scalar reward signal".

Recently, these two technologies have been combined by Google to create *AlphaGo Zero*. This AI uses a new approach that combines supervised learning from human expert games, and reinforcement learning from games that it played against itself, to master the game of Go, long seen as the hardest game to master for an AI due to the enormous search space as described by Silver, Schrittwieser, et al. (2017). The combination of Deep Learning and Reinforcement Learning is called Deep Reinforcement Learning (DRL). AlphaGo Zero has also since mastered the game of (Japanese) Chess (Silver, Hubert, et al., 2017), but also several asymmetrical games (Tuyls et al., 2018), and several different games on the Atari 2600 (Mnih et al., 2015; Wang et al., 2016).

Surprisingly, these technologies have mostly been applied in the gaming domain. They have also been applied separately from each other for Natural Language Processing (NLP) (Bordes, Glorot, Weston, & Bengio, 2012; Socher, Huang, Pennin, Manning, & Ng, 2011), speech recognition (Hinton et al., 2012; Seide, Li, & Yu, 2011), and computer vision (Krizhevsky, Sutskever, & Hinton, 2012). Attempts have been made to implement these techniques within the medical domain. This has been done for both deep learning (Valk, 2018), and reinforcement learning implementations (Doorhof, 2018).

However, outside of the gaming domain, DRL has not yet been implemented widely. This is due to the fact that for DRL to be implemented efficiently and effectively, the application domain needs to be modeled as a game with a set of rules that transforms input to output. For example, while breakthroughs in the medical domain are being made by Poplin et al. (2018) on predicting cardiovascular risks through deep reinforcement learning and computer vision using a dataset of retinal images, the application of these techniques in this domain remains limited. However, the domain would most likely be well suited for these technologies.

This thesis is ordered as follows. First, Chapter 2 gives an overview of the research methodology. Chapter 3 gives a technical, in-depth description more about Deep Learning and

Reinforcement Learning, and how these two techniques can be combined into Deep Reinforcement Learning. Chapter 4 introduces a method of determining whether a domain is predicted to be well suited for the implementation of DRL principles. Chapter 5 goes into detail on the STRIP Assistant (STRIPA), a tool used into the patient-centric healthcare domain to combat and prevent polypharmacy. Chapter 6 will introduce a method that will help to try and implement Deep Reinforcement Learning into the STRIPA, after which Chapter 7 will discuss the implementation process. Chapter 8 will present the results, after which Chapter 9 and Chapter 10 will give a definitive answer to the main question of this thesis.

1.1 Research Context

This section will aim to give further insights into which context this research will be conducted, from both a technical and a domain perspective. This is done in order to explain where this research is positioned, and will help put the problem statement into perspective. Section 1.1.1 will first explain the domain perspective, and explain what kind of research will be conducted. Then, Section 1.1.2 will highlight a technical implementation of Deep Reinforcement Learning that will serve as an example, and will show the technical perspective of this research.

1.1.1 Applied Data Science.

This research will aim to apply Deep Reinforcement Learning into the patient-centric healthcare domain. The research involves a mix of domain expertise, engineering, and machine learning. As such, it can be classified as an ADS problem. Applied Data Science (ADS) is defined by Spruit and Jagesar (2016) as "the knowledge discovery process in which analytical applications are designed and evaluated to improve the daily practices of domain experts." In the case, the STRIP Assistant is the analytical application that will be evaluated (and hopefully improved by implementing DRL principles). The domain experts will be the general practitioners, who will be using the STRIP Assistant. The workings of the STRIP Assistant will be further discussed in Chapter 5.

Applied Data Science is based on Data Science, which is defined by Pritzker and May (2015) as "the extraction of actionable knowledge directly from data through a process of discovery, or hypothesis formulation and hypothesis testing." The difference is highlighted in Figure 1, which is based on (Pritzker & May, 2015; Spruit & Jagesar, 2016).

Both Data Science and Applied Data Science aim to solve problems involving a mixture of domain knowledge, engineering, and statistics & machine learning. However, ADS is more geared towards the use of analytical algorithms, such as machine learning techniques, to solve problems.

1.1.2 Go: "The grand challenge of AI".

Using AI to learn how to play games and beat human players is not a new field. Thorp and Walden used computer analysis in 1972 to research the principles of Go using smaller versions of the classic 19×19 board. Instead, they started off with a very small board, then increasing it. The largest achieved sizes were 1×5, 2×4, and 3×4. Since then, many games have been mastered by means of computerized play. However, Go was long believed to be the 'grand challenge' of Artificial Intelligence (Levinovitz, 2014; Silver, Schrittwieser, et al., 2017). This is due to the extremely large complexity of Go, which has a state-space

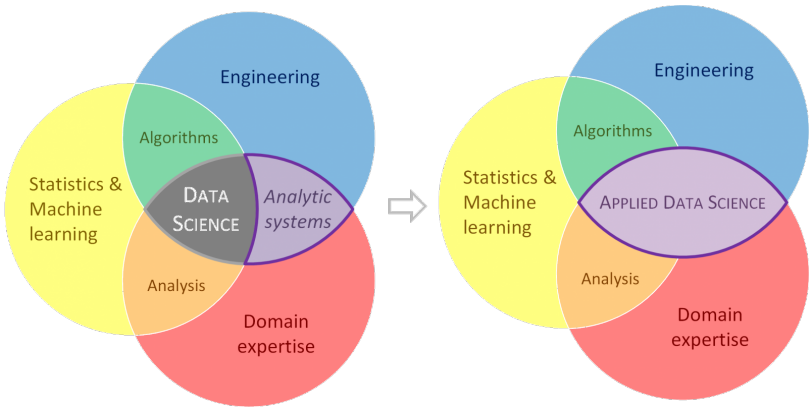


Figure 1. Diagram highlighting the difference between Data Science and Applied Data Science (ADS). As shown in this diagram, ADS has a more distinct focus on the field between engineering and domain expertise.

complexity of 10^{172}, and a game-tree complexity of 10^{320} (Van Den Herik, Uiterwijk, & Van Rijswijck, 2002). Compare this to Chess, which is believed to be the second-most complex game to crack for an AI, but has complexities of 10^{46} and 10^{123} respectively. The state-space complexity of a game is defined by Allis (1994) as "the number of legal game positions reachable from the initial starting position(s) of the game". The game-tree complexity of a game is defined as "the number of leaf nodes in the solution search tree of the initial starting position(s) of the game"(Allis, 1994).
In 2015 however, the curtain fell for Go when AlphaGo, developed by Google Deepmind, won 5-0 against 3-time European Go champion Fan Hui (Silver & Hassabis, 2016). To achieve this, Google used a RL algorithm with a Monte Carlo Search Tree (MCST) implementation. Then, by combining this with a deep neural network as a guide, AlphaGo was able to simulate thousands of possible moves in order to select the most optimal one. An improved version called AlphaGo Zero was introduced in 2017 by Silver, Schrittwieser, et al.. This version did not require any data from games played by humans, and was even stronger than AlphaGo, and all of its successors after 40 days of learning.

1.2 Problem Statement

This section will highlight the current problem this research is aiming to solve. This is done by explaining the current state of DRL, and how it is mostly applied within the gaming domain. It then goes on to ask why this is not done in other domains, and will then introduce the main problem statement for this thesis.
As stated, Deep Reinforcement Learning is mostly applied within the gaming domain. In this domain, the algorithm is trained by applying it to several popular Atari Games, as described by (Mnih et al., 2015). Figure 2 gives an overview of the games played by AlphaGo Zero. As shown, 60% of the games played resulted in an AI capable of playing above human-level play. The other 40% of games are ones that require a higher level of planned strategy, which is not yet able to be grasped by any form of AI, including Deep Reinforcement Learning (Mnih et al., 2015). However, DRL can not only be applied to 2D games, like the ones on the Atari, but also on 3D games. This is described in a paper

by (Lample & Chaplot, 2017), who built an AI capable of outperforming human players in the game DOOM.

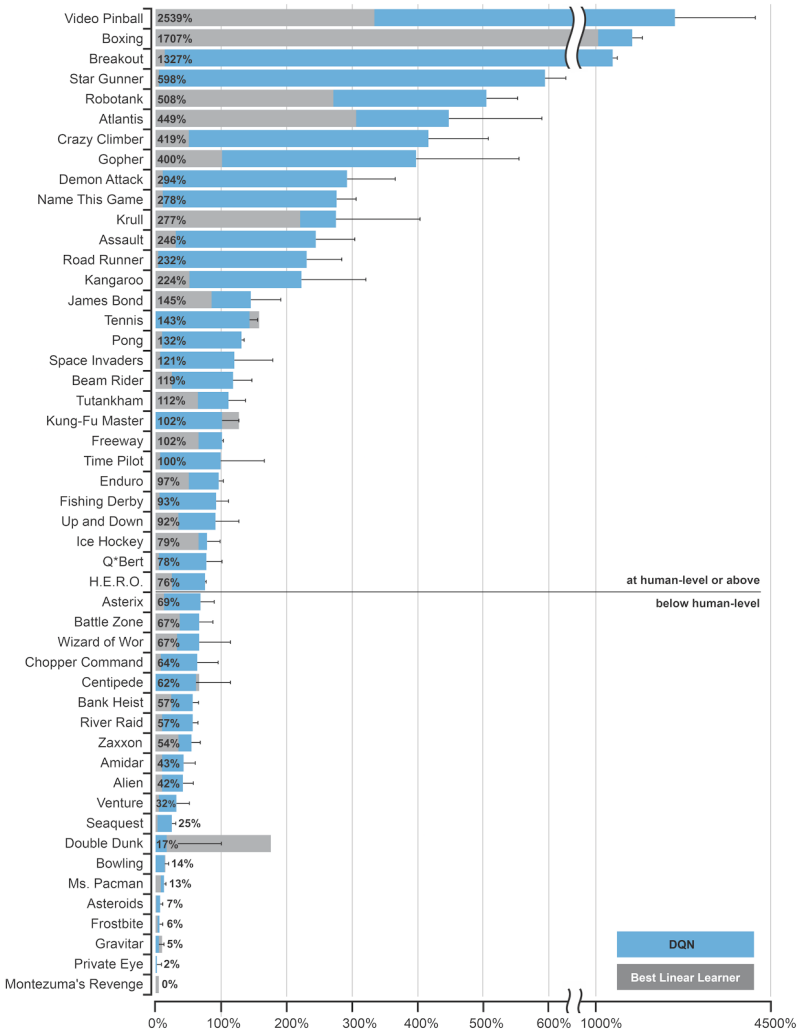


Figure 2. Distribution of games played with DRL algorithm, showing that AlphaGo Zero surpasses human-level play in 60% of the games played

Both these 2D and 3D games share the same abstract structure. They can both be defined as a piece of software that transforms input into output, according to a set of rules. In the case of video games, (Prema & Ramadoss, 2008) describe the input as information such as data or commands, that is entered into the game at run-time from an external source, such as a player or another device. The output is described as information generated by the game and outputted to an external source, but not used again as input to the game. Deep Reinforcement Learning algorithms differ from this approach, as the output of the algorithm is used again as input for a next step of the algorithm. As such, DRL is not used in games, but rather used to construct 'perfect' AIs for specific games. This means that the perfect AI for game A is not perfect for game B. This is why, even though the whole domain shares roughly the same structure, DRL is not applied frequently; the algorithm

needs to be tweaked differently for each and every game.
Outside of the domain, Deep Reinforcement Learning is applied even less frequently. While the games in the gaming domain share the same structure, consisting of an input, an output, and a set of rules, other domains lack this structure. When one wants to apply DRL to a domain, the domain first needs to be modeled into a similar structure as the gaming domain before DRL can even be applied to the domain. However, other domains can benefit greatly from the application of DRL principles. While the benefits won't be as large as the improvements made in the gaming domain, as shown in Figure 2, considerable improvements in terms of usability can be made with respect to the current situation by applying DRL principles into different domains. This leads to the following problem statement:

How can new domains be translated in such a way that Deep Reinforcement Learning principles can be applied more easily in order to further improve the usability of the applications in the new domain?

Here, usability is defined by the International Organization for Standardization (ISO) (2011) as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use".

1.3 Research Questions

Given the main problem statement as introduced in Section 1.2, a research question has been made, which is divided into multiple subquestions. This research question is as follows:

How can the principles of Deep Learning and Reinforcement Learning be applied to applications in new domains in order to improve the usability of these applications?

Q1: How can we determine whether a new domain would be suitable for the usage of Deep Reinforcement Learning?

Q2: How can an application in a new domain be transformed in such a way that it represents a game with rules, input and output?

Q3: How can the combined principles of Deep Learning and Reinforcement Learning improve the effectiveness of an existing application in a new domain?

Q4: How can the combined principles of Deep Learning and Reinforcement Learning improve the efficiency of an existing application in a new domain?

Q5: How can the results of implementing principles of Deep Learning and Reinforcement Learning be used to better determine whether a domain would be suitable enough for these kinds of implementations?

These question will be answered through a combination of literature research and a case study where an actual implementation of a DRL will be made. This will be done for an existing application in the patient-centric healthcare domain. Chapter 5 will give an explanation on the STRIPA, and its position and relevance in the patient-centric healthcare domain. In this research, the STRIP Assistant will be used. The case study will be carried out as follows: A DRL solution will be developed, after which it will be 'trained' existing rules and input. The generated output will then be compared with existing output of the STRIPA, to see whether there is a difference in effectiveness. Additionally, statements about the efficiency of the DRL solution will be made in order to give a view on how this compares to the efficiency of the existing implementation. After this has been done, the results of implementing DRL principles will be used in order to determine whether any changes can be made to the method to determine whether a domain would be well suitable for these kinds of implementations. This research will then result in several pieces of advice on how to improve the current STRIPA, as well as an artifact; a way to apply DRL principles in other domains.

2 Research Methodology

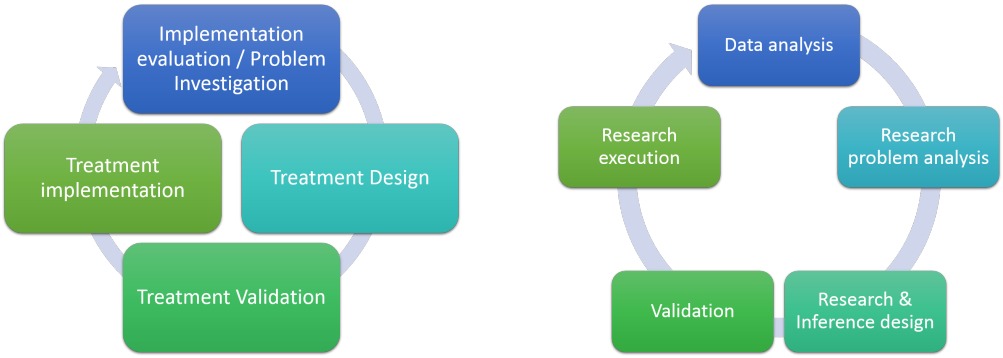
This section aims to describe the used research methodology. This thesis will use a combination of two research frameworks, namely the Design Science Research (DSR) framework (A. Hevner & Chatterjee, 2010), and the Knowledge Discovery Process (KDP), specifically CRISP-DM (Chapman et al., 2000). These two frameworks will be elaborated on in Section 2.1 and Section 2.2 respectively. Section 2.3 will then discuss how these two frameworks are combined to be used as the main research framework in this research. Section 2.4 will give an overview of the research plan of this thesis, and will show the relevant sub questions from Section 1.3 in relation to the research framework from Section 2.3.

2.1 Design Science Research

This section will introduce the Design Science Research framework, which will be one of the two frameworks that this research' framework will be based on, which will be discussed in Section 2.3.

The Design Science Research (DSR) paradigm "seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts" (A. R. Hevner, March, Park, & Ram, 2004). In IT, Design science is defined by Wieringa (2014) as "the design and investigation of artifacts in context" (Wieringa, 2014). Wieringa distinguishes between design problems and knowledge questions. Design problems call for a change in the real world and requires an analysis of stakeholder goals, whereas knowledge questions do not call for a change in the real world, but ask for knowledge about the world as it is now (Wieringa, 2014). Another notable difference is that while design problems do not have a definitive 'correct' answer, a knowledge question should have only one answer.

Design problems and knowledge questions both call for their own way of solving the proposed problem. As such, these two have their own cycle, of which the simplified versions are presented in Figure 3. Wieringa proposes the use of an engineering cycle (Figure 3a) for a design problem, and the use of an empirical cycle (Figure 3b) for knowledge questions. When handling a design problem, the treatment will not always be implemented. As such, the *Treatment Implementation* in the engineering cycle can often be left out. The resulting cycle is called a design cycle.



(a) Engineering Cycle (b) Empirical Cycle

Figure 3. Overview of the two different cycles in Design Science Research, as described by Wieringa (2014)

2.2 Knowledge Discovery Process

This section will introduce the Knowledge Discovery Process framework, which will be the other framework our research' framework will be based on. More specifically, this section will talk about CRoss Industry Standard Process for Data Mining (CRISP-DM).
The Knowledge Discovery Process (KDP), which is based on Knowledge Discovery for Databases (KDD), "creates the context for developing the tools needed to control the flood of data facing organizations that depend on ever-growing databases of business, manufacturing, scientific, and personal information" (Fayyad, Piatetsky-Shapiro, & Smyth, 1996). While a number of different models have been developed, CRoss Industry Standard Process for Data Mining (CRISP-DM) is widely considered to be the best KDP guideline to perform research in the Applied Data Science field with (Azevedo & Santos, 2008; Chapman et al., 2000; KS & Kamath, 2017; Mariscal, Marban, & Fernandez, 2010). The process is visualized in Figure 4.

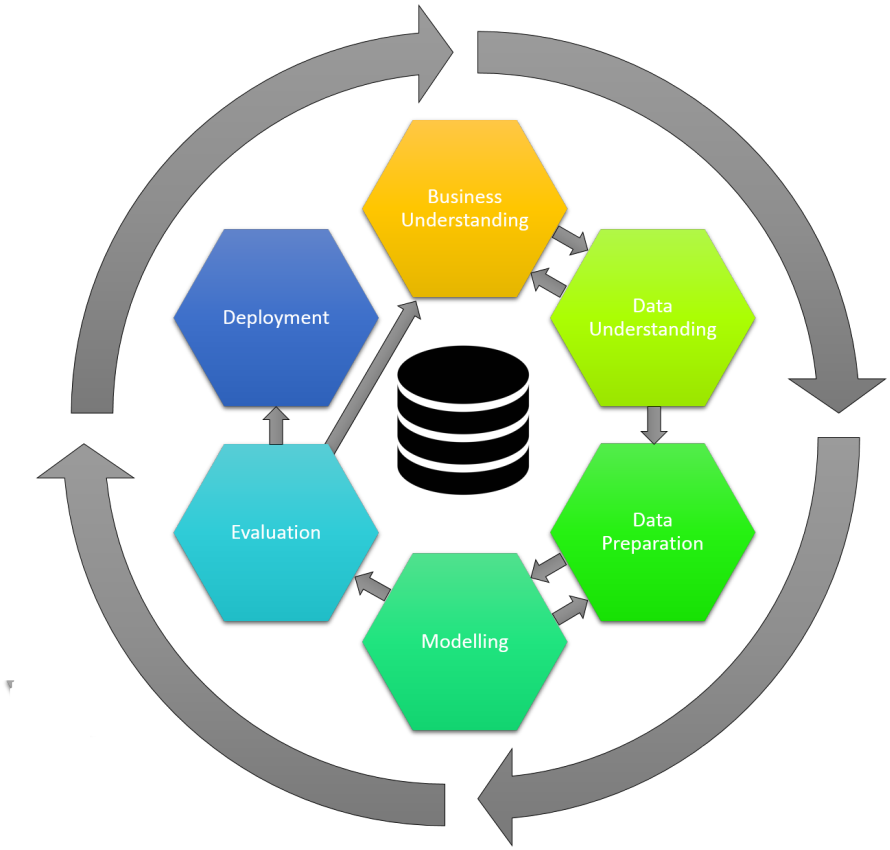


Figure 4. CRISP-DM, as described by Chapman et al. (2000)

CRISP-DM consists of six different stages, as described by Marbán, Mariscal, and Segovia (2009). *Business Understanding* focuses on determining the business objectives, and determining how data should be gathered. *Data Understanding* is about gathering, exploring, and describing the data that will be used to carry out the rest of the process. Here, the data quality is also tested. *Data Preparation* handles the actual selection of the relevant data. The data is cleaned, constructed, integrated and formatted. These steps will most likely be

repeated multiple times and not necessarily in a strict order. *Modeling* is where the data is taken, and modeling techniques are applied to the data set. *Evaluation* focuses on the results of the modeling step, and checks whether the business objectives are met with the modeled data. If not, the process is repeated again. Lastly, *Deployment* is the final step of CRISP-DM, in which the model is implemented.

2.3 Research Framework

This section will introduce the research framework used within this thesis, which is the framework as designed by Spruit and Lytras. This framework combines the principles of the DSR framework introduced in Section 2.1, and the principles of CRISP-DM, which is a KDP framework, introduced in Section 2.2.
In the context of this research, two different problems can be extracted. The first problem is a design problem, which corresponds with sub questions (Q1) and (Q2), which asks whether a domain is suitable for the application of Deep Reinforcement Learning techniques, and how an application in this domain can be transformed in order to incorporate DRL principles into the application. The second problem is more of a knowledge question, which corresponds with sub questions (Q3), (Q4), and (Q5) from Section 1.3, which asks if an implementation of DRL in an existing application can improve the usability of the application, and how results from this implementation can better help determine the suitability of new domains. As these problems differ from one another, they require a different solution, and thus a different process to solve the problems. The first problem is best suited as a design, as described by Wieringa (2014). As such, this problem requires a design cycle. The second problem is best solved by using CRISP-DM, as described by Chapman et al. (2000), because of the placement within the Applied Data Science field. To effectively solve both problems, a combination of both processes is required. As such, the research framework by Spruit and Lytras is used, which combines the principles of the design cycle of Wieringa with the principles of CRISP-DM by Chapman et al., and which can be found in Figure 5 (Spruit & Lytras, 2018). A design cycle is chosen over an engineering cycle, as the CRISP-DM cycle implements the treatment as devised in the design cycle. As such, the *Treatment Implementation* step from the engineering cycle is deemed obsolete.
In this figure, the right diagram represents the Applied Data Science, which was already presented in Figure 1. The left diagram represents the research framework by Spruit and Lytras (2018). In this framework, the six phases of CRISP-DM are represented by the outer cycle. Unlike the standard CRISP-DM cycle from Figure 4, the cycle is complete, and does not contain bidirectional connections anymore. The design cycle by Wieringa is tied to the colors in the cycle. The first two steps of the cycle, which are colored red, correspond to the first step of the design cycle. The yellow steps in Figure 5 correspond to the second step of the design cycle, and the blue steps from the framework coincide with the last step in the design cycle.
Within the context of this thesis, the *Problem Investigation* will be executed in Chapters 3, 4, and 5. More specifically, Chapters 3 and 4 focus on the technical domain aspects of the thesis, whereas Chapter 5 will focus on the patient-centric healthcare domain. While all three aforementioned chapters are part of the *Domain Understanding* phase from CRISP-DM, only Chapter 5 will discuss the *Data Understanding* phase. After the first phase has been finished, the *Treatment Design* will be discussed in Chapter 6 and Chapter 7.

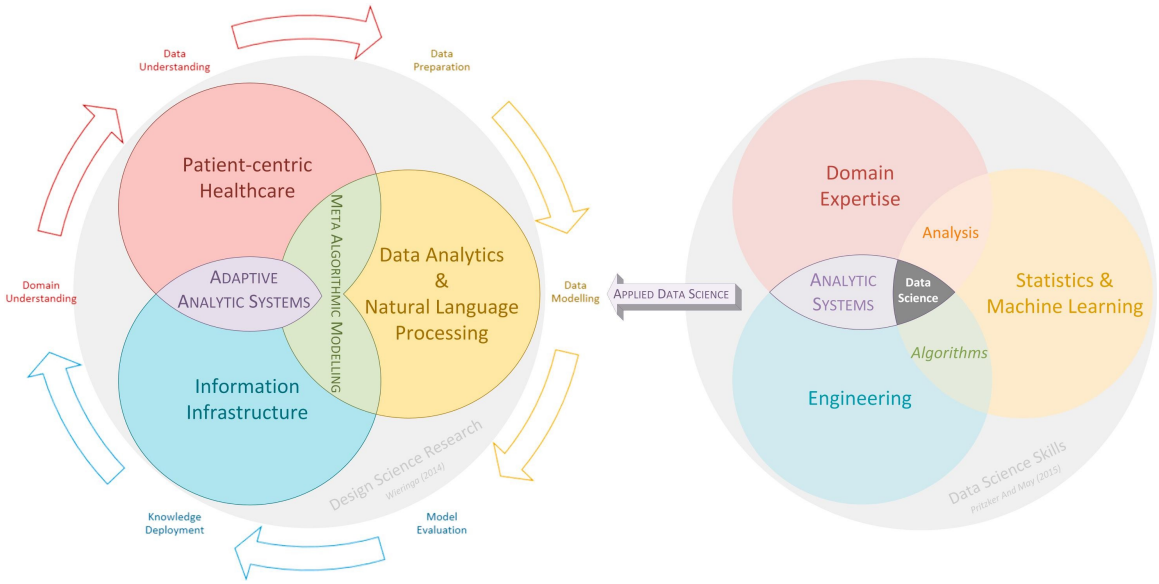


Figure 5. Design Science Framework used for this thesis, as described by Spruit and Lytras (2018), and its' relationship to Applied Data Science

This Chapter will cover both the *Data Preparation* and the *Data Modeling* steps from the CRISP-DM cycle. Lastly, Chapters 8, 10, and 9 will focus on the *Treatment Validation* phase. Within this phase, Chapter 8 will focus on the *Model Evaluation*, whereas Chapter 9 will be the *Knowledge Deployment* phase.

2.4 Research Plan

This section describes the proposed research plan for this thesis. The research will follow the research framework for applied data science in patient-centric healthcare, as introduced in Section 2.3 and as seen in Figure 5. This framework by Spruit and Lytras combines CRISP-DM and Design Science Methodology. This is highlighted in Figure 6. This figure shows the relationship between the phases of DSM and the phases of CRISP-DM in the left figure, whereas the right figure shows which parts of the research framework answer which sub questions.

As shown in Figure 6, the first phase of this research will focus on understanding the domain of the research, and the data that this research will work with. This will answer sub question (Q1), which asks whether a certain domain is suitable for the application of DRL principles.

The second phase of the research focuses on the design of a treatment. This means that in this research, the data in a domain will be prepared and transformed in such a way, that it can be easily worked with. In this research, this step also covers the existing application that will be transformed in such a way that DRL principles can be applied more easily. After this step, sub question (Q2) can be answered, which asks how a domain can be transformed in such a way that it resembles a game with rules, input and output.

Lastly, the third phase of this research covers the validation of the treatment. The proposed DRL application will be evaluated, and knowledge on how to implement these principles

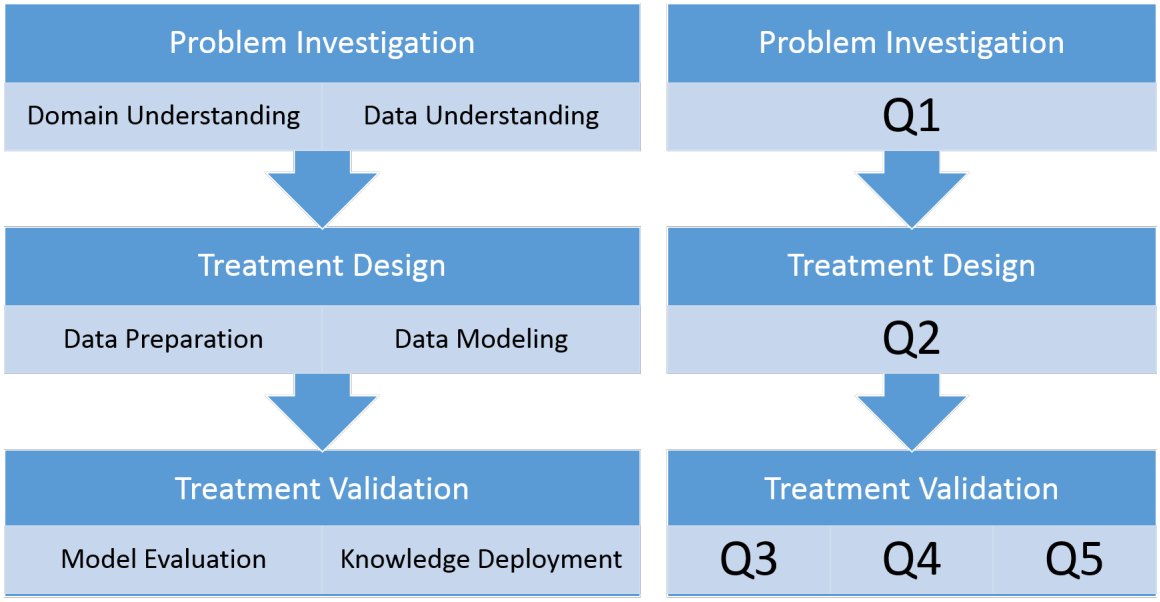


Figure 6. Research plan for this thesis

will be deployed. At the end of this phase, sub questions (Q3), (Q4), and (Q5) can be answered. Sub question (Q3) will be answered through a statistical difference making experiment, where the output of the proposed DRL application will be compared to the existing output. Sub question (Q4) will be answered as thorough as possible through theory. Lastly, the insights from implementing DRL principles will be used in order to answer sub question (Q5).

2.5 Literature Review Method

This section describes how the literature for this research was gathered. By showing how the literature for this research was gathered, the reproducibility of the research will increase. For this research, principles from the literature gathering method called *Snowballing* have been used. This method is described by Wohlin (2014) as "using the reference list of a paper or the citations to the paper to identify additional papers". However, where Wohlin uses a systematic literature review for his method, this thesis uses an informal exploration of literature in order to gather additional informational sources for this research.
The research makes use of both *Forward Snowballing*, where literature is gathered by looking at which papers cite the current paper, and *Backward Snowballing*, where literature is gathered by looking at which papers are cited by the current paper. Some of the starting researches are (X.-W. Chen & Lin, 2014; LeCun, Bengio, & Hinton, 2015; Meulendijk et al., 2015; Mnih et al., 2015; Silver & Hassabis, 2016; Silver, Schrittwieser, et al., 2017; Spruit & Lytras, 2018)
In addition to this, literature is gathered on a case-by-case basis whenever new topics are introduced within the thesis. This is literature that is not gathered by snowballing from one of the starting researches, but is still vital to the thesis.

3 Theoretical Background: Deep Reinforcement Learning Principles

This section will discuss the principles behind deep learning and reinforcement learning, and how these principles can be combined in order to create deep reinforcement learning. First, the principles of deep learning and reinforcement learning will be discussed separately. This will be done in Section 3.1 and Section 3.2 respectively. Afterwards, Section 3.3 will discuss how these two algorithms can be combined. Note that this section will not discuss the implementation of Deep Reinforcement Learning in the STRIP Assistant. This will be done in Chapter 7.

3.1 Deep Learning

Deep learning is a term which is heard and used often in business. It is defined by Deng and Yu as "a class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification". As stated in this definition, deep learning can be implemented in either a supervised, unsupervised, or semi-supervised way. Supervised learning is defined by Russell et al. as the act of learning a function that maps input into output, based on a training set of input-output pairs. Where supervised learning used a training set with a known structure, unsupervised learning does not use this. The goal of supervised learning is learning a function that describes the properties of the dataset (Hastie, Tibshirani, & Friedman, 2009). These two types of deep learning can also be combined into semi-supervised learning. Semi-supervised learning combines both labeled and unlabeled data in order to find a function that predicts future test data better than a function that predicts from the labeled training data alone (Zhu, 2011).
Figure 7 gives a general overview of the workings of the deep learning algorithm. As shown, the algorithm accepts input at the input layer, which is then converted, through the use of (multiple) hidden layers, to the output layer. While Figure 7 gives an example of a deep learning algorithm, in this case a Deep Feed Forward (DFF) network or Deep Neural Network (DNN), many different types of deep learning exist. Schmidhuber lists multiple popular deep learning algorithms in his paper. He states that LSTM trained by Collectionist Temporal Classification (CTC), and Feedforward GPU-based MPCNNs (GPU-MPCNNs) based on Convolutional Neural Networks (CNNs) with Max Pooling (MP) trained through Back-propagation (BP) are the two most popular, cutting edge versions of deep learning as of 2015. While this thesis will not go into detail on the specific inner workings of these algorithms, Section 3.1.1 will address some of the more popular types of deep learning, and go into detail on both the inner workings, as well as the advantages and disadvantages of using these types.
Historically, the concepts of Deep Learning are based on Artificial Neural Networks (ANNs), which is defined by Van Gerven as "a computational model which is loosely inspired by the human brain as it consists of an interconnected network of simple processing units (artificial neurons) that learns from experience by modifying its connections." Based on this is the Deep Neural Network (DNN), which is defined as an Artificial Neural Network with multiple hidden layers between the input layer and the output layer. Figure 7 is an example of this. The first mention of Deep Learning was published by Dechter in the machine learning community in 1986 (Dechter, 1986). However, the first working algorithm for a multi-

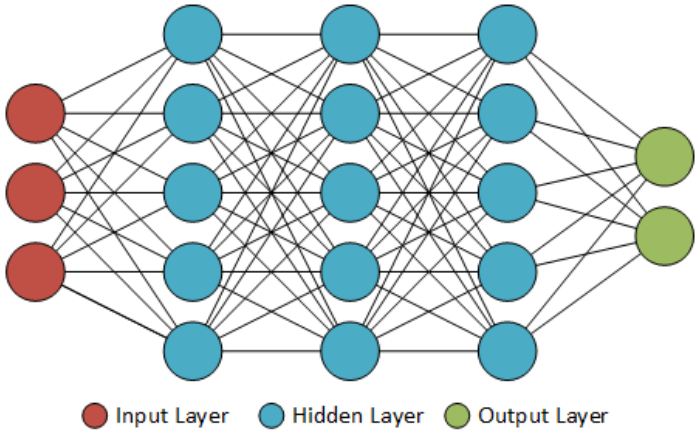


Figure 7. Diagram illustrating a basic version of a Deep Learning implementation

layered deep network in the field of deep learning was published in 1971 by Ivakhnenko (1971).

In the field of computer vision, Back-propagation is first applied in 1989 in order to help recognize handwritten postal codes (LeCun et al., 1989). Back-propagation (BP) is a method that is used to calculate the gradients used for the calculation of the weights used in a neural network (Goodfellow et al., 2016). However, while this algorithm was functional, training the algorithm required three days.

The next major improvement was made in 1992, when MP was first used in an application (Weng, Ahuja, & Huang, 1992). Max Pooling is the procedure of taking an $N \times N$ matrix, and converting it into a smaller $M \times M$ by means dividing the matrix into smaller sections and selecting the largest value as the new value (Graham, 2014).

In the field of speech recognition, most researched stopped using neural nets in favor of LSTM. Long Short-Term Memory networks are a variant of RNNs that are capable of storing long-term dependencies (Hochreiter & Schmidhuber, 1997).

However, in the regular industry, the impact of deep learning began in the early 00s. LeCun et al. states that CNNs processed around 10 to 20% of all the checks written in the USA at that time.

3.1.1 Types of Deep Learning.

As stated in Section 3.1, Deep Learning algorithms can be classified into three different types: supervised, semi-supervised, and unsupervised learning algorithms. However, a specific algorithm can be implemented or trained in either a supervised, and/or an unsupervised manner. For example, while a Neural Network are often classified as a supervised learning algorithm, a Deep Neural Network can be trained in both an unsupervised, as well as a supervised way. An example of this is the Artificial Neural Network (ANN).

3.1.2 Artificial Neural Network.

According to Zell (1994), ANNs are networks consisting of *neurons*, which can take input, and generate output. These neurons can also change their internal state, which can change the output of the neurons. By connecting the outputs of neurons to the inputs of other

neurons, a network can be created. The preferred connections between the neurons, as well as the neurons' triggers (or activation) to change their internal state, are governed by a learning rule.
A neuron N in layer k with label m and an input $p_m(t)$ received from a previous neuron n contains the following elements (Zell, 1994):

- An activation (or trigger) $a_m(t)$, which depends on a time parameter t.
- *Optional* A treshold value θ_m
- An activation function f that calculates the new activation at the next time step $t+1$, based on the current activation $a_m(t)$, the current input $p_m(t)$, and the treshold θ_m. As such, the activation for timestep $t+1$ is

$$a_m(t+1) = f(a_m(t), p_m(t), \theta_m) \tag{1}$$

- The output function f_{out}, for which the identity function is often used, calculates the output from the activation function

$$a_m(t) = f_{out}(a_m(t)) \tag{2}$$

- Each connection between a neuron n and neuron m is given a weight w_{nm}, which determines which path between layers is found best by the algorithm.
- A propagation function, which computes an input $p_m(t)$ to a neuron m from all of the connected neurons' outputs $o_n(t)$ from the previous layer:

$$p_m(t) = \sum_n o_n(t) w_{nm} \tag{3}$$

3.1.3 Deep Neural Network.

A Deep Neural Network (DNN) is defined as an Artificial Neural Network with multiple hidden layers (Bengio, 2009; Schmidhuber, 2015). As such, the structure is not much different from the structure of an ANN as described in the previous paragraph. The main addition to DNNs is the relation between the different layers, which is illustrated in Figure 8. A layer k, which computes a vector h^k of neurons, can compute the input for layer $k+1$ using the output of layer $k-1$ as its own input. The input layer of the network, which is defined as layer x in Figure 8 is specified as layer 0 with an output vector h^0. The output layer outputs a vector h^ℓ. To calculate the output vector of a layer h^k, Equation (4)

$$h^k = \tanh(b^k = W^k h^{k-1}) \tag{4}$$

is used. In this formula, a vector of offsets b^k, a matrix of connection weights W^k, and the output vector of the previous layer h^{k-1} are used to calculate the output vector of layer k. The tanh function is applied to each neuron in the layer, but can be replaced by other non-linear functions, such as the sigmoid function.

$$sigm(u) = \frac{1}{1+e^{-u}} \tag{5}$$

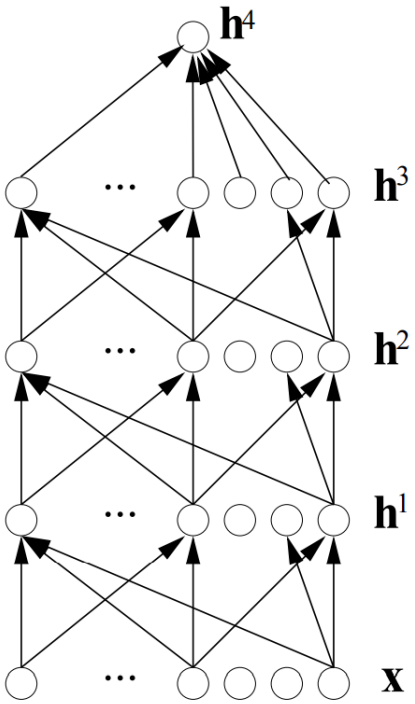


Figure 8. A technical representation of a Deep Neural Network, as defined by Bengio (2009)

The final output vector h^ℓ is used as a predictor, and can be combined with a supervised target y into a loss function

$$L(h^\ell, y) = b^\ell + W^\ell h^{\ell-1} \tag{6}$$

where the goal of the algorithm is to minimize the loss (in accuracy) for the output. For this, the negative conditional log-likelihood $L(h^\ell, y) = -\log P(Y = y|x) = -\log h^\ell_y$ is often used. The result of this calculation, a set of values over (x, y), should be as small as possible in order to achieve a more accurate neural network. The input for this calculation is obtained through the softmax function

$$h^\ell_i = \frac{e^{b^\ell_i + W^\ell_i h^{\ell-1}}}{\sum\limits_j e^{b^\ell_j + W^\ell_j h^{\ell-1}}} \tag{7}$$

where W^ℓ_i is the ith row of the matrix of weights for the output layer W^ℓ, $h^\ell_i > 0$ and $\sum_i h^\ell_i = 1$.

3.1.4 Recurrent Neural Network.

A Recurrent Neural Network (RNN) is a type of ANN where connections between its components form a directed (a)cyclic graph, and keeps an unlimited history which is represented by recurrently connected components (Mikolov et al., 2010; Rumelhart, Hinton, & Williams, 1986). A global overview of a RNN is shown in Figure 9.

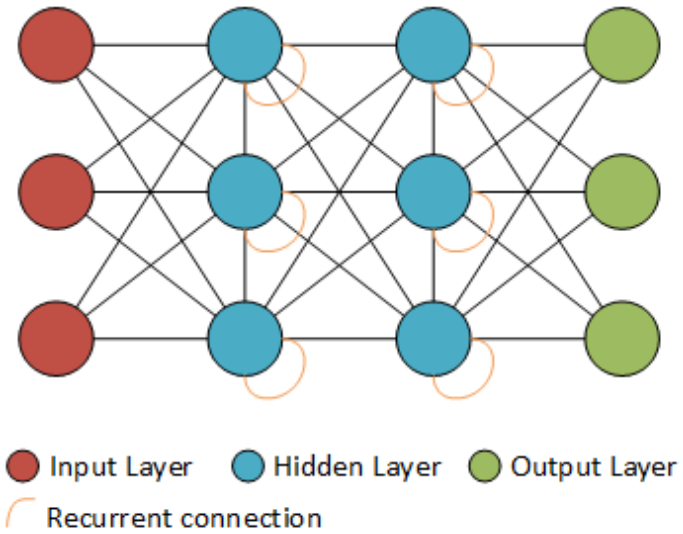


Figure 9. A global overview of a Recurrent Neural Network, as described by Mikolov et al. (2010)

As shown in this figure, a RNN functions similarly to a regular Feedforward Neural Network (FNN), the same category of Neural Networks as ANNs and DNNs. The main difference is the addition of a recurrent connection. This allows a RNN to form a short-term memory in order to deal with input invariances. This is something that regular FNNs are not able to achieve.
Like an ANN and a DNN, a RNN consists of an input layer x with input vector h^0, one or more hidden layers s with input vectors h^1 to h^i, and an output layer y with input vector h^k and output vector h^ℓ. As described by Mikolov et al. (2010), the input, hidden, and output layers can be calculated using Equations (8), (9), and (10)

$$x(t) = w(t) + s(t-1) \tag{8}$$

$$s_j(t) = f(\sum_i x_i(t) u_{ji}) \tag{9}$$

$$y_k(t) = g(\sum_j s_j(t) v_{kj}) \tag{10}$$

with $f(z)$ being the activation function, which is analog to Equation (5), and $g(z)$ being the softmax function

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{11}$$

Another thing that is possible within a RNN, is the ability to implement recursion. As such, the history represented by the network can be infinite, only bound by PC processing power. In order to 'revert' the recurrent network from a directed cyclic graph into a acyclic graph, it has to be unfolded. Figure 10 gives an overview of the unfolding process.

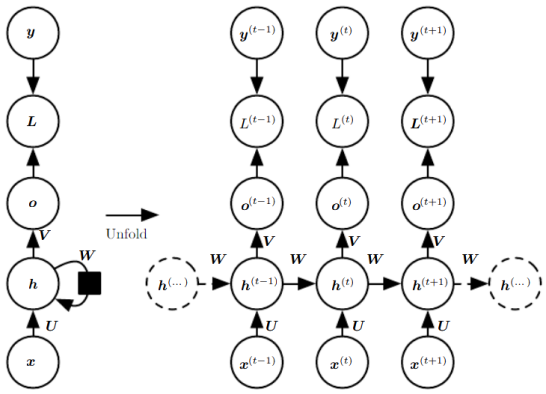


Figure 10. A technical breakdown of the unfolding process of a Recurrent Neural Network, as described by Goodfellow et al. (2016)

In this figure, x is the input vector, which is passed to the hidden layer which has a vector h, after which it is outputted as an output vector o. This value is used as an input to a loss function L, which compares $\hat{y} = softmax(o)$ to a target value y. U, V, W are all weight matrices.

A special kind of RNN is a Long Short-Term Memory (LSTM). A LSTM is a variant of RNNs that is capable of storing long-term dependencies. Its structure is similar to the structure of a RNN, shown in Figure 9. The main difference is the addition of gates and a memory cell. This allows the networks to store information within itself long-term in a memory cell, as well as forget this information through a forget gate. The rest of the gates allow the network to 'shut down' itself, by blocking or delaying signals that pass through a neuron. Figure 11 shows an overview the inner workings of a neuron, with all of the connections between the elements of a neuron, and its corresponding gates.

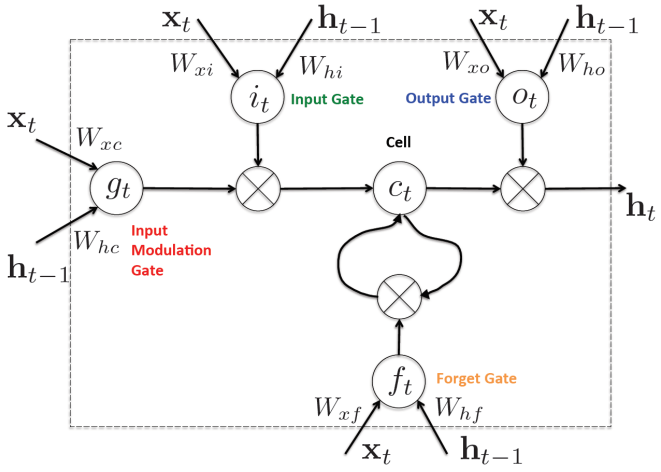


Figure 11. Overview of a LSTM network neuron (G. Chen, 2016)

In this figure, a similar structure is shown to the left part of Figure 9. The main additions are the gates, which have their own formulas, as described by Goodfellow et al. (2016)

and Zaremba, Sutskever, and Vinyals (2014). In these equations, W_q and U_q represent the weights matrices of the input and the recurrent connections. q represents the input gate i, output gate o, forget gate f, or memory cell c, depending on what is being calculated.

$$\Gamma_q = W_q x_t + U_q h_{t-1} + b_q \tag{12}$$

$$f_t = \sigma_g(\Gamma_f) \tag{13}$$

$$i_t = \sigma_g(\Gamma_i) \tag{14}$$

$$o_t = \sigma_g(\Gamma_o) \tag{15}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(\Gamma_c) \tag{16}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{17}$$

In these equations, the $\circ$-operator is used as an entry-wise product. h_t represents the output vector of the LSTM network. σ_g represents a sigmoid function, analog to Equation (5). σ_c and σ_h represent hyperbolic tangent functions. Lastly, b represents a bias vector.

3.2 Reinforcement Learning

In contrast to Deep Learning, Reinforcement Learning is not often heard in the media or in business. However, (Deep) Reinforcement Learning is believed to be one of the top 10 emerging trends in the field of AI, according to Rao, Voyles, and Ramchandani (2018). Reinforcement Learning (RL) is defined by van Otterlo and Wiering (2012) as "a general class of algorithms in the field of machine learning that aims at allowing an agent to learn how to behave in an environment, where the only feedback consists of a scalar reward signal". As stated by Sutton, Barto, and Bach (1998), RL differs from supervised learning, as supervised learning is learning by examples that are provided by an external source, whereas RL is learning by the own experience of the agent. However, as stated by Hilleli and El-Yaniv (2018), RL has its flaws: "it either requires a realistic simulation of the agent's interaction with the environment or requires operating the agent in a real-world environment, which can be quite costly". Attempts have been made to implement RL into applications in the medical domain by (Doorhof, 2018), but these have not been completely successful.

The field of RL is relatively young. It is also known as Neuro-Dynamic Programming (NDP), which is defined by Bertsekas (2008) as "a class of dynamic programming methods for control and sequential decision making under uncertainty". However, the first mention of an algorithm that resembles RL was in 1959, when Samuel used machine learning in order to let the computer play checkers. The algorithm was only provided with the game rules, and a set of parameters, after which it took 8 to 10 hours of learning to surpass the skill level of the person who wrote the algorithm Samuel (1959).

Figure 12 gives a schematic overview of a Reinforcement Learning algorithm. It contains an *Agent*, which works in an *Environment*. Each tick of the algorithm, the *Agent* performs an *Action* in the *Environment*. The result of this action is a tuple, containing an *Observation* of the current state of the environment, as well as a *Reward*, which can be either positive or negative.

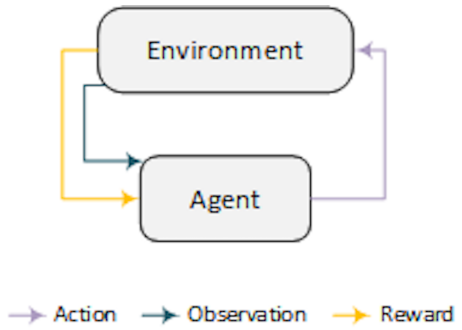


Figure 12. Diagram illustrating a basic Reinforcement Learning implementation

In essence, the elements of a RL algorithm make up a MDP. A MDP is defined as "a model for sequential decision making when outcomes are uncertain" (Puterman, 2014). It consists of decision epochs, states, actions, rewards, and transition probabilities. Figure 13 shows an example of a MDP (Matiisen, 2015).

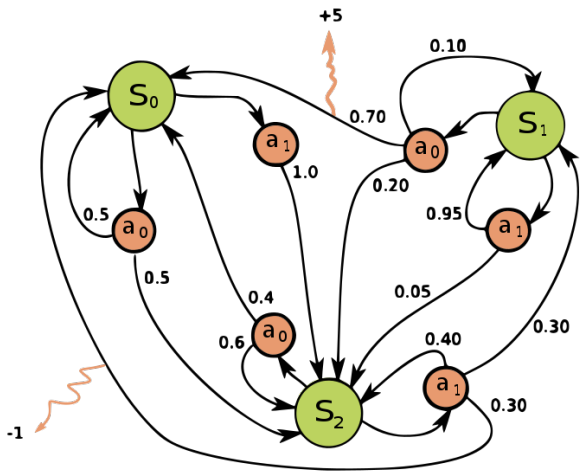


Figure 13. Example of a Markov Decision Process (MDP). The orange arrows represent the rewards, the orange circles represent the actions, and the green circles represent the states

The following sections will go into detail on the various different types of RL algorithms. As stated, RL differs from supervised learning. For more information on supervised learning, refer to Section 3.1. These sections will cover two of the more popular variants of RL.

3.2.1 Q-Learning.

Quality-Learning (Q-Learning) is one of the algorithms used in AlphaGo. Like all other forms of RL, it is a MDP. As such, it follows the same general structure as shown in Figure 12. The Q in Q-Learning stands for quality, as the function used in the algorithm, the *Q-function*, describes the quality of the state the algorithm currently is in (Matiisen, 2015). Melo (2001) proved that for every finite MDP, Q-Learning will find an optimal policy

that maximizes the expected value of the total reward.
Q-learning functions by first summing all of the future rewards of the algorithm, which is calculated in Equation (18).

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \ldots)) = r_t + \gamma R_{t+1} \tag{18}$$

Here, R_t is the total future reward from time step t, r_t is the reward at time step t after performing action a_t. However, not all rewards should be counted equally. The more in to the future a rewards is (and thus, the more time steps it is away), the more the reward may differ, and the less it should be counted. As such, a factor γ is used, which is a value between 0 and 1. A γ of 0 will result in an algorithm that only looks at immediate rewards, whereas a γ of 1 will result in a deterministic environment in which the same actions will always lead to the same answers.
Q-Learning is then defined by Equations (19), (20), and (21).

$$Q(s_t, a_t) = \max_{\pi} R_{t+1} \tag{19}$$

$$\pi(s_t) = argmax_{\alpha_t} Q(s_t, a_t) \tag{20}$$

Equation (19) represents the so-called quality function. $Q(s_t, a_t)$ represents the quality of the current state s_t after performing action a_t, and R_{t+1} represents the future reward as explained in Equation (18). The π represents the action policy, which is a rule that determines which action should be chosen in what state. Equation (20) further explains the Q-Learning policy. In this case, the policy is to just simply choose the action with the highest result. Combining these two equations results in Equation (21)

$$Q(s_t, a_t) = r_t + \gamma \max_{\alpha_{t+1}} Q(s_{t+1}, a_t) \tag{21}$$

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{22}$$

In this equation, it is shown that the maximum future reward of a state s_t after performing action a_t is the immediate reward r_t plus the maximum future reward of state s_{t+1}, weighed by a factor γ. Equation (21) is also known as the Bellman equation. Often times, this equation is written in the form as shown in Equation (22).

3.2.2 SARSA.

Another popular RL algorithm is State-Action-Reward-State-Action (SARSA), which was proposed by Rummery and Niranjan (1994) under the name "Modified Connectionist Q-Learning (MCQ-L)". SARSA is defined as an on-policy learning algorithm used for learning Markov Decision Process policies. It shows a large resemblance to Q-Learning, with the exception of one detail, which is shown in Equation (23):

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left[r_t + \gamma Q(s_{t+1}, \mathbf{a_{t+1}}) - Q(s_t, a_t)\right] \tag{23}$$

While Q-Learning maximizes the reward by assuming it will perform action a in state s_{t+1}, SARSA takes the action a_{t+1} into account as well. This is highlighted in bold in Equation (23).

3.3 Deep Reinforcement Learning

While both DL and RL have been applied separately from each other, the combined usage of these algorithms is not widely used. The combination of these two types of algorithms is called Deep Reinforcement Learning (DRL). A proposed definition for DRL is as follows:

> *Deep Reinforcement Learning is a type of machine learning algorithm in which a deep learning algorithm represents the internal agent of a reinforcement learning implementation as a way to represent its' policy function or notion of expected rewards.*

DRL was first mentioned by Shibata and Okabe (1997), albeit not under the name Deep Reinforcement Learning. This work describes how the performance of a RL implementation improved considerably when guided by a RNN. DRL was later popularized by Mnih et al. (2013), in which they described how DRL vastly improved over all previous approaches on previous games, and beats a human expert in three different games. Later, this work was generalized to more ATARI games (Mnih et al., 2015). More recently, AlphaGo Zero, an AI developed by Google DeepMind, was able to master the game of Go without any previous knowledge about the game (Silver, Schrittwieser, et al., 2017). It achieved this by using DRL in combination with a MCST implementation.

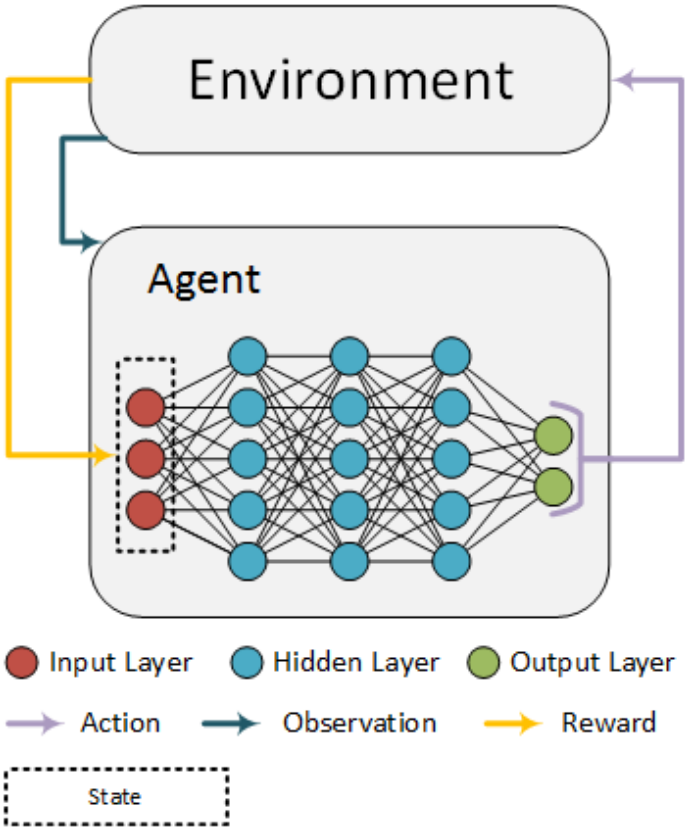


Figure 14. Diagram illustrating a basic Deep Reinforcement Learning implementation

Figure 14 shows a global overview of a DRL implementation. Like Figure 12, the implementation consists of an *Agent*, which performs an *Action* in an *Environment*, for which it obtains a *Reward*. An *Observation* of the outcome in the Environment can be made by the Agent. The main difference from Figure 12 is that the agent is guided by a Deep Neural Network, which is nested in the Agent. This is the DNN from Figure 7. The Action performed by the Agent is based on the output layer of the DNN, whereas the Reward is used in order to alter the input layer of the network.
A Deep Q-Network (DQN) is a type of network developed by Mnih et al., which combines Reinforcement Learning with a Deep Neural Network. More specifically, it uses Q-Learning as its' RL algorithm. This approach combines the strengths of DL and RL approaches, and complements each others' weaknesses. It allows a DL algorithm to learn and improve itself, whereas it allows a RL algorithm to approximate the optimal action-value function. The implementation by Mnih et al. is shown in Equation (24).

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \ | s_t = s, a_t = a, \pi] \tag{24}$$

This formula gives the maximum sum of rewards r_t discounted by γ at each timestep t. This is achieved by a policy $\pi = P(a|s)$, after making observation s and taking action a in response.
There is one flaw in this approach, which is explained by Van Hasselt, Guez, and Silver (2016). In this paper, it is highlighted that the value function and network weights (θ_t) used to select an action are also used to evaluate that action. This results in an overoptimistic value estimation. In order to prevent this, a second value function can be used, which results in a second set of network weights (θ'_t). This principle is called a Double Deep Q-Network (DQN). The main difference between a regular DQN and a Double DQN is shown in Equation (25) and Equation (26) respectively. Equation (25) functions the same as Equation (24), but uses the notation from the paper of Van Hasselt et al. (2016), which decouples the selection and evaluation into separate components. Note the difference between the used network weights in the value function.

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{argmax}\ Q(S_{t+1}, a; \theta_t); \theta_t) \tag{25}$$

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{argmax}\ Q(S_{t+1}, a; \theta_t); \theta'_t) \tag{26}$$

4 Determining the Suitability of New Domains

This section will discuss how a suitable domain for the application of DRL principles can be found. An informal exploration of literature regarding challenges of Deep Reinforcement Learning has been performed. The main challenges that are presented from the literature are presented in Section 4.1. These challenges will be used as a way to check whether a domain would be suitable for the implementation of DRL principles into an application in the domain. Section 4.2 will introduce the 4VATT Suitability model, which consists of a set of questions to aid in estimating the suitability of a new domain.

4.1 Challenges of Deep Reinforcement Learning

As stated in Chapter 1, Deep Reinforcement Learning (DRL) is mostly applied in the gaming domain. This is done to either beat complex board games (Silver, Hubert, et al., 2017; Silver, Schrittwieser, et al., 2017), 2D videogames (Mnih et al., 2013, 2015), or even older 3D videogames (Lample & Chaplot, 2017). However, outside of this domain, the combined principles of DL and RL have not been applied frequently. There are some challenges for DRL in order to be applied more widespread, which have been identified through an informal literature exploration on the challenges of implementing DRL principles into new domains. These challenges are as follows:

- Factors corresponding to Big Data's 4V model
  - Volume
  - Variety
  - Velocity
  - Veracity
- Accountability of decisions made by implementation
- Domain Transparency
- Domain Transformability

The next subsections will go more into the different factors of, and challenges for a successful DRL implementation. These challenges are based on work by (Berman, 2013; X.-W. Chen & Lin, 2014; IBM, 2018; Miotto, Wang, Wang, Jiang, & Dudley, 2017; Valk, 2018)

4.1.1 Big Data's 4V model.

One of the most important factors of a successful DRL is the data that the implementation is built on. In order to make a good representation of the data, the data itself must be of high quality and quantity, as stated by Najafabadi et al. (2015). The amounts of data needed in order to build a large scale, high quality implementation of DRL is often classified as 'Big Data'. Big Data in the context of this thesis is associated with 4 key concepts, which are *Volume*, *Variety*, *Velocity*, and *Veracity*. These 4 concepts are also known as the *'4 V's of Big Data'*, or the *'4V model'*, which is visualised in Figure 15 (IBM, 2018).

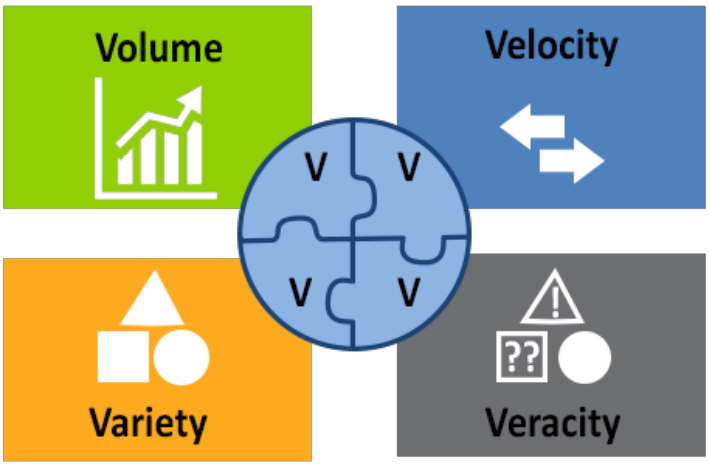


Figure 15. Diagram showing the 4V model, containing the four different components of Big Data

In order to make an accurate representation of the data, enough training data is needed. As such, the *Volume* of the data within the domain is very important. According to Goodfellow et al. (2016), at least 50000 training examples are needed to get 'acceptable' results out of a DL Neural Network implementation, whereas 10 million training examples are needed in order to achieve human-like results. However, while a successful DRL is dependent on a high volume of data, the amount of data stored by Google, Microsoft, Yahoo! and other large internet companies exceeds the data need for an implementation like this. According to Council (2013), the stored data of these companies is measured in exabytes (10^{18} bytes). Another factor is the *Variety* of the data. While DRL implementations are suited for handling different kinds of data, this works best when the algorithm is presented with large amounts of unstructured data. In this case, the algorithm can detect higher level complexities and patterns in the data set, where other more shallow learning hierarchies fail to do so (Najafabadi et al., 2015). For data that is semi-structured, fully structured, or has a lack of variety, a DRL implementation might not outperform other machine learning algorithms. However, this is not solely dependent on the variety, but also on other characteristics of the data.

The *Velocity* of the data is also an important factor. It is described by Berman (2013) as data that is "constantly changing through the absorption of complementary data collections, through the introduction of previously archived data or legacy collections, and from streamed data arriving from multiple sources." One of the challenges is to adapt a DRL implementation to the changing speed of input, and to adapt it to streaming data.

Lastly, one of the aspects of Big Data deals with Data *Veracity*. Veracity, which was mentioned by IBM as the *fourth V*, represents the unreliability of data (Gandomi & Haider, 2015), along with the confidentiality, integrity, and availability of data (Kepner et al., 2014). This means that the data, data analytics, and outcomes should be error-free and credible (Raghupathi & Raghupathi, 2014). According to Raghupathi and Raghupathi (2014), this is especially critical in the healthcare domain, as "life or death decisions depend on hav-

ing the accurate information, and the quality of healthcare data, especially unstructured data, is highly variable and all too often incorrect."
While some characteristics to the 4V model have been added over time, such as the *Value* characteristic, these will not be considered in this thesis. As such, only the four characteristics described in this section will be taken into consideration in the context of this thesis.

4.1.2 Implementation and Accountability.

When implementing a DRL solution, one will deal with a lot of (Big) data. And with data comes a responsibility. After all, data should be kept safe and secure. According to Gantz and Reinsel (2012), 68% of the information is created or consumed by consumers, yet enterprises have liability or responsibility for nearly 80% of the data. This imbalance can create lots of privacy and accountability issues. Nature (2007) states that "any data on human subjects inevitably raise privacy issues, and the real risks of abuse of such data are difficult to quantify".
Even when researches try to be cautious and thorough about following procedures, the (intended) outcome from their research can sometimes be harmful. Gantz and Reinsel gives the example of findings made by Emmens and Phippen (2010). In their research, they found that there was a correlation between self-harm and suicide. In order to combat this, they designed and tested an educational intervention to help prevent self-harm, and thus to help prevent suicide. However, the research had an adverse effect and increased suicidal thoughts under teenagers, after which the intervention was rolled back.
This is one of the examples in which accountability plays a large role. The dilemma with implementations such as these, is who is to be held accountable when something goes wrong. Should it be the people who gathered the data, the ones who made the models, the ones implementing the solution, or should someone higher up in an organization be held accountable for any mistakes? The more possible issues an implementation could bring up, the less suitable a domain will be for such a 'black box' style implementation.

4.1.3 Domain Transparency.

Another factor which plays a role in determining the suitability of a new domain, is the amount of transparency needed in a domain. This is slightly related to the issues highlighted in Section 4.1.2. Because DRL is seen as a 'black box' algorithm, the decision making process of the algorithm is either not known to, or not understandable for the user.
While the decision making process is not important in some domains, an example being the gaming domain, transparency is very important in other domains. For example, the deeper decision making process is not very important in the gaming domain. For example, SethBling (2015) created a neural network capable of playing Super Mario World without any prior knowledge of the game controls and mechanics, similarly to how AlphaGo Zero functions. In this case, the inner decision making process was not important. This implementation is shown in Figure 16
However, as stated in Section 4.1.1 by Raghupathi and Raghupathi, the healthcare domain deals with decisions about life our death, which depend on the correct information and results. If a wrongful decision is made, it is critical to know why a certain decision was made, in order to prevent this same decision to be made in the future.

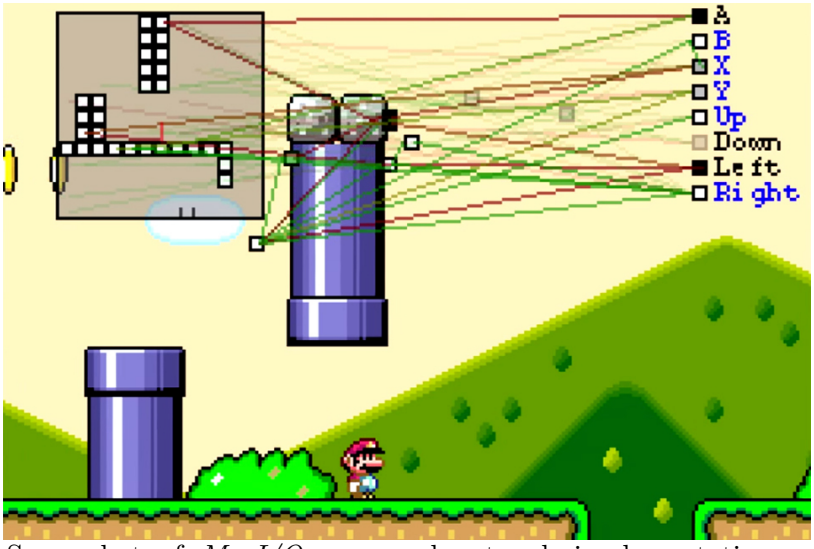


Figure 16. Screenshot of *MarI/O*, a neural network implementation capable of autonomously learning and playing *Super Mario World*

In light of the recent introduction of the GDPR, this currently is a large issue, as described by Goodman and Flaxman (2016). Persons that are affected by decisions made by an AI, now have a right to know on what grounds this decision is made. This poses a challenge for Deep (Reinforcement) Learning, which is commonly described as a 'black box'. Furthermore, Samek, Wiegand, and Müller (2017) states that it is a necessity for certain domains to have insight in the decision making process of the AI, as small (, fixable) errors could have a large outcome on the decision.
This issue is being solved from multiple perspectives. Explainable AI (XAI) aims to solve this by 'opening the black box' and giving insight in the decision process of the AI. This has been explored by Valk (2018), who researched an interpretable recurrent neural network for heart failure re-hospitalization predictions. Gunning (2017) claims that there is a need for Explainable AI, stating that it "will be essential if users are to understand, appropriately trust, and effectively manage this incoming generation of artificially intelligent partners". However,while Bau, Zhou, Khosla, Oliva, and Torralba (2017) proposes a framework to quantify interpretability of CNNs, the interpretability is not an axis-independent phenomenon. This means that the framework does not work the same on every layer of the neural network hidden layer, and that each hidden layer of the network disentangles different abstraction levels of information. This has a significant effect on the representation learned by different hidden levels.

4.1.4 Domain Transformability.

As stated before in Section 1.2, all of the previous examples share the same structure; *"a piece of software that transforms input into output, according to a set of rules"*. As such, not all applications within a domain are suitable for the application of DRL principles. Thus, the ability to *transform* (an application within) a domain to a more structured, game-like form, will aid in the process of implementing DRL.
This process consists of rewriting inputs in order to be of a certain structure, determining

a reward function for the DRL algorithm, as well as the environment for the RL part of the algorithm. Additionally, RL requires the application to be some kind of simulation, with a time element involved in it (Hilleli & El-Yaniv, 2018). Thus, if the current application does not have this structure, more transformations are needed.

4.2 Selecting a Suitable Domain

Based on the challenges and factors proposed in Section 4.1, this section will propose a model that can aid the user in determining how suitable a domain is for the implementation of DRL principles. This model, which will be called the *4VATT* Suitability Model, consists of a set of questions, divided over 7 different categories. The outcome of the model is a suitability score, which gives an estimation of the suitability of the domain.
The core part of the 4VATT Suitability Model is the set of questions. The complete list of questions can be found in Appendix A. Each question belongs to a factor or challenge mentioned in Section 4.1. The answers to those questions are in the form of a five-point Likert scale. For example, one of the questions on the topic of *Transparency* is as follows: *"What amount of governance of the application in this domain is needed or will be used?"*, where the answers range from "Very high" to "Very low".
In order to calculate the suitability of a domain, all of the questionnaire answers have to be transformed into a score. For some answers, it is desirable to have an inverted score. For example, it is desirable to give a high score when the amount of work needed to transform an application into one with a game-like structure is very low.
Once all of the answers have been converted, the scores of all of questions have to be added up, and be divided by the maximum score in order to get a suitability percentage. This is expressed by Equations (27) and (28), where Q represents the set of questions, and F represents the set of factors.

$$\upsilon = \sum_{f \in F} \sum_{q_f \in Q_f} score_{q_f} \tag{27}$$

$$suitability = \frac{\upsilon}{\max(\upsilon)} * 100\% \tag{28}$$

A similar percentage can also be calculated for each individual factor. This is done by summing the scores of a single factor, and dividing by the maximum score for that factor. The result is a suitability score, a percentage which gives a rough indication of the suitability for that factor. An example of the (random) suitability scores for two domains is shown in Figure 17. In this example, both example domains have a suitability score of **75.6%**. However, the score distribution between the different factors is not the same. This is shown in Table 1.
In Section 1.3, multiple research questions were posed. More specifically, (Q1) asked *"How can we determine whether a new domain would be suitable for the usage of Deep Reinforcement Learning?"*. With the 4VATT Suitability Model, a method is created which aids in determining whether a domain is suitable for the implementation of DRL principles. By receiving the aid of domain experts, or possible stakeholders with sufficient domain knowledge, in order to answer the questions of the model, a rough indication of the domain suitability can be given.

Table 1
Score distributions for the domains illustrated in Figure 17, showing that different distributions can lead to an equal suitability score.

Category	Domain A		Domain B	
	Score	Suitability	Score	Suitability
Volume	7	70.00%	9	90.00%
Variety	5	50.00%	10	100%
Velocity	6	60.00%	10	100%
Veracity	9	90.00%	10	66.67%
Accountability	14	93.33%	10	66.67%
Transparency	14	93.33%	10	66.67%
Transformability	13	86.67%	9	60.00%
Total	68	75.56%	68	75.56%

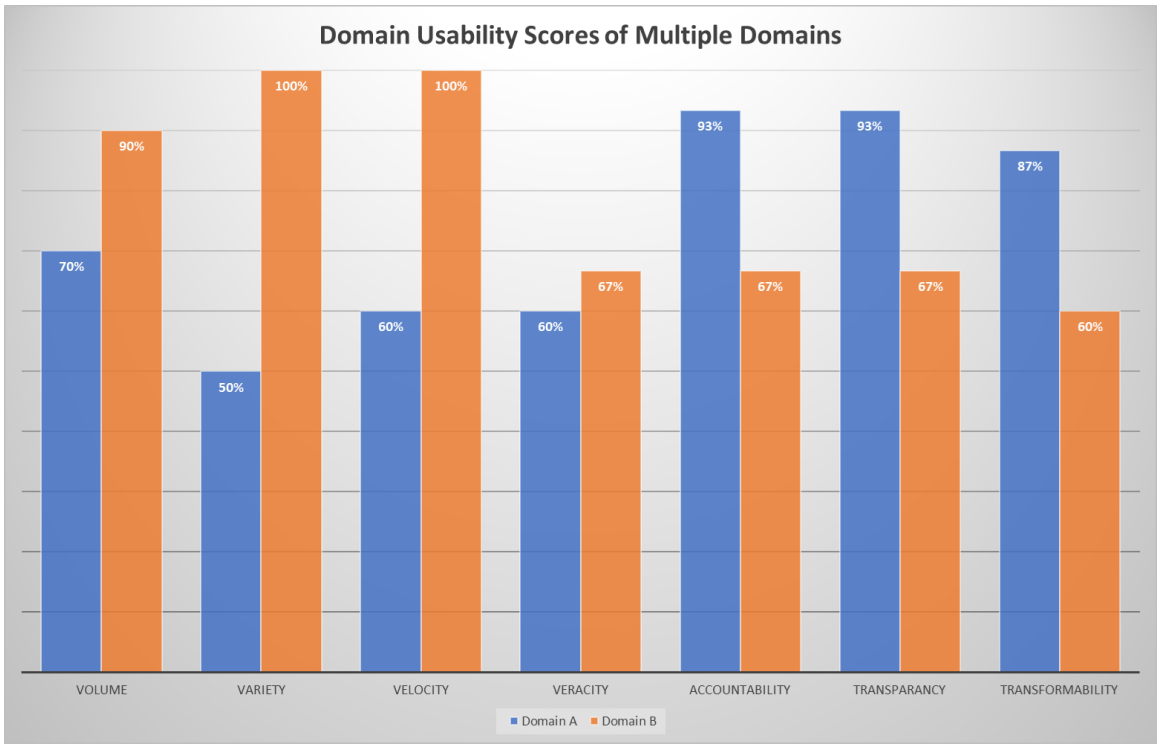


Figure 17. An example of a visualization of the suitability scores per factor for two different domains

While the success of a successful implementation of DRL principles in an existing application depends on much more factors than the ones proposed in the 4VATT Suitability Model, these are factors that are not challenges or aspects of the domain at hand. For example, a success factor of an implementation could be the financial aspect of the implementation, or whether there is enough computing power to carry out the calculations of the algorithm. This, however, falls outside of the scope of the 4VATT Suitability Model.

5 Empirical Background: STRIP Assistant and the Medical Domain

This section will introduce the medical domain and the STRIP Assistant, an application within this domain. As stated within Chapter 1, Deep Reinforcement Learning principles are not applied frequently within the medical domain yet. However, while applying these principles could have a positive influence on the usability of applications within the domain, it is not yet known whether the domain is really suitable for the implementation of DRL principles.
Section 5.1 will see the 4VATT Suitability Model applied to the medical domain, in order to determine whether this domain would be suitable for the application of DRL principles. In particular, the scores on *Transformability* are important, as these will tie into both Chapter 7, as well as research question (Q2).
Section 5.3 will give an introduction into the STRIP Assistant in the context of the medical domain. The section will discuss its place within the domain, will go into detail on the inner workings of the STRIPA, and how it helps to solve the problem of *polypharmacy*.

5.1 Applying the 4VATT Suitability Model

This section will discuss the suitability of the medical domain for the application of DRL principles. More specifically, the prescriptive healthcare domain will be looked into, in which the STRIP Assistant is positioned. This positioning will be discussed further in Section 5.3. In order to determine the suitability of the prescriptive healthcare domain, the 4VATT Suitability Model will be used. The outcome of the model will determine whether or not DRL principles should be implemented into the STRIP Assistant. The answers to the 4VATT Question List are based on multiple sources on the STRIP Assistant itself, as well as own experience from the medical domain as gathered during the development of *Valeas* (Knemeijer, 2018). Appendix B shows the given answers to all of the questions of the 4VATT Question List, complete with an explanation on how the answer is conceived.

Table 2
Score distributions for the suitability of the prescriptive healthcare domain

Category	Prescriptive Healthcare	
	Score	Suitability
Volume	7	70.00%
Variety	8	80.00%
Velocity	6	60.00%
Veracity	11	73.33%
Accountability	9	60.00%
Transparency	8	53.33%
Transformability	10	66.67%
Total	59	65.56%

Table 2 gives a detailed view of the Domain Suitability Scores per factor, as a result of the answered 4VATT Question List. These results are visualized in Figure 18. The 4VATT Suitability Model shows that the prescriptive healthcare domain has an expected suitability of approximately **66%**. This is mainly due to a relatively low score on the *Transparency* factor, which is compensated by a relatively high score on the *Variety* factor.

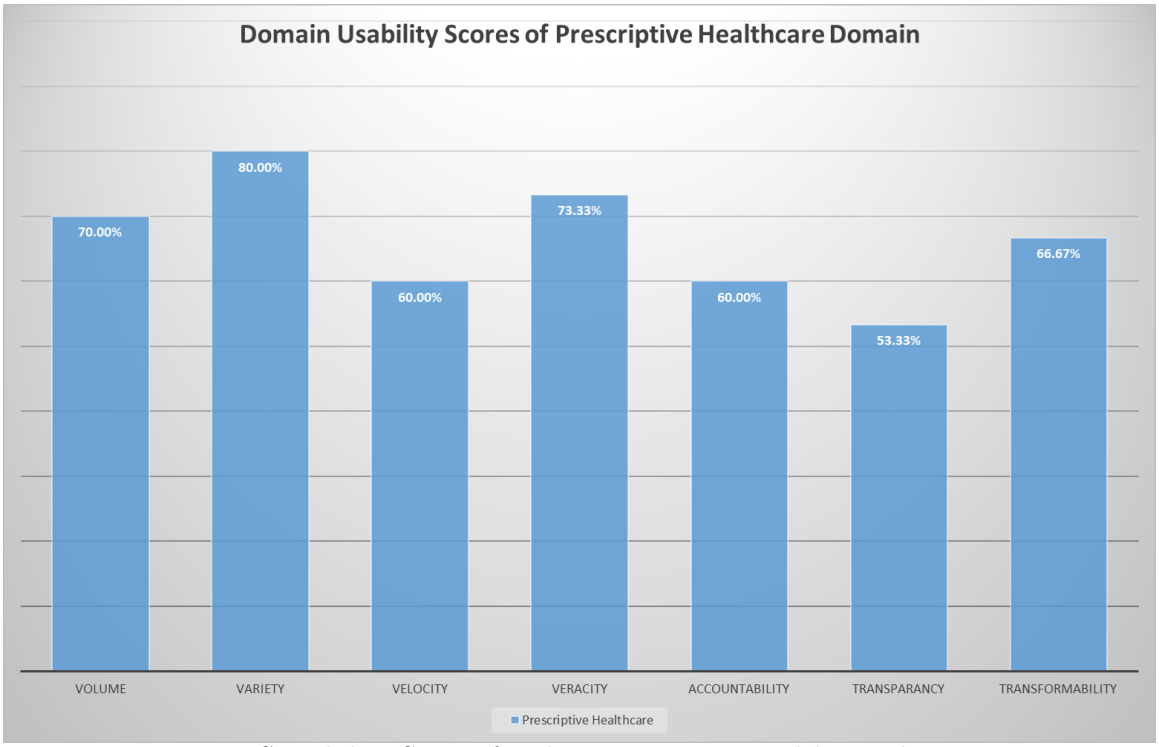


Figure 18. Domain Suitability Scores for the Prescriptive Healthcare domain

While a percentage of 66% might not be considered to be very high, a slight improvement in the *Transparency* factor will significantly improve the overall expected suitability of the domain. As such, there is reason to believe the prescriptive healthcare domain would be suitable for the implementation of DRL principles. The principles will be implemented in the STRIP Assistant, which will be introduced in Section 5.3. A more detailed overview of the implemented principles in the STRIP Assistant will be shown in Chapter 7.

5.2 The Prescriptive Healthcare Domain

This section will give an introduction to the prescriptive healthcare domain. More specifically, this section is an introduction to the problem of *polypharmacy*. Section 5.2.1 explains the concept of polypharmacy. Section 5.2.2 will give insight in how this problem was attempted to be solved in the past.

5.2.1 Polypharmacy: When one pill turns into ten.

Polypharmacy is described by Jansen and Brouwers (2012) as "the concurrent use of five or more different drugs". While positive consequences for this level of drug usage exist, such as a longer life expectancy, polypharmacy also has negative consequences. The inappropriate usage and dosage of drugs may have an adverse effect on the health of the patients, which is caused by drug-drug, drug-disease, or adverse drug reactions (ADRs) (Jansen & Brouwers, 2012; Meulendijk et al., 2015).

Multiple issues are linked to polypharmacy, such as underprescription (Kuijpers, Van Marum, Egberts, Jansen, & OLDY (OLd people Drugs & dYsregulations) Study Group, 2008; Sloane

et al., 2004; Wright et al., 2009), inappropriate usage or over-treatment (Steinman et al., 2006), and a decrease in drug adherence (Claxton, Cramer, & Pierce, 2001; Kuzuya et al., 2000). A study by Claxton et al. (2001) shows that the percentage of patients' drug administration adherence drops from 79% for once daily administration to 51% for four-times daily administration. The HARM study, a 2008 study performed in Dutch hospitals, shows that the percentage of patients admitted into the hospital suffered from medication-related problems is 5.6% (Leendertse, Egberts, Stoker, & van den Bemt, 2008). For elderly over the age of 65, who make up half of the chronically ill polypharmacy patients (Meulendijk et al., 2015), this percentage rises significantly to 16% (Jobse, Mulder, ter Borgh, & Grundmeijer, 2009). However, it has been proven that GPs and geriatricians play a major role in preventing or optimizing polypharmacy under elderly patients(Rollason & Vogt, 2003; Sergi, De Rui, Sarti, & Manzato, 2011).

5.2.2 STRIP: Methods for medication.

Several different methods have been developed in order to aid GPs and geriatricians with preventing or optimizing polypharmacy under elderly patients. These include both implicit and explicit methods. Implicit methods make use of patient information, which is combined with medical rules and knowledge in order to make an informed decision, whereas explicit methods mostly consists of screening tools, question lists, and clinical interactions (Jansen & Brouwers, 2012). The most popular implicit methods are the Medical Appropriateness Index (MAI) (Hanlon et al., 1992), and the "Use, Indication, Safety & Effectiveness (UISE)" (Dutch: "Gebruik, Indicatie, Veiligheid & Effectiviteit (GIVE)") model as described in the PHARM-study (Leendertse et al., 2011). The most popular explicit methods are the Beers criteria (Campanelli, 2012), the Screening Tool of Older People's Prescriptions (STOPP), and the START criteria (Gallagher, Ryan, Byrne, Kennedy, & O'Mahony, 2008). The STOPP and START criteria consists of lists of drugs that are possibly not suitable for elderly people (Verdoorn, Kwint, Faber, Gussekloo, & Bouvy, 2016). Each criterium lists a drug, a condition, and an explanation. An example is shown in Figure 19

Figure 19. An example of a STOPP criterium. This criterium shows that patients with COPD who use non-cardioselective beta-blockers have an (increased) risk of bronchospasm

In order to further combat polypharmacy under elderly people, some of the aforementioned implicit and explicit methods have been combined into one single, systematic medication review method, called the Prescription Optimization Method (POM). Drenth-van Maanen, van Marum, Knol, van der Linden, and Jansen (2009) shows that the quality of prescriptions by GPs for (elderly) polypharmacy patients is increased significantly by using the POM, albeit in an experimental setting.
Today, the GIVE model, POM, STOPP, and START criteria have all been combined into

a single method called the 'Systematic Tool to Reduce Inappropriate Prescribing (STRIP)' method. The STRIP method is a combination of several rules and methods that has been designed to be a complete drug optimization process in primary care, taking the pharmacotherapeutic analyses, as well as patients' medical histories and preferences into account (Meulendijk et al., 2015). Since its creation, STRIP has been included in the Dutch multidisciplinary guideline on polypharmacy in elderly patients (NHG & Verzorging, 2012). Figure 20 illustrates the working of STRIP.

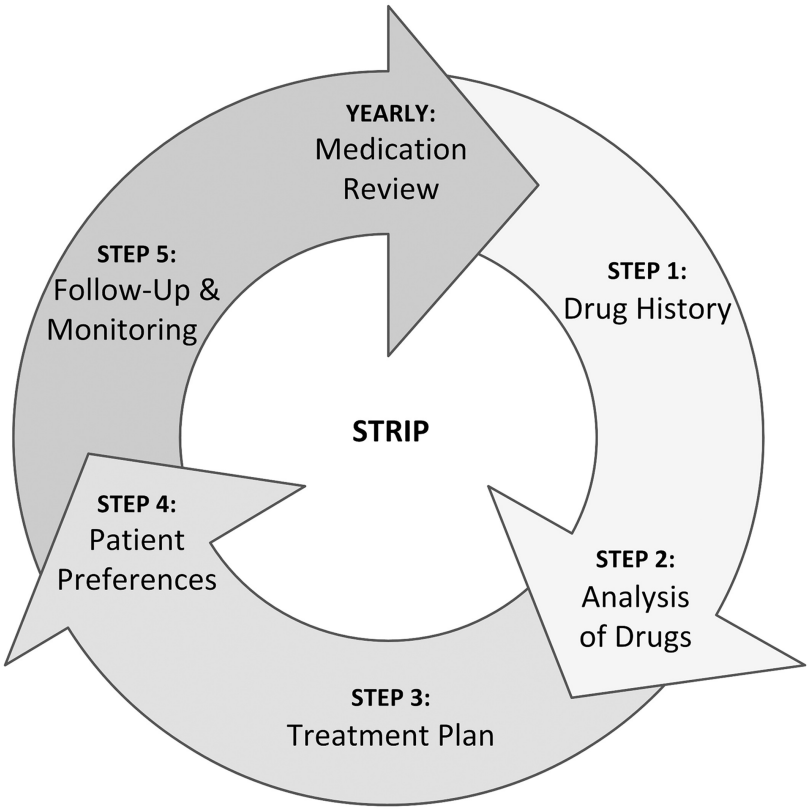


Figure 20. Illustration of the STRIP method by Meulendijk et al. (2015) showing the six different stages of the method

The STRIP method combines both the earlier described implicit (MAI, GIVE), and explicit (START, STOPP) methods, as well as the POM, into a singular pharmotherapeutical analysis method. Most of these methods are performed in step 2 of the STRIP method, as shown in Figure 20. Based on the pharmoterapeutic analysis (step 2), combined with the drug history (step 1) of the patient, a treatment plan can be developed (step 3), which can be tweaked according to the preferences of the patient (step 4). In order to determine the effectiveness of the newly prescribed or stopped-usage drugs, a follow-up is required, as well as monitoring of the patients' health (step 5). In order to prevent polypharmacy from scaling out of control, this process should be repeated yearly.

5.3 STRIP Assistant: From man-work to machine

This section describes the STRIP Assistant, which is a tool that is currently being used to apply the STRIP method in a more effective way. Where Section 5.2 gave insight in how the problem of polypharmacy was solved in the past, this section describes how this problem is currently aimed to be solved with current-day technology. Section 5.3.1 will show the UI of the STRIP Assistant, whereas Section 5.3.2 will go into detail on the back-end of the STRIP Assistant.
The STRIP Assistant is a web-based application that aims to assist General Practitioners (GPs) and pharmacists with pharmacotherapeutic analysis on medical records of patients (Meulendijk et al., 2015). It is based on the STRIP method, which is described in Section 5.2.2. It uses the START and STOPP criteria, as well as established guidelines on clinical interactions, double-medication, contraindications, dosage strength and frequency (Meulendijk et al., 2015).

5.3.1 User Interface.

The STRIP Assistant consists of five different sections. The first section, the *Personalia*, has basic information about the patient, such as age and gender. This information can later be used to help determine the best medication for the patient. The second section, which is contains the medical *History* of the patient, is a list of all of the previous medication and treatments the patient has received in the past, complete with additional information such as frequency, duration, and medication strength. This is shown in Figure 21. It also contains lists of previous medical conditions, measurements, adverse events, and scores.

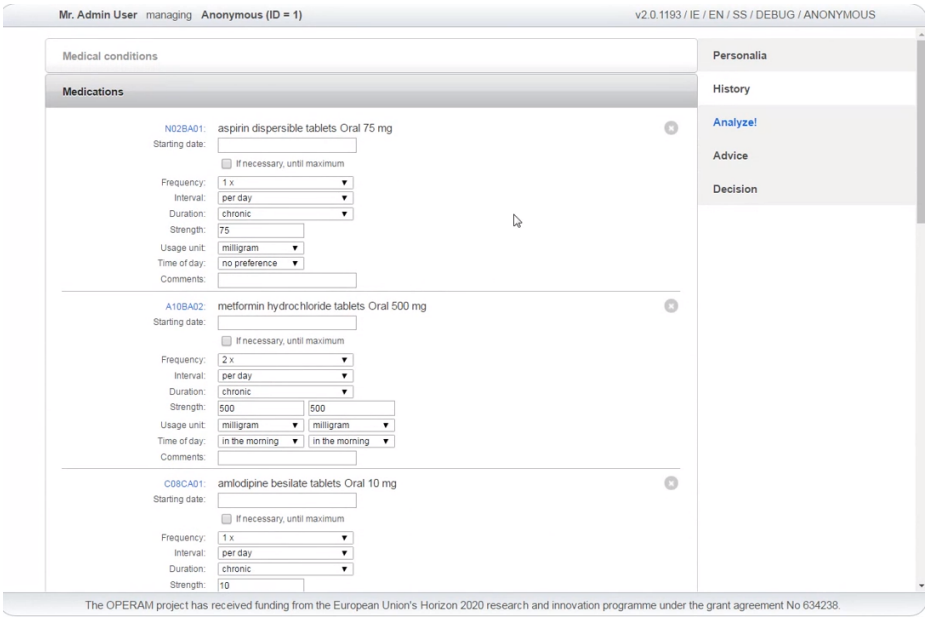


Figure 21. Second section of the STRIP Assistant, which shows the complete medical history of the patient

The third section, which is the largest out of the five, is the *Analysis* section. This section consists of multiple subsections, which is shown in a list on the right. The first subsection

is the *Medications* tab. In this part of the STRIP Assistant, a list of patient complications is shown, as well as a list of treatments. Users of the STRIP Assistant can drag the treatments to the corresponding complications, as shown in Figure 22. The next two categories, *Undertreatment* and *Overtreatment*, consists of the START and STOPP criteria that match with the known complications and medication for the patient. This is shown in Figure 23. The last four categories cover several different types of interactions. These interactions are based on the previously mentioned established guidelines. This is shown in Figure 24.

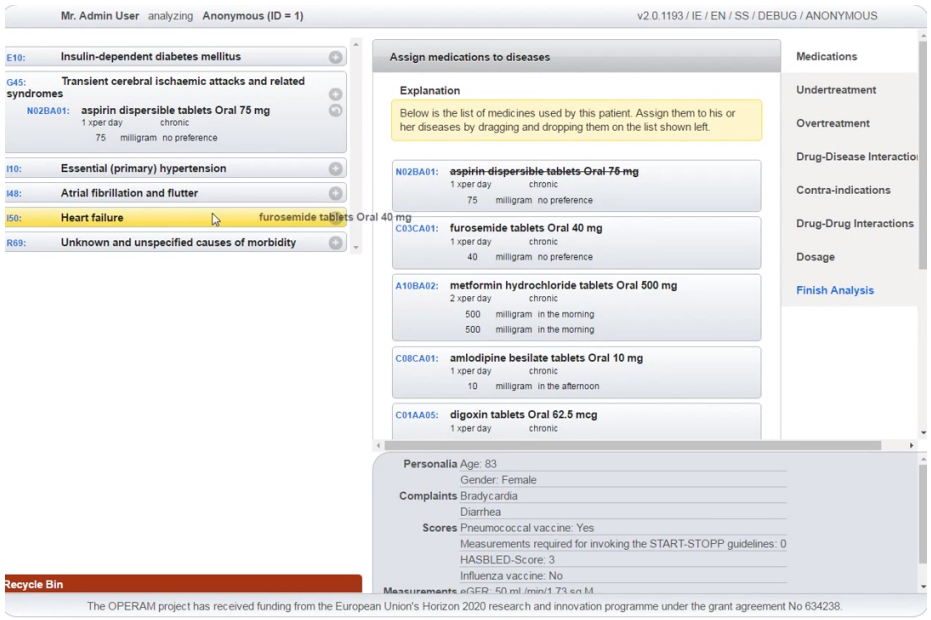


Figure 22. Analysis section of the STRIP Assistant, which is comprised of multiple subsections. This subsection shows the known medication, which can be dragged and dropped on to known complications of the patient

After the analysis has finished, an advice report is generated, which is done in the *Advice* section of the STRIP Assistant. This can be done in multiple different formats, to enable the STRIP Assistant to work alongside other medical applications if needed. It shows a complete overview of all of the current complications of the patients, with the corresponding medication. The UI for this part of the STRIP Assistant is shown in Figure 25 When generating an internal physician report, it also states all the started, stopped, and changes medication of the patient. The last step in the proces is the fifth section of the STRIP Assistant, the *Decision* section. This section is shown in Figure 26. Here, the user can enter comments on why certain medication is stopped, started or changed.

5.3.2 Back-end.

The STRIP Assistant is an application, which is mostly written in Java. It is comprised of ten different packages, with possible expansion possible if more data sources are coupled to the STRIP Assistant. It is made up by five core packages. The complete set of packages can be found in Figure 27. The **strip.*core*** package is the central package of the application. It contains a definition for a basic servlet, which is a small Java program, running within a Web server, that receives and responds to requests from Web clients (Oracle, 2015). The

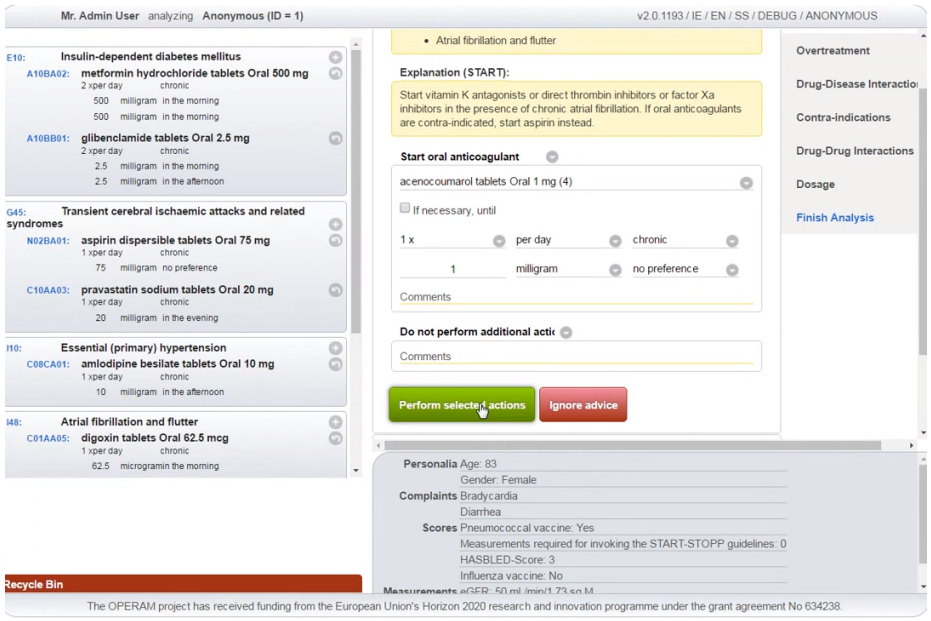


Figure 23. This subsection of the Analysis section of the STRIP Assistant shows the Undertreatment tab, which lists all of the matching START criteria for the known medication-complication combinations of the patient

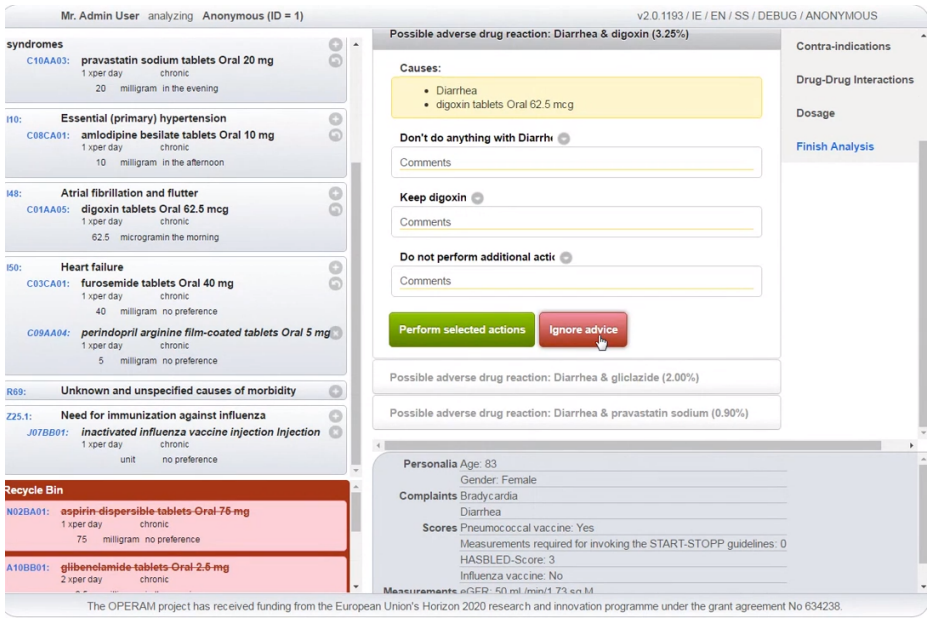


Figure 24. This subsection of the Analysis section of the STRIP Assistant shows the drug reactions. The known medication-complication combinations of the patients are matched against established guidelines on medication interactions

package also contains definition for a SQLManager (to communicate with its' database), a patient manager (to handle patient data), and a class to create singletons of these two classes.

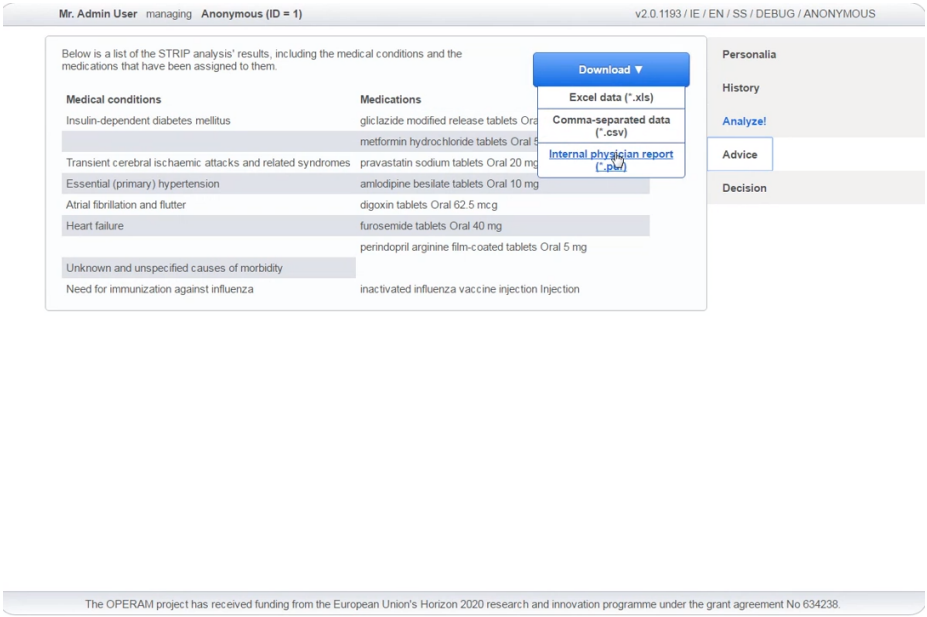


Figure 25. Advice section of the STRIP Assistant. In this section, the user is shown an overview of all of the complication-medication combinations for the patient, and has the option of generating and downloading a detailed report

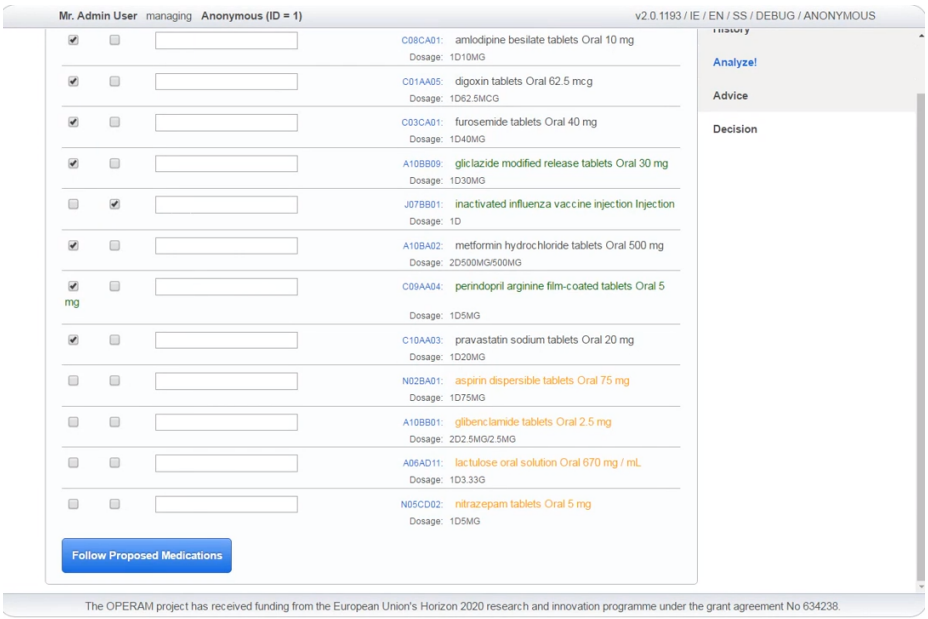


Figure 26. Decision section of the STRIP Assistant. In this section, the user can comment on why certain medication is started, stopped or changed

The second class, **strip.*model***, is dependent on **strip.*core***. Likewise, **strip.*core*** is dependent on **strip.*model***. This class contains all of the type definitions for the STRIP Assistant, and acts similarly to a Model class from a Model-View-Controller (MVC) framework. **strip.*input*** is dependent on both **strip.*core*** and **strip.*model***, and is responsible

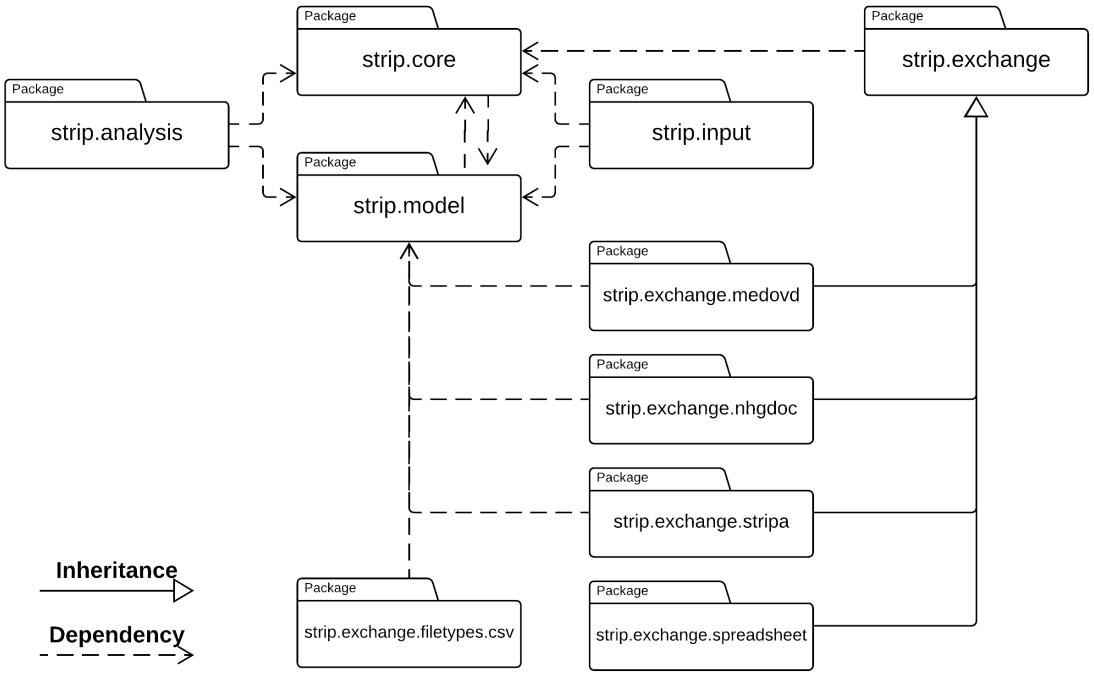


Figure 27. UML Class Diagram of the Java packages within the STRIP Assistant, with inter-package dependencies and inheritances

for handling the autocomplete request received from the UI, and for serving an answer back to the user. **strip.*exchange*** is responsible for the communication between the core of the STRIP Assistant and external data sources, from which the STRIP Assistant receives patient data. Coupled to this are multiple different exchange packages, which all have a dependency on the **strip.*model*** package, and inherit their basic functionality from the **strip.*exchange*** package. Lastly, the **strip.*analysis*** package is the package which takes the patient data, and applies the START/STOPP to it. This will be the main focus of Chapter 7, in which it will be looked to be replaced by implementing DRL principles.
The back-end of the STRIP Assistant is run on a Apache Tomcat 7.0 server. From here, the application generates servlets, which generate valid HTML, JavaScript, and JQuery code. The patient data is stored in a MySQL database, on a separate MySQL server. Additionally, data about medication, complications, and dosages is also stored in this database.
It has already been shown that medication optimization improves significantly when using the STRIP Assistant (Meulendijk et al., 2016). However, earlier validations of the STRIP Assistant showed a decrease in efficiency, despite an improvement in effectiveness (Drenth-van Maanen et al., 2009; Meulendijk et al., 2015). It is hypothesized by the author that the implementation of DRL principles will improve the usability of the STRIP Assistant, and will thus increase the efficiency, effectiveness, and satisfaction of the STRIP Assistant. The implementation process will be discussed further in Chapter 6.

6 CRISP-DRL

This chapter introduces a new variant on the CRISP-DM method, aimed at DRL. This method will thoroughly be explained, with each of the stages, steps, tasks and outputs listed. This chapter, combined with the next chapter, will also aim to answer research question (Q2), which asks: *"How can an application in a new domain be transformed in such a way that it represents a game with rules, input and output?"*.
The implementation of DRL principles into the STRIP Assistant will be carried out in a method akin to the CRISP-DM method. An overview of this implementation methodology can be seen in Figure 28. In this figure, five different stages can be made out, which are shown with different colors. The tasks and output for each specific step of the implementation are laid out in Figure 29. In this figure, the colors of the elements match the colors of the tasks in Figure 28. The stages, steps, tasks, and outputs are based on Chapman et al. (2000).
The next subsections will go into detail on the different stages that make up this implementation methodology, which will be named *CRISP-DRL* within the context of this thesis.

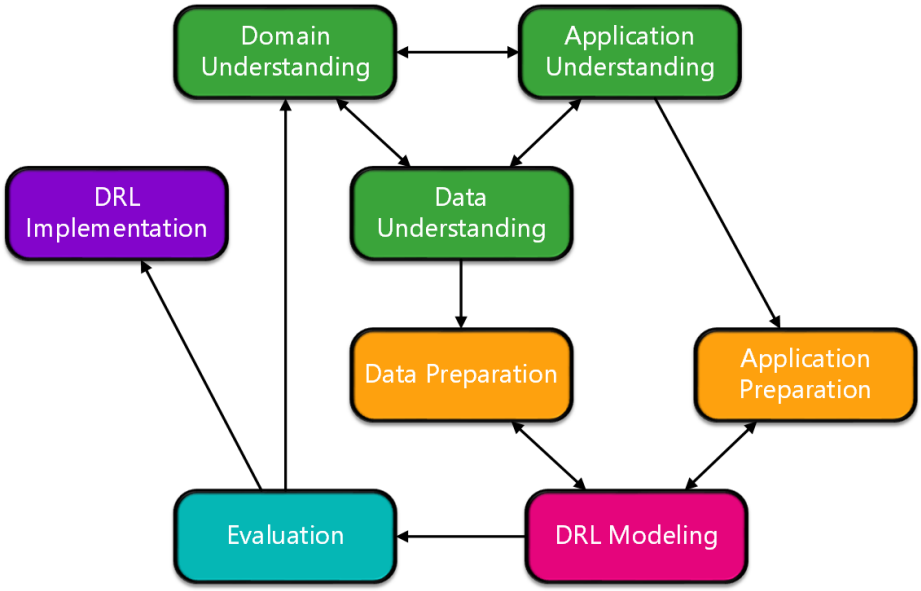


Figure 28. CRISP-DRL: A variation of CRISP-DM, aimed at Deep Reinforcement Learning, consisting of five main stages

6.1 Understanding

The first stage, which is the *Understanding* stage, is about scoping out the application within the domain, and the data this application handles. In order to aid the implementation process, mapping out these three parts is vital. The three steps within this stage are carried out simultaneously, and new insights within one stage might lead to changes within the other stages.
Table 3 gives an overview of the tasks and output of the *Application Understanding* step. This step is one of the two steps that has been added to CRISP-DM in order to create

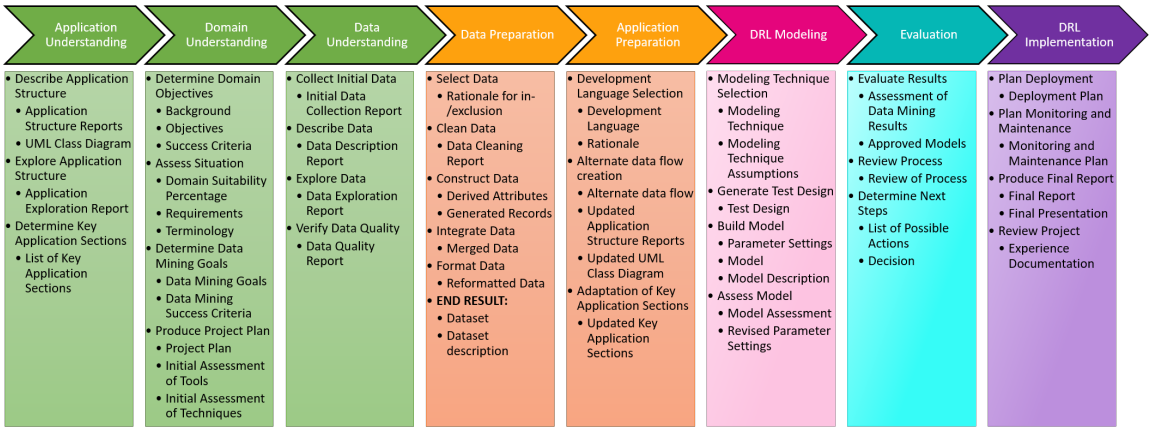


Figure 29. Overview of the different steps of CRISP-DRL, with their respective tasks and outputs

Table 3
Tasks and goals of the Application Understanding step, which is part of the Understanding stage

Task	**Output**
Describe Application Structure	Application Structure Reports UML Class Diagrams
Explore Application Structure	Application Exploration Report
Determine Key Application Sections	List of Key Application Sections

CRISP-DRL. It consists of three different tasks. The first task is to describe the application structure. When implementing DRL principles into an application, it is important to know how this application works, and how it is structured. This task outputs an Application Structure Report, and UML Class Diagrams. These two outputs will later be updated in the *Application Preparation* step. This step will be discussed in Section 6.2.

The second task is to further explore the application structure. This is done in order to further get a grip on the application, and to identify how the application functions at runtime. This tasks results in an Application Exploration Report. Lastly, the final task of this step is to determine the key sections of the applications. It is important to know which parts (e.g. which classes or packages in a Java application) are to be enhanced or replaced with a DRL implementation. This task outputs a list of key application sections.

The second step of this stage, the *Domain Understanding* step, was already present within CRISP-DM, albeit under the name of *Business Understanding*. However, as shown by Spruit and Lytras (2018), this can be exchanged with *Domain Understanding* as well. The main tasks and outputs for this step are shown in Table 4.

The first task of this step is to determine the domain objectives. When implementing DRL principles into an application within this domain, it is important to know how to make sure this implementation goes according to plan. As such, it is important to gather information about the domain at hand, and determine objectives and success criteria based on this knowledge.

Table 4
Tasks and goals of the Domain Understanding step, which is part of the Understanding stage

Task	**Output**
Determine Domain Objectives	Background Objectives Success Criteria
Assess Situation	Domain Suitability Percentage Requirements Terminology
Determine Data Mining Goals	Data Mining Goals Data Mining Success Criteria
Produce Project Plan	Project Plan Initial Assessment of Tools Initial Assessment of Techniques

After this has been established, the next task within this step is to assess the current domain situation. Before proceeding with the implementation of DRL principles, it is important to know whether the domain is suitable for this technology. If it is deemed suitable, then it is important to gather requirements for the implementation, and to establish a terminology.

Determining a set of data mining goals is the next task within this step. This is done in order to make sure that the right amount of correct data is being gathered in a structured way, with minimal overhead. As such, this task results in a list of data mining goals, as well as a list of data mining success criteria in order to test if the data mining process has been carried out correctly.

The fourth and final task in the *Domain Understanding* step is the production of a project plan. This is done to make sure all of the documentation is in a single place, and contains a single version of the truth. The outputs for this task are the aforementioned project plan, as well as initial assessments of the tools and techniques needed in order to carry out the project plan, and thus the complete implementation.

Table 5
Tasks and goals of the Data Understanding step, which is part of the Understanding stage

Task	**Output**
Collect Initial Data	Initial Data Collection Report
Describe Data	Data Description Report
Explore Data	Data Exploration Report
Verify Data Quality	Data Quality Report

The last step of the *Understanding* stage is the *Data Understanding* step. This step is all about understanding the data that is used within the application, and the testing dataset that will be gathered in order to train the DRL implementation. As shown in Table 5, it consists of four tasks, each with one report as an output.

6.2 Preparation

Once the first stage is complete, the whole application should be scoped out. The second stage, which is the *Preparation* stage, aims to prepare the application and the data it uses for the implementation of DRL principles. This stage involves rewriting data inputs, or mapping out exactly where a DRL solution will be implemented within the application.

Table 6
Tasks and goals of the Data Preparation step, which is part of the Preparation stage

Task	**Output**
Select Data	Rationale for in-/exclusion
Clean Data	Data Cleaning Report
Construct Data	Derived Attributes Generated Records
Integrate Data	Merged Data
Format Data	Reformatted Data
Step Output	Dataset Dataset description

The first step of this stage is called the *Data Preparation* step. The main goal of this step is to get a suitable dataset, with which the neural network will be tested and be ran in production. As such, the main output of this step is a dataset, as well as a description of said dataset. In order to achieve this, five different tasks have been selected, which is shown in Table 6

The first task of this step is to select the correct data. Within the application, lots of data is flowing from and to different endpoints. However, not all of this data is needed for the implementation of DRL principles. For example, in the current version of the STRIP Assistant, information about a patient's name is stored within the database. This is useful for the GP in order to know who is being treated. However, for a neural network, this information is not needed. This task outputs a rationale for inclusion and exclusion of data.

The next task in this step is to clean the data. In order to build an accurate DRL implementation, it is important to have correct data as an input, as this will greatly increase the accuracy of the implementation. Thus, it is undesirable to have incorrect, incomplete, or misplaced data in the dataset. After the dataset has been cleaned, this task will result in a data cleaning report.

Next, the data can be constructed. In this step, new records can be generated if needed, and new attributes can be added to the dataset. For example, when both a person's body weight and body length is present in a dataset, one can opt to add the 'Body Mass Index (BMI)' attribute to the dataset. Once this is done, the data can be integrated. The merging of different datasets from different sources will result in one merged master data set. Lastly, the data might need reformatting in order to be passed to the DRL implementation at a later stage.

The other step in this second stage is called the *Application Preparation* step. This step, which follows the *Application Understanding* step from the previous phase, was not present in CRISP-DM. It consists of three different tasks, which are listed in Table 7.

Table 7
Tasks and goals of the Application Preparation step, which is part of the Preparation stage

Task	Output
Development Language Selection	Development Language Rationale
Alternate Data Flow Creation	Alternate Data Flow Updated Application Structure Reports Updated UML Class Diagrams
Adaptation of Key Application Sections	Updated Key Application Sections

The first task of this step is to choose a development language for the implementation. While the logical option might be to choose whichever language the application itself is written in, this is not always the best option. Some languages might be preferable over others. As such, this task outputs a development language, as well as a rationale for this choice.
The next task is to create alternate data flows within the application. As discussed during the data preparation step, data flows from and to lots of different endpoints within the application. However, with the implementation of DRL principles, some flows might become obsolete, or incorrect. As such, an alternate data flow should be created for these flows. The outputs for this task are, apart from the alternate data flows, updated structure application reports, and updated UML Class Diagrams.
Lastly, not only the data flows should be taken care of, the key application sections might also need adaptation in order to correctly handle DRL. For example, model classes might need to be rewritten in order to better accommodate the data structure that is needed for a DRL implementation to work properly. The output for this task are the updated key application sections. With this final task complete, the application and data should now be fully prepared for the next stage, in which the modeling process will start.

6.3 Modeling

The *Modeling* stage involves creating training and validation data, training the DRL implementation, and tweaking the parameters of the network if the results are unsatisfactory. This stage has only one step, which is called the *DRL Modeling* step. This step has four different tasks, which are listed in Table 8.
The first task in this step is to select a modeling technique. Like the selection of the development language from the *Application Preparation* step, selecting a suitable modeling technique can be quite a challenge, as some techniques are more favourable than others. The output of this task is a modeling technique, as well as a list of assumptions made about this technique, the model, the dataset, and the rest of the application.
Next, it is time to generate a test design. In order to correctly test whether the model works, it is important to test its performance. After this task is completed, the resulting output will thus be a test design.
When these two tasks are done, the model building can begin. This task is the part of the step where the actual programming is done. The output for this step is the model, which is the (conceptual) implementation of DRL, as well as a description for the model, and the

Table 8
Tasks and goals of the DRL Modeling step, which is part of the Modeling stage

Task	Output
Modeling Technique Selection	Modeling Technique Modeling Technique Assumptions
Generate Test Design	Test Design
Build Model	Parameter Settings Model Model Description
Assess Model	Model Assessment Revised Parameter Settings

parameters of the model.
In order to test whether the model is correct, the final task of the step needs to be performed, namely to assess the model. This means tweaking the parameters of the model in order to determine if the model gives correct answers. If this is not the case, parameter settings can be revised. This task also outputs a model assessment. When this assessment is positive, it is safe to proceed to the next stage.

6.4 Evaluation

After the training has been completed, the *Evaluation* stage starts. While the model has already been evaluated somewhat in order to tweak the network parameters, this stage will evaluate the final product, and will evaluate whether all domain needs have been satisfied. If this is not the case, the complete process can be started again. If the evaluation is positive, the final stage can begin.

Table 9
Tasks and goals of the Evaluation step, which is part of the Evaluation stage

Task	Output
Evaluate Results	Assessment of DRL Implementation Results Approved Models
Review Process	Review of Process
Determine Next Steps	List of Possible Actions Decision

Table 9 shows the tasks and output of the *Evaluation* step, which is the only step in the equally named stage. It consists of three different tasks, the first of which is the evaluation of the results. After the modeling has been done, the parameters have been tweaked, and the remodeling has also been completed, the model is approved. This is the output of the task, along with an assessment of the (preliminary) DRL implementation results.
The second task of this step is to review the complete implementation process. This is the time to check whether everything has been carried out according to the project plan that was produced as an output in the *Domain Understanding* step.
Lastly, the final task of this step is to determine the next steps. If the project was carried

out successfully, the implementation is working as intended, and the results it produces are correct, then it might be wise to automate this application further. However, if the process has not gone as planned, and major design overhauls need to be made, it is possible to decide to scrap all of the delivered work and to start all over again. In this case, the next stage in the model is the *Understanding* stage.

6.5 Implementation

When the *Evaluation* stage has resulted in a positive decision, the final stage begins. In this fifth and final stage, the *Implementation* stage, the final model is implemented into the application.

Table 10
Tasks and goals of the DRL Implementation step, which is part of the Implementation stage

Task	***Output***
Plan Deployment	Deployment Plan
Plan Monitoring and Maintenance	Monitoring and Maintenance Plan
Produce Final Report	Final Report Final Presentation
Review Project	Experience Documentation

Apart from the deployment itself, which has not been listed here, there are four different tasks that make up the *DRL Implementation* step. These tasks, complete with their output, are listed in Table 10. The first task, which is the planning of the deployment, seems the most logical to start with. This tasks not only covers *when* the implementation will take place, but also *where*, in *what* way, and by *who* the implementation will be carried out.
After this has been completed, the next task is to plan monitoring and maintenance. Like all pieces of software, the work doesn't end when the software is released. It needs to be monitored constantly and maintained regularly. As such, the output for this task is a monitoring and maintenance plan.
After these plans have been set up, the application is running, being monitored, and being maintained, it is time to produce a final report. This means gathering all of the previously outputted plans, combining these into a single master document, and writing this in a coherent, structured manner. This tasks does not only produce a final report as an output, but can also output a final presentation.
Lastly, it is important to review the project. This can be done on many different levels, such as development level, requirements level, or even on a personal level. By precisely documenting what went good and what went not so good, mistakes that were made in the past are less likely to happen again in the future.

7 Applying CRISP-DRL for STRIPAI

This section describes the development process of the implementation of DRL principles into the STRIP Assistant. The development process will follow the staged approach introduced in Chapter 6. Each of the following subsections will discuss a stage of the CRISP-DRL methodology, and will show the outputs for each of the tasks, albeit in a simpler version. When spoken about a report in the output for a task, an explanation of what has been done will be given instead. The fourth stage, the Evaluation stage, will not be discussed in here, but will be discussed in Chapter 8. Lastly, as this is a thesis, the solution will not (yet) be implemented. As such, the final stage of CRISP-DRL , the Implementation stage, will not be discussed.

7.1 Understanding

The first stage of the development is the Understanding stage. The goal of this phase, which is divided into three different steps, is to get a clear overview of the domain, the application, and the data that will be used. While the steps within this stage can be done in any particular order, the same order of steps from Section 6.1 will be used.

7.1.1 Application Understanding.

As described in Section 5.3, the STRIP Assistant is an application which is primarily written in Java. It is based on the principle of servlets, which are small Java programs running on a web server. The application is running these servlets on a Tomcat 7.0 server, and its data is being stored in a MySQL database on a MySQL server. Figure 27 shows an overview of the different classes that comprise the main part of the STRIP Assistant.

While looking further into the application, it was noted that the **strip.*core*** and **strip.*model*** both play a large part in the STRIP Assistant, whereas the **strip.*analysis*** package has a dependency on both of these classes. The last package is also where the rule engine is located, which is responsible for taking the input, and firing rules that give a certain output. As such, it is one of the locations where there should be searched for an implementation possibility.

Lastly, while looking deeper into the application, some key application sections can be listed. As mentioned before, both the **strip.*core*** and **strip.*model*** packages play a major role in the STRIP Assistant, as well as the **strip.*analysis*** package. As such, these three packages are listed as key application sections. In addition to this, the MySQL database is also a key application section.

7.1.2 Domain Understanding.

As discussed in Section 5.2, the chosen domain is the prescriptive healthcare domain. The main goal for the implementation is to help combat the concept of polypharmacy, which is explained in Section 5.2.1. As such, the main objective for this implementation is to improve the STRIP Assistant in order to help preventing polypharmacy. This is done successfully when the usability of the 'improved' STRIP Assistant is significantly higher than the usability of the 'regular' STRIP Assistant. Since usability is a construct of three different variables, there are three different success criteria.

As an assessment, a Domain Suitability Percentage has been calculated for the implementation of DRL principles in the prescriptive healthcare domain. Section 5.1 showed that the prescriptive healthcare domain has an expected suitability of 66%.
It is vital for the 'improved' STRIP Assistant to be as correct or more correct as the current version of STRIPA. This means that the number of false positives and false negatives of the improved STRIPA should be equal to, or lower than the regular STRIPA. In addition to this, the usability of the improved version needs to be significantly higher than the usability of the regular version.
Because the implementation concerns principles of DRL, data is extremely important. Since a DRL implementation best performs when presented with a large dataset, it is important to gather as much data points as possible, while keeping the number of features relatively low. This is done to prevent running into the so-called *Curse of Dimensionality*, a phenomenon described by (Bellman, 2013) that states that the volume of the data space is defined as the number of features times the number of data points used for training. A rule of thumb proposed by (Theodoridis, Koutroumbas, et al., 2008) states that there should be at least five training points for each feature in the dataset.
The main techniques that will be used for implementing DRL principles into the STRIP Assistant will be the programming language Python (version 3), which will be working in tandem with *Tensorflow* and *Keras*, two DL packages for Python. In addition to this, the *numPy* and *pandas* packages will be used.

7.1.3 Data Understanding.

The initial data used is a testing dataset from the STRIP Assistant. While this does concern testing data, the medical aspects are not randomly generated, and are thus deemed as medically 'correct'. At a later stage, a larger dataset with real data is used.
The data in the STRIP Assistant's database is divided over 39 different tables. However, this also contains tables that are associated with end-user roles, or tables that contain data such as chat logs, or medication-id-name translation tables. As such, not all 39 tables will be used. The main tables that will be used are shown in Table 11.
The dataset contains data about 512 patients. This data has been reviewed by medical experts connected to the OPERAM 2020 project. As such, the correctness of this dataset is guaranteed, which makes the data quality very high.

7.2 Preparation

Once the preparation is complete, the second stage of the process can start: The Preparation stage. The goal of this stage, which is divided into two steps, is to prepare the application and the data for the implementation process. Just like the first stage, the steps in this stage can be done in any particular order. However, the same order of steps from Section 6.2 will be used.

7.2.1 Data Preparation.

As mentioned in Section 7.1, only seven of the 39 available tables will (partially) be used within this implementation. Additionally, some data will be grouped together. For each set of data, the relevant SQL statement will be given. Additionally, all Python scripts used for the Data Preparation step can be found in Appendix C

Table 11
Table names and table descriptions of the main tables used within the implementation of DRL principles into the STRIP Assistant

Name	Description
patient	Contains all the relevant personal information about a patient, such as their age and sex
medications	Contains an overview of all of the known medication within the STRIP Assistant
measurements	Contains a set of medical measurements for each patient
treatments	Contains a log of which patient uses what medication, for what complaint
complaints	Contains a list of what patients suffers from which complaints
adverse-events	Table that contains a translation from complaintID to natural language, as well as a grouping code that merges multiple complaints into one category
track	A log of of which patient has been assigned what medication to treat which complaint

Firstly, the patient data is gathered from the patient table. For each patient, their age, ethnicity, and sex is collected. While more information is stored in the database (namely the first and last names of the patients), this information is not queried from the database for privacy reasons.

```
SELECT
        `id`, `ethnicity`, `gender`, `age`
FROM
        `patients`;
```

Figure 30. SQL statement to collect patient information

The measurements table contains medical measurements for a patient. These measurements can be a patient's heart rate, or oxygen saturation levels.

```
SELECT
        `patientID`, `measureID`, `value`
FROM
        `measurements`;

/* Final Python output for single patient: ['1' '0' '78' '0' '0' '0' '0' '0' '0' '0'
'0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0' '0'] */
```

Figure 31. SQL statement to collect medical measurements from each patient

Next, both the medications and the treatments table have been used in order to gather all medication categories a patient uses. While using all medication instead of medication categories might give a more accurate result, this has been done in order to prevent a

massive increase in the feature space. In order to make sure only correct verified medication is chosen, the medication status needs to be 'ASCERTAINED'.

```
SELECT
        `patientID`, `standardCode`
FROM
        `medications` MDS
                JOIN
        `treatments` TRT
                ON
        MDS.id = TRT.medicationID
WHERE TRT.status = 'ASCERTAINED';
```

Figure 32. SQL statement to collect all used medication categories for each patient

Like the medication categories, the complaint categories (from now on: complications) are created by joining the complaints table with the adverse-events table. This merge is made in order to constrain the amount of features the implementation needs to handle. The *WHERE* clause in the SQL statements make sure that only verified complaints are extracted from the database.

```
SELECT
        `patientID`, `standardCode`
FROM
        `complaints` COM
                JOIN
        `adverse-events` ADV
                ON
        COM.adverseeventID = ADV.id
WHERE COM.status = 'ASCERTAINED';
```

Figure 33. SQL statement to collect all used complications for each patient

Finally, in order to test the implementation, a query is done on the track table in order to retrieve the firing of START and STOPP rules for each patient.

```
SELECT
        `id`, `start`, `stopp`
FROM
        `track`
WHERE `ruleKey` LIKE 'ST%';
```

Figure 34. SQL statement to collect patient information

Now that all of the data is gathered from the database, it is loaded into the STRIP Assistant by a MySQL connector. After all data has been loaded, it is transformed into a NumPy array. The code for loading the data from the database into the application can be found in Figure 43 and Figure 44.

The data for complications and medications requires additional preprocessing, for a couple of reasons. Within Tensorflow, which will be used within this implementation, and these kinds of implementations in general, it is not possible to use input data with variable input length. As such, it is not possible to feed the implementation 7 medications for the first patient, 9 medications for the second, and so forth. It is also not possible to feed the implementation both integer data and lists with variable input, for the reason explained as above.
In order to remedy this, both the medication and complication sets have been transformed into a multi-hot encoded vector. This is a vector consisting of only zeroes and ones, where the one can occur more than once. This is in contrast to a one-hot encoded vector, where a one can only occur once. Figure 45 shows an additional Python module that aids in preprocessing the data. After this is done, the data is divided up into training sets and test sets. A code snippet for this operation is shown in Figure 35.

```
### BEGIN IMPORTS
from strip import input_preprocessor as sip
import numpy as np
import random as rd

### END IMPORTS

personalia, complications, medications, target = sip.generateIO()

### Start construction of the NN structure
### First, generate the training and testing datasets
rd.seed(42)
trainingIndices = np.sort(random.sample(range(1,len(personalia)),
 ↪  int(len(personalia)*0.9)))
testIndices =
 ↪  np.sort(np.array(list(set(range(1,len(personalia))).difference(trainingIndices))))

#Then, construct training and test sets
def train_test(data, column='x'):
    if column == 'x':
        return (data[trainingIndices,:],data[testIndices,:])
    elif column == 'y':
        return (data[trainingIndices].flatten(), data[testIndices].flatten())

pers_x_train, pers_x_test = train_test(personalia)
comp_x_train, comp_x_test = train_test(complications)
med_x_train, med_x_test = train_test(medications)
y_train, y_test = train_test(target, 'y')
```

Figure 35. Code snippet that handles the creation of training sets and test sets

7.2.2 Application Preparation.

As apparent from the Data Preparation step, the main development language for this implementation will be Python. Python has been chosen over Java, the main development language of the STRIP Assistant, for multiple reasons.

The first and main reason is the choice of machine learning library. For this implementation, both Tensorflow and Keras will be used. These libraries are both available for Python. Next, the implementation will be developed parallel to the STRIP Assistant, and not *in* the STRIP Assistant. This means that the data is pulled directly from the database, and fed into the implementation. As a result, no changes need to be made to the list of key application sections, the dataflows, and the class diagrams from Section 7.1.

7.3 Modeling

This stage and step of the process concerns the selection, building, and assessment of the DRL model that will be built. The first task in this step is to select a modeling technique. As a DRL implementation is built, the main modeling technique will be a neural network. This network will be a fully connected neural network, with one input layer, one output layer, and one or more hidden layers.

To train and test the network, an 80/20 split is made, where 80 percent of the data set is reserved as training data, and 20 percent is used to verify the results. This is done by the code snippet from Figure 35.

Once this split is made, the models are trained. One model is trained on personalia and measurement data, one on medications, and one on complications. Each model is trained for 20 iterations (or epochs), uses Tensorflow's *adam* optimizer, and a sparse categorical crossentropy as a loss function. Initially, each model is also trained with two (fully connected) hidden layers, each consisting of 8 nodes. The output layer consists of four output nodes, one node for each possible output class, which is activated by a softmax function. A visual representation can be found in Figure 36.

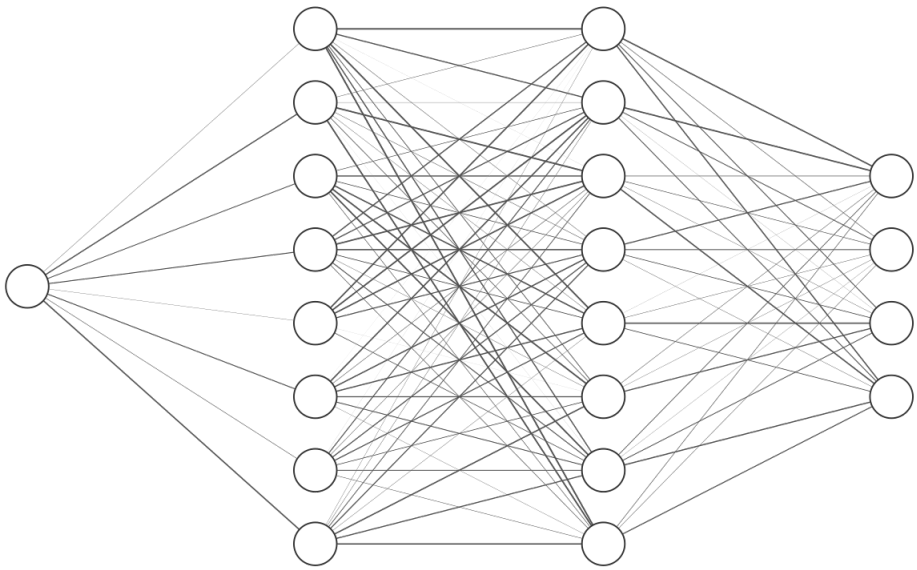


Figure 36. Visual representation of a trained neural network, where the gradient of an edge represents the weight of the edge between two nodes

Further revision of parameters results in a significant improvement in model results. These results will be discussed in the next stage. The final parameters can be found in Table 12.

Table 12
Final configurations for all three trained neural networks

	Personalia	**Medications**	**Complications**
Hidden Layer 1			
- Nodes	4	16	16
- Regularization	L2(0.001)	NA	NA
- Activation	ReLu	ReLu	ReLu
Hidden Layer 2			
- Nodes	4	16	16
- Regularization	L2(0.001)	NA	NA
- Activation	ReLu	ReLu	ReLu
Output Layer			
- Nodes	4	4	4
- Regularization	NA	NA	NA
- Activation	SoftMax	SoftMax	SoftMax

7.4 Evaluation

This section will discuss the fourth stage of the method: the Evaluation Stage. This section will present the results gathered by passing all of the testing data through the trained networks that have been modelled in Section 7.3. First, results for the networks from Table 12 will be shown. After this, a new set of networks will be trained and evaluated in order to come to a better performing model.
After the completion of the Modeling stage, which was discussed in Section 7.3, a model has been constructed, with a revised set of parameters. This model has been trained with 80% of the available data, as discussed in Section 7.2. Using the remaining 20% for validating this model results in a set of multi-class confusion matrices, as well as other measurements about the model. These measurements can all be found in Appendix D, but the main results will be presented in this chapter.
In order to determine the performance of the networks, multiple metrics are used. The first one is the loss function of the network. As stated in Section 7.3, all three neural networks used a sparse categorical cross-entropy as a loss function. The aim during training is to reduce the entropy, without introducing overfitting. Overfitting is usually indicated by 'flat-lining', which happens when the entropy does not increase or decrease during training after a certain period of time.
Another set of metrics that can be used to determine how a network is performing is with Confusion Matrices. These matrices, which are generated by passing the test data through the network using Tensorflow, can give insights in the precision, recall, and accuracy of the network.
To improve the readability of the confusion matrices, an integer coding is used to label the output categories. Table 13 shows the definition of these class labels. This table also shows how often a certain class label occurs within the complete dataset.

Table 13
Explanation of class labels used in classification matrices. The Meaning column indicates whether a START or STOPP rule would have fired in the old STRIP Assistant

Label	Meaning	Occurrence
0	No START, no STOPP	87
1	START, no STOPP	57
2	STOPP, no START	21
3	START and STOPP	347

As shown from this table, there is a severe class imbalance. The class with label 3, which indicates that both START and STOPP rules have fired, represents 67.8% of the population. When aggregating class labels 1, 2, and 3, this percentage rises to 83%. As such, there is a high degree of skewness. In order to combat this, a technique called undersampling can be performed on the majority class. This technique is defined by Japkowicz and Stephen (2002) as "a sampling method in which the entities in the majority class will be, at random, eliminated until a desired distribution is achieved". In order to get a more accurate result, a separate set of networks has been trained, which will be discussed in Section 7.4.2. For these networks, the testing and training data class labels have been pre-aggregated to 0's and 1's. This makes it possible to perform undersampling on the majority class.
The next subsection will describe the results for the networks that have been modelled in Section 7.3. For the networks, it will show the overview of the sparse categorical cross-entropy loss function plotted over time. Then, the results for each network will be presented. These networks have been trained on non-aggregated data.

7.4.1 Non-aggregated Network Results.

This subsection will present the results for the trained networks from Section 7.3. As stated in Section 7.4, there has been no aggregation of the data that has been performed. As a result, there is a severe class imbalance within the dataset, as shown in Table 13. This will most likely have an impact on the results of these networks, which means that a second set of networks should be constructed. These networks will be discussed in Section 7.4.2.
As stated, one of the metrics to determine the performance of the networks is by looking at the loss function. A lower value from the loss function, which in this case will be a sparse categorical cross-entropy, usually indicates a better performing network. Figure 37 shows a graph for the cross-entropies for both the training and validation data sets of each network. As the output layer consists of four different nodes, a cross-entropy within the range of 0 to 3 is deemed acceptable.
The other metric, which are the confusion matrices for each network, will give insight in how a network is performing. These matrices can be generated by feeding the testing data into the network, and comparing the generated outputs to the actual outputs. For these networks, two types of confusion matrices will be generated.
The first type is a binary confusion matrix, where the zero-class and the remainder class will be compared. In this case, the remainder class is defined as the aggregation of the remaining, non-zero classes. The second type of confusion matrix is a multi class confusion matrix. In order to determine the classification results (True Positives, False Positives, False Negatives, and True Negatives) for each class, the method described by Krüger (2016) is

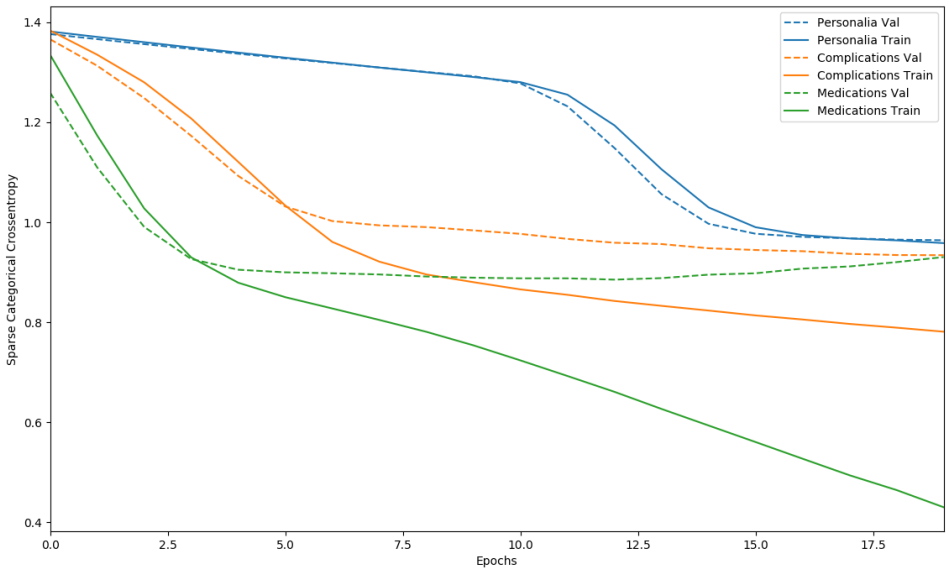


Figure 37. Overview of the sparse categorical cross-entropies of the three different networks, trained on the non-aggregated dataset, plotted over the training epochs. A lower sparse categorical cross-entropy usually indicates a better performing network

used. This method is visualised in Figure 38.
For each network , the distributions for the predicted class labels can be compared to the actual class labels. This is plotted in Figure 39. For each network, the number of occurrences per class label for each network is shown as a bar, along with the normal distribution of these class labels per network as a line.
From this figure, a few observations can already be made from this graph. Firstly, it can be seen that the personalia and complications networks only 'predict' class label 3. This means that the network simply does not predict anything, but always returns that the network always would say to fire both START and STOPP rules. Next, the medications network does not contain class labels 1 and 2 in the predicted class labels. While this network actually tries to predict results instead of outputting a single class label 100 percent of the time, the results differ significantly from the actual class labels.
The next paragraphs will go further into detail on the results of the individual networks, and will aim to show why a new set of networks should be trained that aims to fix the class imbalance currently present within the dataset.

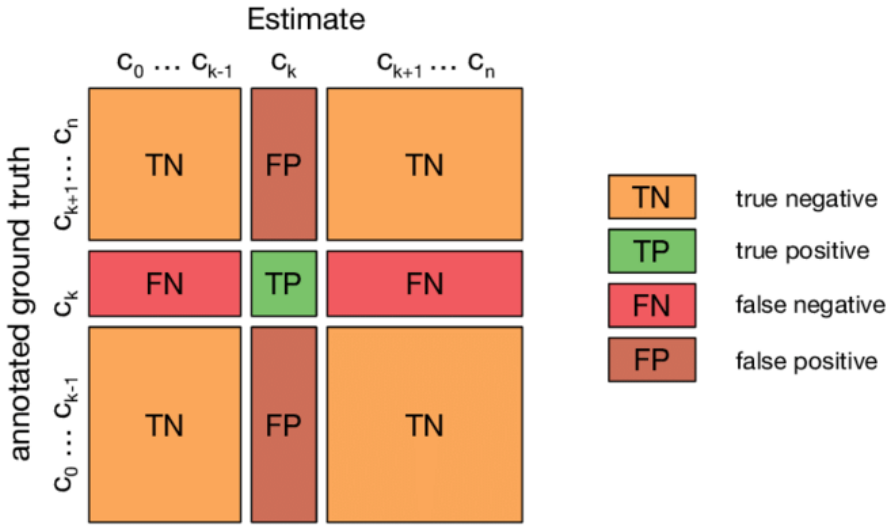


Figure 38. Diagram showing how for a class c_k, the four different classification results can be obtained

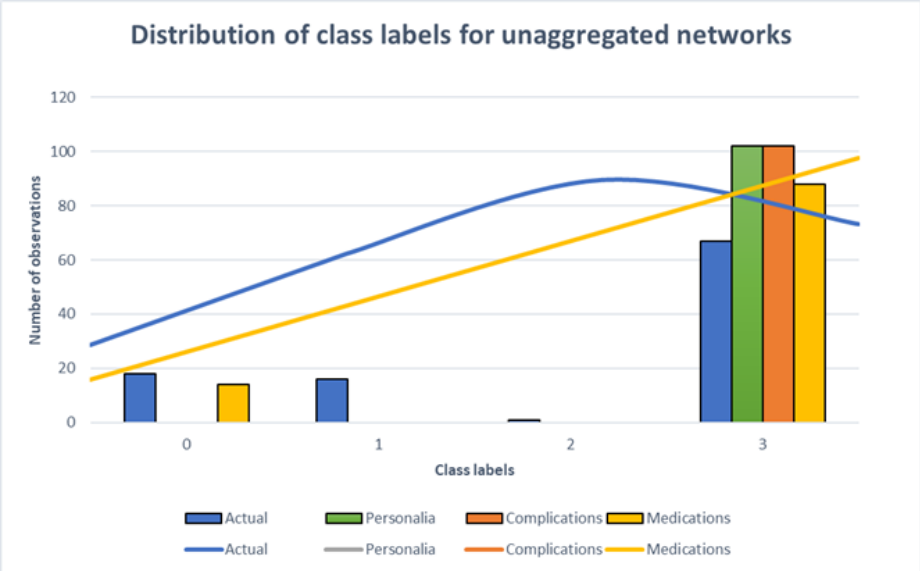


Figure 39. Diagram showing the number of occurrences per class label for each network trained on non-aggregated data as bars, and showing the normal distribution of these class labels per network as a line

Personalia Network.
The first constructed network uses the personal information of a patient combined with medical measurements as input. 102 data points have been used as validation, of which the result can be found in Table 14. The metrics for the confusion matrices from Table 14a and Table 14b can be found in Section D.1.

Table 14
Confusion matrices of the validation results for the trained personalia network trained on the non-aggregated data

(a) *Confusion matrix of the validation results on the trained personalia network, showing the confusion between the zero-class and the remainder class*

		Predicted		
		0	∼	Total
Actual	**0**	0	18	18
	∼	0	84	84
	Total	0	102	102

(b) *Confusion matrix of the validation results on the trained Personalia network*

		Predicted				
		0	**1**	**2**	**3**	Total
Actual	**0**	0	0	0	18	18
	1	0	0	0	16	16
	2	0	0	0	1	1
	3	0	0	0	67	67
	Total	0	0	0	102	102

As shown earlier, the network fails to predict any other value than the class label 3, which indicates that both START and STOPP rules have been triggered. This is mostly due to the class imbalance within the data. As such, the metrics that can be derived from the confusion matrices don't have a significant meaning. The accuracy derived from the confusion matrix in Table 14a is 83.25%, which is higher than the accuracy as reported by Tensorflow, which showed an accuracy of 68.46% on the training data, and only an accuracy of 65.69% on the validation data. The overall accuracy of the larger confusion matrix in Table 14b is 65.69%, which is in line with the accuracy that Tensorflow reports.

Complications Network.
The second constructed network uses the possible complications a patients has in order to determine whether START or STOPP rules should fire. Again, 102 data points have been used as validation. The results can be found in Table 15. The metrics for the confusion matrices from Table 15a and Table 15b can be found in Section D.2.
As visible from Table 15a, the confusion matrix shows the same distribution as the personalia confusion matrix, where the zero class is simply not predicted at all. This means that the metrics, shown in Table 28 show the same metrics as the ones in Table 24. This is also reflected in the larger confusion matrix from Table 15b, which shows the same distribution as the personalia confusion matrix.

Medications Network.
Lastly, the third and final constructed network uses the medications used by patients in order to determine whether START or STOPP rules should be fired. 102 data points have been used as a validation, of which the resulting confusion matrix can be found in Table 16.The metrics for the confusion matrices from Table 16a and Table 16b can be found in Section D.3.

Table 15
Confusion matrices of the validation results for the trained complications network trained on the non-aggregated data

(a) *Confusion matrix of the validation results on the trained complications network, showing the confusion between the zero-class and the remainder class*

		Predicted		
		0	∼	Total
Actual	**0**	0	18	18
	∼	0	84	84
	Total	0	102	102

(b) *Confusion matrix of the validation results on the trained complications network*

		Predicted				
		0	**1**	**2**	**3**	Total
Actual	**0**	0	0	0	18	18
	1	0	0	0	16	16
	2	0	0	0	1	1
	3	0	0	0	67	67
	Total	0	0	0	102	102

Table 16
Confusion matrices of the validation results for the trained medications network trained on the non-aggregated data

(a) *Confusion matrix of the validation results on the trained medications network, showing the confusion between the zero-class and the remainder class*

		Predicted		
		0	∼	Total
Actual	**0**	8	10	18
	∼	6	78	84
	Total	14	88	102

(b) *Confusion matrix of the validation results on the trained medications network*

		Predicted				
		0	**1**	**2**	**3**	Total
Actual	**0**	8	0	0	10	18
	1	3	0	0	13	16
	2	0	0	0	1	1
	3	3	0	0	64	67
	Total	14	0	0	88	102

As shown in these confusion matrices, the network performs better than the networks for personalia and complications in the sense that the network does not 'predict' class label 3 100 percent of the time. However, this network fails to predict class labels 1 and 2, which ultimately results in a network that either does nothing, or fires both START and STOPP rules. This makes the metrics for these confusion matrices highly unreliable.

7.4.2 Pre-aggregated Network Results.

This subsection will present the results for three additional networks that have been trained on a prepared dataset with no class imbalance. The aim of this subsection is to present networks that perform better than their respective counterparts from Section 7.4.1.
As evident from Section 7.4.1, all three constructed networks did not perform well. The networks for personalia and complications only returned class label 3 as a result, whereas the network for medications never returned class labels 1 and 2. This is due to the class imbalance present within the data. As explained in Section 7.4, one method to improve this is through the use of oversampling. In order to get better results, three additional networks will be trained.
The training dataset has been reduced to 138 entries. Exactly half of these entries has a class label of 0, and the other half has a class label of 1. The testing dataset still has 102 entries, and contains a random distribution of 0's and 1's. With respect to Table 12, the only changes that have been made are the amount of nodes in the hidden layers of the *Personalia* network. The size of both hidden layer 1 and 2 have been increased to 8. The training results can be seen in Figure 40.

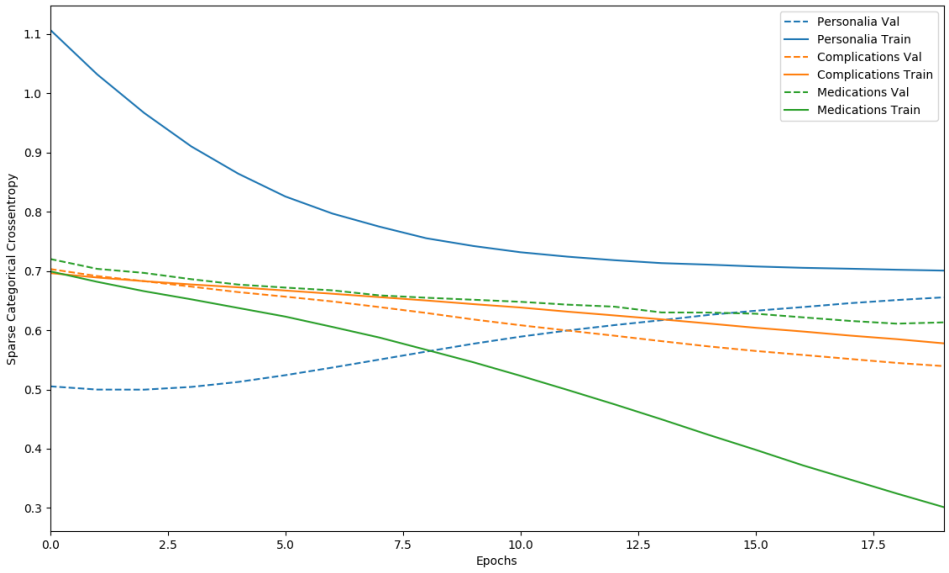


Figure 40. Overview of the sparse categorical cross-entropies of the three different networks, trained on the aggregated dataset, plotted over the training epochs. A lower sparse categorical cross-entropy usually indicates a better performing network

Because this data only contains the class labels 0 and 1 as output, a sparse categorical cross-entropy between 0 and 1 is deemed as acceptable. As visible from Figure 40, all three networks have a cross-entropy value lower than 1 after 20 training epochs. For each network, the distributions for the predicted class labels can again be compared to the actual class labels. This is plotted in Figure 41.

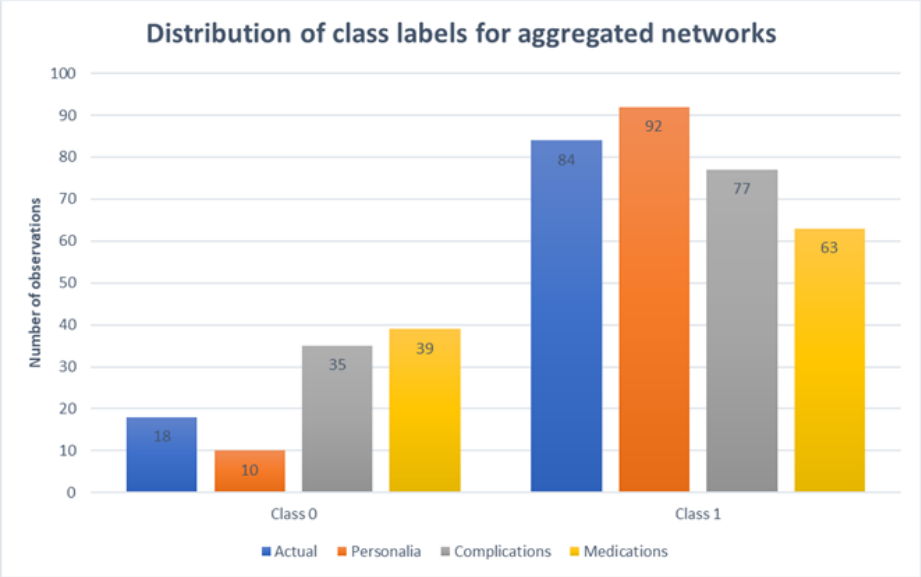


Figure 41. Diagram showing the number of occurrences per class label for each network trained on pre-aggregated data, compared to the actual class labels

From this figure, it is visible that there is a difference between the predicted class labels and the actual class labels, which are coloured blue in this figure. As visible in this figure, there seems to be a significant difference between the results of the trained networks and the actual class labels. These results will be explored in the next paragraphs.

Personalia Network - *Aggregated*.
As a variation to the network from Section 7.4.1, an additional network has been constructed. This network has been trained on less variables, has been aggregated, but contains no skewness. The resulting network is validated with 102 data points. The results of this validation can be found in Table 17.

Table 17
Confusion matrix of the validation results on the trained personalia network, trained with aggregated data, showing the confusion between the two classes

		Predicted		
		0	**1**	Total
Actual	**0**	9	9	18
	1	1	83	84
	Total	10	92	102

As shown in this confusion matrix, the network does not predict the label 1 every single time any more, as was the case in the previous version of this network. This network has an accuracy of 90.1%, but an f1-score of only 64%. While this does seem to suggest that the network performs better than the network trained on non-aggregated data, the network still does not perform optimally.

Complications Network - *Aggregated.*
As a variation to the network from Section 7.4.1, an additional network has been constructed. This network has been trained on less variables, has been aggregated, but contains no skewness. The resulting network is validated with 102 data points. The results of this validation can be found in Table 18.

Table 18
Confusion matrix of the validation results on the trained complications network, trained with aggregated data, showing the confusion between the two classes

		Predicted		
		0	**1**	Total
Actual	**0**	14	4	18
	1	21	63	84
	Total	35	77	102

As shown in this confusion matrix, the network does not predict the label 1 every single time any more, as was the case in the previous version of this network. This network has an accuracy of 75.5%, and an f1-score of only 53%. This suggest that the network performs better in the sense that it actually predicts instead of outputting a single value every time, but worse in the sense that it has a lower accuracy than the previous network.

Medications Network - *Aggregated.*
As a variation to the network from Section 7.4.1, an additional network has been constructed. This network has been trained on less variables, has been aggregated, but contains no skewness. The resulting network is validated with 102 data points. The results of this validation can be found in Table 19.

Table 19
Confusion matrix of the validation results on the trained medications network, trained with aggregated data, showing the confusion between the two classes

		Predicted		
		0	**1**	Total
Actual	**0**	10	8	18
	1	29	55	84
	Total	39	63	102

As shown in the table, this network is more reserved, opting to not trigger any rules and generating an output of 0 significantly more often than the actual class labels. This leads to an accuracy of 63.7%, which is lower than the accuracy of the non-aggregated network data, which showed an accuracy of 84.31%. Furthermore, the f1-score for this network is 35%, which is lower than the f1-score from the non-aggregated data network (50%). With these metrics in mind, it seems that training the network on pre-aggregated data gives a worse classifier than training the network with non-aggregated data, and applying post-aggregation on the results.

8 Results

8.1 STRIPAI: Usability

This section will discuss the results gathered in Section 7.4. More specifically, this section will aim to use the results gathered in the previous section in order to answer research questions (Q3), (Q4), and (Q5) from Section 1.3. (Q3) will be answered in Section 8.1.1, after which Section 8.1.2 will aim to answer (Q4). Lastly, the results from these two questions will be used in order to give an answer to (Q5). This will be done in Section 8.2.

8.1.1 STRIPAI: Effectiveness.

This subsection aims to answer (Q3), which asks *how the combined principles of Deep Learning and Reinforcement Learning can improve the effectiveness of an existing application in a new domain.* In order to answer this question, a null hypothesis first needs to be established. Our hypotheses for this question are as follows:

H_0 There is no significant difference in effectiveness in the current version of the application, and the application with Deep Learning and Reinforcement Learning implemented into it.

H_1 There is a significant difference in effectiveness in the current version of the application, and the application with Deep Learning and Reinforcement Learning implemented into it.

In order to answer this hypothesis, a chi-squared test will be conducted on the confusion matrices for the three networks, as presented in Section 7.4.2, in order to determine the difference in effectiveness between the original STRIP Assistant, and the STRIPAI. For this, a confidence level of 95% ($\alpha = 0.05$) is used. The effectiveness will be tested for all three constructed networks that have been trained on the pre-aggregated dataset. The networks for personalia and complications from Section 7.4.1 will not be considered, as these networks are all outperformed by their respective counterparts.
For the *Personalia* network, which' confusion matrix is presented in Table 17, a chi-squared test has been performed. Based on the data, there is a statistically significant difference in the proportion of patients where no START or STOPP rules would have fired and patients where those rules would have fired in the original STRIP Assistant versus the STRIPAI, $X^2(1, N = 102) = 39.93, p = 0.000$.
For the *Complications* network, which' confusion matrix is presented in Table 18, a chi-squared test has been performed. Based on the data, there is a statistically significant difference in the proportion of patients where no START or STOPP rules would have fired and patients where those rules would have fired in the original STRIP Assistant versus the STRIPAI, $X^2(1, N = 102) = 18.31, p = 0.000$.
For the *Medications* network, which' confusion matrix is presented in Table 19, a chi-squared test has been performed. Based on the data, there is no statistically significant difference in the proportion of patients where no START or STOPP rules would have fired and patients where those rules would have fired in the original STRIP Assistant versus the STRIPAI, $X^2(1, N = 102) = 2.78, p = 0.096$. In addition to this, a second chi-squared test has been performed on the medications network that has been trained on the non-aggregated

dataset. For this network , which' confusion matrix is presented in Table 16a, a significant difference in the proportion of patients where no START or STOPP rules would have fired and patients where those rules would have fired in the original STRIP Assistant versus the STRIPAI was found, $X^2(1, N = 102) = 17.42, p = 0.000$.
Based on the results from all networks, a significant difference between the outputs is shown in all three networks. However, this significant difference is a decrease in effectiveness, and is also largely dependent on whether the respective network is trained on pre-aggregated data or normal data. For example, the medications network showed a significant difference between the actual and predicted class labels when trained on the non-aggregated data, but did not show this difference when trained on the pre-aggregated data, where there was no class imbalance.
These significant differences, in combination with the confusion matrices from Section 7.4, mean that the null hypothesis can be rejected. Using the STRIPAI in order to predict START and/or STOPP rules being fired does not lead to an increase in effectiveness with respect to using the old STRIP Assistant. However, based on the results from the *Medications* network, there seems to be a possibility that a significant increase in effectiveness might be possible in the future.

8.1.2 STRIPAI: Efficiency.

This section will go into detail on the efficiency of the implementation. More specifically, this section will theorize about the differences in efficiency between the original STRIPA, and the STRIPAI, in order to give an answer to sub question (Q4).
Due to the nature of the implementation of the STRIPAI, it is not possible to statistically compare both of the implementations on their runtime. The original STRIP Assistant is written in Java, connects to a MySQL database server, and interfaces with a web front-end. The STRIPAI is written in native Python version 3, connects to the MySQL database server, but doesn't have an interface. The implications will be further discussed in Chapter 10.
In order to determine which implementation is more efficient, one possible solution is to look at the average computational complexity. This is written in the *Big $\mathcal{O}$ notation*. This states that for an algorithm with n inputs, the algorithm will need $\mathcal{O}(x)$ amount of units of time on average. Here, x can be any possible value that contains n, or can be equal to 1. For the original STRIP Assistant, the firing of m START and/or STOPP rules for n patients has an average computational complexity of $\mathcal{O}(n*m)$. This does not include any operations that are performed on something else than the patient dataset, and does also not include time waiting for the data to be retrieved from MySQL, or being pushed to the web front-end.
For the STRIPAI, the firing of the START and/or STOPP rules can be broken down into two parts. The first part consists of training, and the second one consists of running data through the network to get a result. Both parts depend on the characteristics of the network, but can be generalized to answer the question.
To run data through a fully trained, fully connected network, one has to consider all the weights w for all the nodes l_i in all the layers l. We can assume that our network is fully connected. As such, the weights w can be defined as $l * l_i$. Subsequently, the amount of time for all data to pass through the network is $\mathcal{O}(n * w)$.
For the training of the network, the computational complexity depends on the number of training epochs e that are used for training the network. The implementation uses

backwards propagation to train the network, which has a complexity of $\mathcal{O}(n * w)$, where we assume that the network is fully connected. As this done a number of times, the total complexity of training is defined as $\mathcal{O}(n * e * w)$. Thus, the total complexity of the STRIPAI can be described as $\mathcal{O}(n(e * w + w)$. Likewise, does not contain all the necessary steps to extract, transform, and load the data.
In order to determine whether the STRIPAI performs more efficiently than the original STRIPA, one needs to determine whether the number of weights times the amount of training is larger than the amount of rules present in the original STRIPA. In other words, we need to determine whether $(e * w + w) > m$.
For the original STRIPA, there currently are 109 START/STOPP rules in place (Knol et al., 2015). This is divided in 72 STOPP, and 39 START rules. This would mean that $m = 109$.
For the STRIPAI, 20 epochs were used. Furthermore, the smallest network used 3 layers, with 4 nodes in each layer. This is not accounting for the input layer. As such, the smallest network has $3 * 4 = 12$ weights in total. This means that $(e * w + w) = 20 * 12 + 12 = 252$.
This means that in the best case, using the smallest network trained in Section 7.3, the original STRIPA performs more efficiently than the STRIPAI implementation.
Another inefficiency within the STRIPAI, is that it is currently optimized to deal with the small amount of patient data. This is further explained in Chapter 10. As such, any medication or complication that is not yet present in the current dataset is not taken into account to limit the search space of the application and to improve the network effectiveness. This means that for each time an unknown medication or complication is presented to the network, it needs to train the complete network all over again. This is in stark contrast to the original STRIPA, of which the rules don't need to be updated.
Based on these two criteria, we can conclude that the STRIPAI fails to perform more efficiently than the original STRIP Assistant. The computational complexity is higher in the STRIPAI, without taking the inefficiencies as described above taken into account. While a smaller network, or less training epochs can be used to create a more efficient implementation, this will go at the expense of the effectiveness of the implementation.

8.2 Improving the 4VATT Suitability Model

This section describes how the results from Section 8.1.1 and Section 8.1.2 can be used to further improve the 4VATT Suitability model introduced in Section 4.2. By improving this model, a more accurate estimation of the suitability of domains can be made. This will be further exemplified by redetermining the suitability of the prescriptive healthcare domain, from which it was determined in Section 5.1 to have an estimated suitability of 66%.
From the implementation process, some takeaways can be made, which will be discussed in more detail in Chapter 10. For example, the data volume within the STRIP Assistant was far lower than originally expected, due to the aggregations made. As such, the scores given in Section 5.1 for the questions on Data Volume should have been lower, thus lowering the overall suitability. Secondly, the application was not as transformable as previously thought. This was also due to the original model not taking the application maturity level into account.
In order to improve the 4VATT model, six different questions were added, divided over four different categories. These questions are all answered with a Likert Scale, ranging from Very

high to Very low (or vice versa). The new questions are listed below, with their respective keys showing which category the questions are added to.

- **(Vol.3)** What is the amount of features within the data an average application in this domain receives as an input? (Very high - Very low)
- **(Vol.4)** What is the amount of time-based data that an average application within this domain currently uses? (Very low - Very high)
- **(Vel.3)** What is the required speed that the data needs to travel through an application in this domain? (Very high - Very low)
- **(Ver.3)** What is the amount of data preparation needed in order to feed application data into a DRL solution for an average application in this domain? (Very high - Very low)
- **(Trf.4)** What is the application maturity level of an average application in this domain? (Very low - Very high)
- **(Trf.5)** What is the level of clarity with respect to determining what is a 'good' outcome of an average application within this domain? (Very high - Very low)

These questions, which have been added to the 4VATT model, are reflected in the improved version of the 4VATT Suitability Model as presented in Appendix E. After the questions have been added, the suitability percentage for the prescriptive healthcare domain was calculated again. The result was an expected suitability of **53%**. This is lower than the results from the original 4VATT Model, which showed an expected suitability of 66%. The distribution for the new suitability percentage is shown in Table 20. Furthermore, a comparison between the 'old' and 'new' 4VATT Suitability Models is shown in Figure 42.

Table 20
Score distributions for the suitability of the prescriptive healthcare domain, using the improved 4VATT Suitability Model

Category	Prescriptive Healthcare	
	Score	Suitability
Volume	10	50.00%
Variety	8	80.00%
Velocity	10	66.67%
Veracity	11	55.00%
Accountability	7	46.67%
Transparency	6	40.00%
Transformability	12	48.00%
Total	64	53.33%

As shown in Figure 42, the prescriptive healthcare domain scores a lower Domain Suitability Score in nearly every category of the model when scored with the new 4VATT Model in comparison to when scored with the old 4VATT Model, save for the Variety and Velocity. This is in line with the results of sub questions (Q3) and (Q4), which showed that the

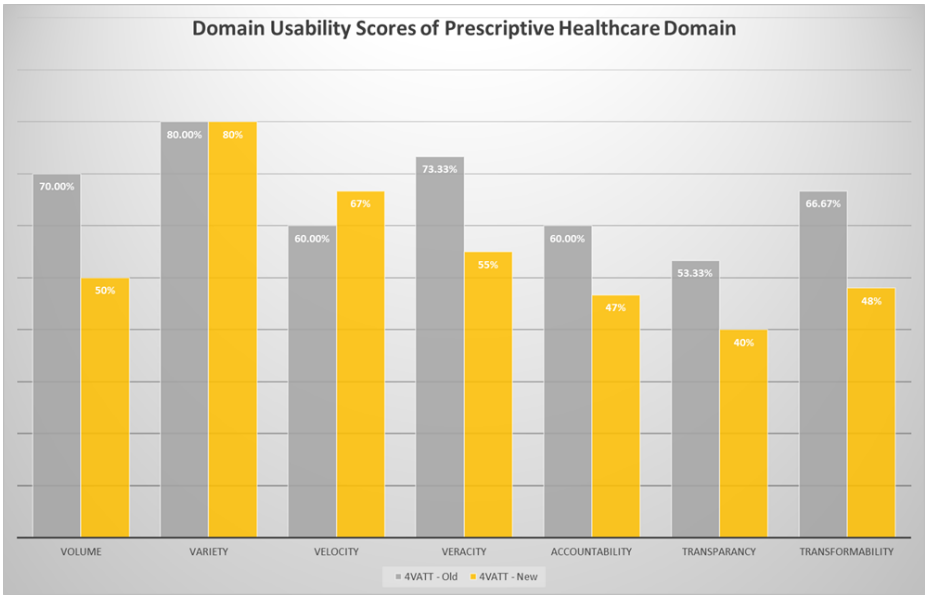


Figure 42. Domain Suitability Scores for the Prescriptive Healthcare domain, using both the old 4VATT Model from Section 4.2, and the improved 4VATT model

implementation principles did not lead to a significant improvement in effectiveness and efficiency respectively.

9 Conclusion

This section will summarize all of the findings from this thesis in order to answer the sub research questions introduced in Section 1.3. This will in turn answer the main research question as introduced in Section 1.2.

In order to determine whether a new domain would be suitable for the usage of Deep Reinforcement Learning, the 4VATT Suitability Model can be used. This model can estimate the suitability of (an application in) a domain by means of percentage. Initial results showed that for the prescriptive healthcare domain, the estimated suitability percentage is 66%. In order to transform an application in a new domain in such a way that it represents a game with rules, input, and output, the CRISP-DRL framework can be used. This framework will act as a guidance to support the complete implementation process, akin to CRISP-DM. After implementing the combined principles of Deep Learning and Reinforcement Learning into the STRIP Assistant, which has been dubbed as *STRIPAI*, the results show that these principles do not lead to a significant improvement in effectiveness. Additionally, with the use of computational complexity theory, it can be shown that these principles also do not lead to a significant improvement in efficiency. As a result, the insights from this implementation have been used as input to improve the 4VATT Suitability Model. The model has seen questions added to multiple categories. Calculating the suitability for the prescriptive healthcare domain again now shows a percentage of 53%.

In conclusion, this thesis has shown how determine which domains would be suitable for the usage of Deep Reinforcement Learning, and how to translate (an application in) a new domain in order to implement the combined principles of Deep Learning and Reinforcement Learning into it. However, this does not lead to a significant increase in usability.

10 Discussion

This chapter will aim to put the results from Chapter 8 into perspective. This chapter will show what possible influences could have been in place on the performed experiment, and how future research can help to further improve these results. This chapter will discuss the limitations of the research, from both a technical and theoretical standpoint. These research limitations will be introduced in Section 10.1. Section 10.2 will elaborate on how and where this research can be improved and continued, taking into account the limitations of the research as discussed in Section 10.1.

10.1 Research Limitations

This section will discuss all of the limitations to this research, which will help put the results from Chapter 8 and the conclusion from Chapter 9 into perspective. This section will introduce a multitude of possible factors, grouped into overarching themes, which will each be discussed in their separate subsection.
The research did not turn out as expected. Applying DRL principles does not seem to lead to significant improvements in the usability of the application it is applied in. This is due to a multitude of possible factors, which will be explained in the following paragraphs. Each factor is part of an overarching theme. These themes will be discussed in the next subsections.

10.1.1 Data Issues.

This subsection will discuss the limitations of this research with respect to the theme of data issues. Multiple issues arose during the execution of this research, which will be explained in the following paragraphs.
The first possible factor is the choice of class label. By aiming to replace the STRIP Assistants internal rule engine with a DRL-driven system, and thus choosing to predict the firing of STOPP and START rules, several decisions and aggregations had to be made with respect to data preparation. The data has been aggregated to a patient level, in order to predict the changes to medication (and thus the firing of STOPP and START rules). In addition to this, all of the different kinds of medication had to be aggregated into medication categories, and all of the possible complaints within the system to complaint categories (or complications). This is connected with another factor, namely the factor of data dimensionality, which will be discussed in a further paragraph. Lastly all the separate STOPP and START criteria are aggregated into the four class labels the implementation has been trained on.
The aggregation of class labels (and connected to this, the aggregation of medications and complaints) was done in an attempt to reduce the dimensionality of the DRL implementation. The original dataset contains data of only 512 patients. Every patient can have an arbitrary amount of medications, and complications, and can trigger one of 60 STOPP/START criteria. If data would not be aggregated, the data that needed to be entered into the network would have contained many more columns than it would have rows. Also, up to 60 different outputs could have been possible. With this dimensionality, it is not possible to train a well-performing DRL implementation. By aggregating the data, some level of detail is lost, but this is compensated by a better performing network.

The issue of dimensionality is closely tied to the issue of data volume. As stated, the dataset only contained data for 512 distinct patients. A part of this data can not be used for training the implementation, as this is used for validation. In the case of this thesis, this split was 80% training data, 20% validation data. This means that only 408 data points were used to train the networks. However, the amount of columns within the dataset was not proportionate to the amount of rows that was available for training or testing. As such, it was very hard to train a network both performed well enough, yet also was effective enough in predicting whether STOPP/START rules should fire.
Lastly, both the training data and the testing data contained a class imbalance. Only about 1/6th of the class labels indicated that no action needed to be taken with respect to firing STOPP/START rules. In order to combat this, undersampling was performed in order to alleviate the class imbalance within the dataset, albeit at the cost of the size of the training dataset. However, due to the nature of the dataset, this class imbalance is to be expected. All of the patients are situated within the field of polypharmacy, which means that all of the patients take substantial amounts of different kinds of medicine. It is highly expected that medication either has to be started, stopped, or changed for these patients.

10.1.2 Implementation Issues.

This subsection will discuss the limitations of this research with respect to the theme of implementation issues. Multiple issues arose during the execution of this research, which will be explained in the following paragraphs.
As explained in Section 10.1.1, the data dimensionality issues meant that data on complaints and medications had to be aggregated. After aggregations, the data still contained more columns than rows, but its dimensionality was reduced down considerably. In order to feed the complications and medications into the implementation, further preprocessing of this data was required. Our used implementation was not able to handle inputs with variable length. This means that for the patient medication, the algorithm could not handle a list of medications which had a length of 10 at one time, and then a length of 7 the next time. In order to combat this, a multi-hot encoded (MHE) vector for both complaints and medications needed to be constructed in order to properly feed the data into the network. These vectors were highly sparse. This means that the vector consisted mostly of zeroes, with non-zero values occurring very sparsely. The MHE vectors could also only be constructed from the complete set of medication *categories*. The implementation also did not handle mixed inputs, meaning that a combination of integers and lists is not possible to enter into a DRL implementation.
In order to then further reduce the data dimensionality, all unused compaints and medication categories were pruned from the list before constructing MHEs. This meant that the complications that did not occur within the dataset were not considered as possible when constructing the MHE vectors, making the vectors themselves less sparse. While this did show an increase in network effectiveness, this means that whenever a completely new category is introduced in the network (i.e a category that was not present in the training or test data), the whole network needs to be retrained. Retraining the network is something that should be done in order to keep fine-tuning the network with more data. This in turn lowered the efficiency of the DRL implementation, as adding more patient data to this system will inherently cause the STRIPAI to encounter ‘unknown’ complications or

medication categories.
Due to the lack of a clear reward function, the lack of simulation, and no goal, the current version of the STRIPAI is not driven by a self-learning agent. As stated in Section 3.2, these are criteria for implementing this style of algorithm into an application. The current state of the application (including the data that resides in it) is not far enough for a self-learning agent. For the STRIP Assistant, there is a goal, namely to minimize the amount of unnecessary medication being taken by patients. However, it is not defined what the criteria are to meet this goal, and what is done after this goal is achieved. Is the goal then to optimize even further, all the way to zero? What if some medication is strictly needed in order for patients to survive, how does the goal function cope with this? Furthermore, this goal can be different for each patient group, or even for each patient. In order to set up a realistic reward function, one needs to know whether the predicted set of medication is 'good'. Is it better or worse to remove two medicine categories and add one, instead of solely removing one? This is something that is also heavily dependent on the patients current set of medication, complaints, and its medical data.
As of now, the current version of the STRIPAI can only predict whether a STOPP and/or START rule should fire. However, because this is aggregated, it can not predict *which* STOPP/START rule should fire. With the current data volume, it is not possible to train a network that can reliably predict which STOPP/START rule should fire. However, when more data becomes available, this might be possible.
Lastly, each patient has its own set of restrictions with respect to the medication (categories) they can or can not take. This can be due to allergies, or religious beliefs. Although this is also not accounted for within the original STRIP Assistant, it is an important factor that may influence the possibilities of implementing DRL in the future, when it is done to completely replace the General Practitioner instead of replacing a rule engine.

10.1.3 4VATT Suitability Model.

This subsection will discuss the limitations of this research with respect to the 4VATT Suitability Model. Multiple issues arose during the execution of this research, which will be explained in the following paragraphs.
Initally, the 4VATT Suitability Model was developed as a way to determine whether (applications in) a domain would be well suited for the implementation of Deep Reinforcement Learning principles. The suitability calculated for the prescriptive healthcare domain was initially calculated with the first version of the 4VATT Suitability Model. However, the result turned out to be a too high estimation of the suitability of the domain. The model did not have enough focus on the data volume, data quality, and application transformability. As such, the improved version of the 4VATT Suitability Model has more questions on these topics, in order to give a better insight on the suitability.
The 4VATT Suitability Model is based on an informal exploration of the literature on implementing principles of Deep Learning and Reinforcement Learning into applications. However, outside of the refinement made to the model based on the results from implementing these principles into the STRIP Assistant, no validation has been done. As a result, it might be possible that critical factors are missing from the model, as they have not been identified yet.
Because no validation on the model has been done as of yet, it is not possible to say what

results from the 4VATT Suitability Model can be classified as 'good' or 'satisfactory'. For example, the prescriptive healthcare domain scored 66% while using the first version of the model. However, it was unknown whether this was actually deemed acceptable enough to proceed.
The questions in the model are all scored on a Likert Scale of 1 to 5. All questions within the model are equal, meaning that no weights are applied to any question, or question category. This was done as to prevent the domain covered in this research to influence the weight choices for the questions within the model. The model does not contain any control questions, which might be of influence on the results of the model within this research.
From this research, it has become apparent that the data volume, domain transparency, and domain transformability are the most important factors of the 4VATT Suitability Model. However, as the questions within the model are all scored equal, relatively bad scores within these categories can be compensated with good scores in the other categories. While weights were not considered for this research, it is a possibility to look into adding *knock-out criteria*, meaning that a minimum score needs to be achieved for certain categories in order to be deemed 'suitable' enough to go ahead with implementing DRL.
The 4VATT Suitability Model has been developed in 2018, when DRL has not been implemented widespread within applications outside of the gaming domain. Within the medical domain, the retinal imaging solution as described by Poplin et al. (2018) is currently the only well-described implementation of Deep Reinforcement Learning. However, as stated by Gartner, both techniques will reach the plateau of productivity within 10 years. After that, technological advancements within these two fields could have a severe impact on the results of this model. For example, if a way is found to give insight into the exact decision making process of a neural network, and this technique is standardized, then the questions on domain transparency will become irrelevant.

10.2 Future Work

From this thesis and the artifacts it presents, some takings can be made. This thesis introduced the 4VATT Suitability Model, a way to determine whether a new domain would be suitable for the implementation of Deep Reinforcement Learning principles. Within this thesis, an improvement has been made to this artifact, based on the results from an experiment to implement these principles into an application in the prescriptive healthcare domain. For future research regarding this model, multiple paths can be taken. The first path would be to further look into developing 4VATT. By adding more categories, or more questions to the existing categories, a further deepening of the model can be mode. Another possible solution is to add weights to the existing categories. This has not been done in the current version of the model, as to not bias the model towards the prescriptive healthcare domain. Lastly, as described, there is a possibility of adding 'knock-out' criteria to the model.
Another path to take regarding 4VATT is to test its' validity against other new domains. Within the context of this thesis, only the prescriptive healthcare domain has been tested. For further improving the model, and to put the results for the prescriptive healthcare domain into perspective, a more widespread usage of the 4VATT model is needed. When the model is used to determine the suitability of other domains, the accuracy should increase as more tweaks are made to the model.

This thesis also introduced the CRISP-DRL framework. While this is named similarly to CRISP-DM, it is by no means a standard framework yet. As future research, one can look into standardizing this framework, or look into maybe adapting the CRISP-DM framework itself.
Lastly, this thesis introduced the STRIPAI, which is the implementation of DRL principles into the STRIP Assistant. While this implementation did not lead to a significant increase in usability, it might me interesting for future research to try this experiment again at a later date. Once both artifacts have matured, the STRIP Assistant has reached a higher application maturity level, technology has advanced, and more patient data is present, the experiment might give better results than it did today.
While this research did not show an improvement in usability, some additional research can already be done on the STRIP Assistant. First, a similar implementation can be done on a smaller or larger scale by tweaking the unit of analysis. For this research, the unit of analysis was not in line with the possibilities in this domain.
Another possibility to help implement the principles of DRL into the STRIP Assistant is by combining the foundation provided by this research with the work of Huibers et al. (2019), which covers an improved version of the STOPP/START criteria, converted to XML. These new criteria can be applied to any software application, and could serve as a fall-back for the STRIPAI.

11 References

Allis, L. V. (1994). *Searching for solutions in games and artificial intelligence.* Rijksuniversiteit Limburg.

Azevedo, A. I. R. L., & Santos, M. F. (2008). KDD, SEMMA and CRISP-DM: a parallel overview. *IADS-DM*.

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. *arXiv preprint arXiv:1704.05796*.

Bellman, R. (2013). *Dynamic programming.* Courier Corporation.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, *2*(1), 1–127.

Berman, J. J. (2013). *Principles of big data: preparing, sharing, and analyzing complex information.* Newnes.

Bertsekas, D. P. (2008). Neuro-dynamic programming. In *Encyclopedia of optimization* (pp. 2555–2560). Springer.

Bordes, A., Glorot, X., Weston, J., & Bengio, Y. (2012). Joint learning of words and meaning representations for open-text semantic parsing. In *Artificial intelligence and statistics* (pp. 127–135).

Campanelli, C. M. (2012). American geriatrics society updated beers criteria for potentially inappropriate medication use in older adults: the american geriatrics society 2012 beers criteria update expert panel. *Journal of the American Geriatrics Society*, *60*(4), 616.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000, August). *CRISP-DM 1.0 Step-by-step data mining guide* (Tech. Rep.). The CRISP-DM consortium. Retrieved from `http://www.crisp-dm.org/CRISPWP-0800.pdf`

Chen, G. (2016). A gentle tutorial of recurrent neural network with error backpropagation. *arXiv preprint arXiv:1610.02583*.

Chen, X.-W., & Lin, X. (2014). Big data deep learning: challenges and perspectives. *IEEE access*, *2*, 514–525.

Claxton, A. J., Cramer, J., & Pierce, C. (2001). A systematic review of the associations between dose regimens and medication compliance. *Clinical therapeutics*, *23*(8), 1296–1310.

Council, N. R. (2013). *Frontiers in massive data analysis.* National Academies Press.

Dechter, R. (1986). *Learning while searching in constraint-satisfaction problems.* University of California, Computer Science Department, Cognitive Systems Laboratory.

Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197–387.

Doorhof, D. (2018). *Using reinforcement learning to improve clinical decision making in neonatal care* (Unpublished master's thesis). Utrecht University.

Drenth-van Maanen, A. C., van Marum, R. J., Knol, W., van der Linden, C. M., & Jansen, P. A. (2009). Prescribing optimization method for improving prescribing in elderly patients receiving polypharmacy. *Drugs & aging*, *26*(8), 687–701.

Emmens, T., & Phippen, A. (2010). Evaluating online safety programs. *Harvard Berkman Center for Internet and Society.[23 July 2011]*.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, *39*(11), 27–34.

Gallagher, P., Ryan, C., Byrne, S., Kennedy, J., & O'Mahony, D. (2008). Stopp (screening tool of older person's prescriptions) and start (screening tool to alert doctors to right treatment). consensus validation. *International journal of clinical pharmacology and therapeutics*, *46*(2), 72–83.

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*(2), 137–144.

Gantz, J., & Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and

biggest growth in the far east. *IDC iView: IDC Analyze the future*, *2007*(2012), 1–16.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* (Vol. 1).

Goodman, B., & Flaxman, S. (2016). European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*.

Graham, B. (2014). Fractional max-pooling. *arXiv preprint arXiv:1412.6071*.

Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*.

Hanlon, J. T., Schmader, K. E., Samsa, G. P., Weinberger, M., Uttech, K. M., Lewis, I. K., ... Feussner, J. R. (1992). A method for assessing drug therapy appropriateness. *Journal of clinical epidemiology*, *45*(10), 1045–1051.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning* (pp. 485–585). Springer.

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems* (pp. 9–22). Springer.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105. Retrieved from `http://www.jstor.org/stable/25148625`

Hilleli, B., & El-Yaniv, R. (2018). Toward deep reinforcement learning without a simulator: An autonomous steering example. In *Thirty-second aaai conference on artificial intelligence.*

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huibers, C. J., Sallevelt, B. T., de Groot, D. A., Boer, M. J., van Campen, J. P., Davids, C. J., ... Meulendijk, M. C. (2019). Conversion of stopp/start version 2 into coded algorithms for software implementation: a multidisciplinary consensus procedure. *International Journal of Medical Informatics*, *125*, 110-117. Retrieved from `https://github.com/MichielCM/stoppstart` doi: 10.1016/j.ijmedinf.2018.12.010

IBM. (2018). *Extracting business value from the 4 v's of big data.* Author. Retrieved from `https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data` (Accessed October 4th, 2018)

International Organization for Standardization (ISO). (2011). ISO/IEC 25010:2011:. *Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models*. Retrieved from `http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733`

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*(4), 364–378.

Jansen, P. A., & Brouwers, J. R. (2012). Clinical pharmacology in old persons. *Scientifica*, *2012*.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, *6*(5), 429–449.

Jobse, L., Mulder, M., ter Borgh, J., & Grundmeijer, H. (2009). Polyfarmacie. *Huisarts en wetenschap*, *52*(12), 599–602.

Kepner, J., Gadepally, V., Michaleas, P., Schear, N., Varia, M., Yerukhimovich, A., & Cunningham, R. K. (2014). Computing on masked data: a high performance method for improving big data veracity. In *High performance extreme computing conference (hpec), 2014 ieee* (pp. 1–6).

Knemeijer, D. (2018, February 5). *Valeas - early warning system for the critically ill.* [YouTube video]. Retrieved from `https://www.youtube.com/watch?v=3qZIh3TAnhE` (Accessed October 31st, 2018)

Knol, W., Verduijn, M. M., Lelie-van der Zande, A., van Marum, R. J., Brouwers, J., van der Cammen, T. J., ... Jansen, P. A. (2015). Onjuist geneesmiddelgebruik bij ouderen opsporen.

de herziene stopp-en start-criteria. *Nederlands Tijdschrift voor Geneeskunde*, *159*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Krüger, F. (2016). *Activity, context, and plan recognition with computational causal behaviour models* (Unpublished doctoral dissertation). University.

KS, D., & Kamath, A. (2017). Survey on techniques of data mining and its applications. *International Journal of Emerging Research in Management & Technology*, *6*(2), 198–201.

Kuijpers, M. A., Van Marum, R. J., Egberts, A. C., Jansen, P. A., & OLDY (OLd people Drugs & dYsregulations) Study Group. (2008). Relationship between polypharmacy and underprescribing. *British journal of clinical pharmacology*, *65*(1), 130–133.

Kuzuya, M., Endo, H., Umegaki, H., Nakao, M., Niwa, T., Kumagai, T., ... Iguchi, A. (2000). Factors influencing noncompliance with medication regimens in the elderly. *Nihon Ronen Igakkai zasshi. Japanese journal of geriatrics*, *37*(5), 363–370.

Lample, G., & Chaplot, D. S. (2017). Playing fps games with deep reinforcement learning. In *Aaai* (pp. 2140–2146).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Leendertse, A. J., de Koning, F. H., Goudswaard, A. N., Jonkhoff, A. R., van den Bogert, S. C., de Gier, H. J., ... van den Bemt, P. M. (2011). Preventing hospital admissions by reviewing medication (pharm) in primary care: design of the cluster randomised, controlled, multi-centre pharm-study. *BMC health services research*, *11*(1), 4.

Leendertse, A. J., Egberts, A. C., Stoker, L. J., & van den Bemt, P. M. (2008). Frequency of and risk factors for preventable medication-related hospital admissions in the netherlands. *Archives of internal medicine*, *168*(17), 1890–1896.

Levinovitz, A. (2014). The mystery of go, the ancient game that computers still can't win. *Wired Magazine*.

Marbán, Ó., Mariscal, G., & Segovia, J. (2009). A data mining & knowledge discovery process model. In *Data mining and knowledge discovery in real life applications.* InTech.

Mariscal, G., Marban, O., & Fernandez, C. (2010). A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*, *25*(2), 137–166.

Matiisen, T. (2015). Demystifying deep reinforcement learning. *Computational Neuroscience LAB*.

Melo, F. S. (2001). Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, 1–4.

Meulendijk, M. C., Spruit, M. R., Drenth-van Maanen, A. C., Numans, M. E., Brinkkemper, S., Jansen, P. A., & Knol, W. (2015). Computerized decision support improves medication review effectiveness: an experiment evaluating the strip assistant's usability. *Drugs & aging*, *32*(6), 495–503.

Meulendijk, M. C., Spruit, M. R., Willeboordse, F., Numans, M. E., Brinkkemper, S., Knol, W., ... Askari, M. (2016). Efficiency of clinical decision support systems improves with experience. *Journal of medical systems*, *40*(4), 76.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association.*

Miotto, R., Wang, F., Wang, S., Jiang, X., & Dudley, J. T. (2017). Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1.

Nature. (2007, Oct). A matter of trust [editorial]. *Nature*, *449*(7163), 637âĂŞ638. Retrieved from `https://doi.org/10.1038/449637b` (Accessed October 9th, 2018)

NHG, N. H. G., & Verzorging, L. E. V. (2012). Multidisciplinaire richtlijn polyfarmacie bij ouderen. (Accessed November 16th, 2018)

Oracle. (2015). *Servlet (java(tm) ee 7 specification apis.* Author. Retrieved from `https://docs.oracle.com/javaee/7/api/javax/servlet/Servlet.html` (Accessed January 8th, 2019)

Poplin, R., Varadarajan, A. V., Blumer, K., Liu, Y., McConnell, M. V., Corrado, G. S., ... Webster, D. R. (2018, 2). Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, 1–7. Retrieved from `http:https://doi.org/10.1038/s41551-018-0195-0` doi: 10.1038/s41551-018-0195-0

Prema, P., & Ramadoss, B. (2008). A classification scheme for game input and output. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, *2*(6), 1870–1876.

Pritzker, P., & May, W. (2015). Nist big data interoperability framework (nbdif): Volume 1: Definitions. *NIST Special Publication*, *1500*(1).

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons.

Raghupathi, W., & Raghupathi, V. (2014). Big data analytics in healthcare: promise and potential. *Health information science and systems*, *2*(1), 3.

Rao, A., Voyles, J., & Ramchandani, P. (2018, Jan). *Top 10 artificial intelligence (ai) technology trends for 2018.* PwC. Retrieved from `http://usblogs.pwc.com/emerging-technology/top-10-ai-tech-trends-for-2018/` (Accessed July 7th, 2018)

Rollason, V., & Vogt, N. (2003). Reduction of polypharmacy in the elderly. *Drugs & aging*, *20*(11), 817–832.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533.

Rummery, G. A., & Niranjan, M. (1994). *On-line q-learning using connectionist systems* (Vol. 37). University of Cambridge, Department of Engineering Cambridge, England.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (2003). *Artificial intelligence: a modern approach* (Vol. 2) (No. 9). Prentice hall Upper Saddle River.

Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association.*

Sergi, G., De Rui, M., Sarti, S., & Manzato, E. (2011). Polypharmacy in the elderly. *Drugs & aging*, *28*(7), 509–518.

SethBling. (2015, June 13). *MarI/O - machine learning for video games.* [YouTube video]. Retrieved from `https://www.youtube.com/watch?v=qv6UVOQ0F44` (Accessed October 10th, 2018)

Shibata, K., & Okabe, Y. (1997). Reinforcement learning when visual sensory signals are directly given as inputs. In *Neural networks, 1997., international conference on* (Vol. 3, pp. 1716–1720).

Silver, D., & Hassabis, D. (2016). Alphago: Mastering the ancient game of go with machine learning. *Google AI Research Blog*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Bolton, A. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354.

Sloane, P. D., Gruber-Baldini, A. L., Zimmerman, S., Roth, M., Watson, L., Boustani, M., ... Hebel, J. R. (2004). Medication undertreatment in assisted living settings. *Archives of Internal Medicine*, *164*(18), 2031–2037.

Socher, R., Huang, E. H., Pennin, J., Manning, C. D., & Ng, A. Y. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems* (pp. 801–809).

Spruit, M., & Jagesar, R. (2016). Power to the people! In *Proceedings of the international joint conference on knowledge discovery, knowledge engineering and knowledge management* (pp. 400–406).

Spruit, M., & Lytras, M. (2018). *Applied data science in patient-centric healthcare.* Elsevier.

Steinman, M. A., Seth Landefeld, C., Rosenthal, G. E., Berthenthal, D., Sen, S., & Kaboli, P. J. (2006). Polypharmacy and prescribing quality in older people. *Journal of the American Geriatrics Society*, *54*(10), 1516–1523.

Sutton, R. S., Barto, A. G., & Bach, F. (1998). *Reinforcement learning: An introduction.* MIT press.

Theodoridis, S., Koutroumbas, K., et al. (2008). Pattern recognition. *IEEE Transactions on Neural Networks*, *19*(2), 376.

Thorp, E., & Walden, W. E. (1972). A computer assisted study of go on m× n boards. *Information Sciences*, *4*(1), 1–33.

Tuyls, K., Perolat, J., Lanctot, M., Ostrovski, G., Savani, R., Leibo, J. Z., ... Legg, S. (2018). Symmetric decomposition of asymmetric games. *Scientific reports*, *8*(1), 1015.

Valk, M. (2018). *Interpretable recurrent neural networks for heart failure re-hospitalization prediction* (Unpublished master's thesis). Utrecht University.

Van Den Herik, H. J., Uiterwijk, J. W., & Van Rijswijck, J. (2002). Games solved: Now and in the future. *Artificial Intelligence*, *134*(1-2), 277–311.

Van Gerven, M. (2017). Computational foundations of natural intelligence. *Frontiers in computational neuroscience*, *11*, 112.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Aaai* (Vol. 2, p. 5).

van Otterlo, M., & Wiering, M. (2012). Reinforcement learning and markov decision processes. In *Reinforcement learning* (pp. 3–42). Springer.

Verdoorn, S., Kwint, H.-F., Faber, A., Gussekloo, J., & Bouvy, M. (2016). Stopp-start-criteria kunnen medicatiebeoordeling niet vervangen. *Huisarts en wetenschap*, *59*(10), 439–442.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (pp. 1995–2003).

Weng, J., Ahuja, N., & Huang, T. S. (1992). Cresceptron: a self-organizing neural network which grows adaptively. In *Neural networks, 1992. ijcnn., international joint conference on* (Vol. 1, pp. 576–581).

Wieringa, R. J. (2014). *Design science methodology.* Springer.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering* (p. 38).

Wright, R. M., Sloane, R., Pieper, C. F., Ruby-Scelsi, C., Twersky, J., Schmader, K. E., & Hanlon,

J. T. (2009). Underuse of indicated medications among physically frail older us veterans at the time of hospital discharge: results of a cross-sectional analysis of data from the geriatric evaluation and management drug study. *The American journal of geriatric pharmacotherapy*, *7*(5), 271.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zell, A. (1994). *Simulation neuronaler netze* (Vol. 1). Addison-Wesley Bonn.

Zhu, X. (2011). Semi-supervised learning. In *Encyclopedia of machine learning* (pp. 892–897). Springer.

A 4VATT Question List

This appendix contains all of the questions for the 4VATT Suitability Model (see Section 4.2). The questions are sorted by their respective factors in the leftmost column. Identification keys are noted prior the questions, in order to be able to refer to specific questions in other appendices or from within the main body of this thesis. The other 5 columns indicate the answer on a five-point Likert Scale and its corresponding score for that question.
When spoken about an application, there should be referred to a chosen example application within the domain the 4VATT Suitability Model is filled in for. When spoken about an implementation, there should be referred to the same application, but with Deep Reinforcement Learning built into it.

Question	**Score**				
	1	**2**	**3**	**4**	**5**
Volume					
(Vol.1) What is the data volume the application in this domain is capable of handling?	Very high	High	Moderate	Low	Very low
(Vol.2) What is the data volume the application in this domain receives as input?	Very high	High	Moderate	Low	Very low
Variety					
(Var.1) What is the current level of data variety in an the application in this domain?	Very high	High	Moderate	Low	Very low
(Var.2) What is the expected level of change in data variety after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low
Velocity					
(Vel.1) What is the number of data sources the application receives data from?	Very high	High	Moderate	Low	Very low
(Vel.2) What is the speed of the data that enters the application in this domain?	Very high	High	Moderate	Low	Very low

Question	Score				
	1	**2**	**3**	**4**	**5**
Veracity					
(Ver.1) What is the average data quality level within this domain?	Very low	Low	Moderate	High	Very high
(Ver.2) What is the level of variety in data quality the application within this domain receives as input?	Very high	High	Moderate	Low	Very low
(Ver.3) How major is it that the data fed into the application is completely correct (i.e. fault-free)?	Very major	Major	Moderate	Minor	Very minor
Accountability					
(Acc.1) What is the expected seriousness level of an accountability issue after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low
(Acc.2) What is the expected amount of people involved in an accountability issue after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low
(Acc.3) What is the total expected amount of accountability issues after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low

Question	Score				
	1	**2**	**3**	**4**	**5**
Transparency					
(Trp.1) What amount of governance of the application in this domain is needed or will be used?	Very high	High	Moderate	Low	Very low
(Trp.2) How major is it to know the exact decision making process for the application in this domain?	Very major	Major	Moderate	Minor	Very minor
(Trp.3) What is the amount of sensitive data the application will be dealing with?	Very high	High	Moderate	Low	Very low
Transformability					
(Trf.1) What is the amount of similarity in structure between the implementation in this domain, and an implementation in the gaming domain?	Very low	Low	Moderate	High	Very high
(Trf.2) What is the expected difficulty level of transforming the application in this domain into one having roughly the same structure as a game?	Very high	High	Moderate	Low	Very low
(Trf.3) What is the level of connectedness of the application to other applications in this domain? (e.g what is the number of applications this application is embedded in, or connected to?)	Very high	High	Moderate	Low	Very low

B Answered Questions for Prescriptive Healthcare Domain

This appendix contains the answered 4VATT Question List from Appendix A, concerning the prescriptive healthcare domain. The first column of the table contains the question ID, which can be used to look up the question itself. The next two tables contain the answer to the question, and its respective score. The last column gives an explanation on how the given answer is conceived.

Question ID	Answer	Score	Explanation
Volume			
(Vol.1)	Moderate	3	While not being able to handle excessive amounts of data, applications within the prescriptive healthcare domain can handle a regular amount of data
(Vol.2)	Low	4	The amount of data the STRIP Assistant handles is low, as only during visits to a GP more data is passed to the system
Variety			
(Var.1)	Low	4	The current data variety is low; only patient data on medication, treatments, complaints, and medical measurements is stored. The first three are categorical data, the last one is continuous data
(Var.2)	Low	4	The data that is stored and handled in the system will not change after an implementation of DRL principles
Velocity			
(Vel.1)	High	2	The application receives input from multiple sources; amongst those are medical measurement devices, START rules, STOPP rules, and different kind of contra-indication rules
(Vel.2)	Low	4	The data enters the application periodically, namely every time a patient visits a GP

Question ID	Answer	Score	Explanation
Veracity			
(Ver.1)	Very high	5	The data is generated by carefully constructed START and STOPP rules, by reliable medical instruments, or entered by a GP with a high level of medical knowledge
(Ver.2)	Low	4	While a perfect level of quality cannot be guaranteed, there is little variety in data quality
(Ver.3)	Major	2	Algorithmic decision making cannot completely guarantee a foolproof decision making process. As such, a set of decision made by an algorithm will contain false positives and/or false negatives. Within the medical field, it is vital to minimize these false results
Accountability			
(Acc.1)	High	2	Whenever something goes wrong in this domain, someone needs to be held accountable. Usually, when something goes wrong in this domain, it ends in a patient dying a preventable death
(Acc.2)	Low	4	In this domain, the amount of people that can be held accountable is usually pretty slim, as only a select group of people will prescribe medicine to patients
(Acc.3)	Moderate	3	Ideally, the algorithm will make a perfect prediction every time. However, as this is not possible, GPs will not be replaced by algorithms, but instead will be supported by them

Question ID	Answer	Score	Explanation
Transparency			
(Trp.1)	High	2	If a decision is made by a GP, it needs to be clear exactly *why* this decision is made. Likewise, if an algorithm is in charge for this decision, it needs to be known why a certain decision has been made, as this could be a vital decision that could impact someone's life
(Trp.2)	Moderate	3	For most decisions, it is not major to know exactly how a decision is made. However, in some instances, it is very major to know the exact decision making process (i.e. why is certain medication chosen in favour of other medication)
(Trp.3)	Moderate	3	The implementation will be dealing with medical measurements of a patient, as well as complaints and medications of a patient. While this is data about a patient, it is not highly sensitive data which can be used to uniquely identify a single patient
Transformability			
(Trf.1)	Low	2	Most implementations in the gaming domain offer both a *state* and a *goal*. Usually, this goal is to get a high score, or a low time. However, the domain of the application lacks a clear goal, and has no overall 'state' in which the application resides
(Trf.2)	Low	4	It is expected that the amount effort to reform the application into a game-like structure is low.
(Trf.3)	Low	4	The application itself functions as a standalone application. As such, it is connected to little to no other applications. The main connections this application has, are to databases with patient data and rules

C DRL Implementation

```python
import mysql.connector
from mysql.connector import errorcode

### Import the config file for the STRIP Assistant Database
from .python_mysql_dbconfig import read_db_config

### Get the config for the local MySQL instance ###
config = read_db_config()

def connect_to_db():
    try:
        cnx = mysql.connector.connect(**config)
    except mysql.connector.Error as err:
        if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
            print('Something is wrong with your username or password')
        elif err.errno == errorcode.ER_BAD_DB_ERROR:
            print('Database error, it seems that your database is non-existent')
            print(err)
        else:
            print(err)
    else:
        return cnx

def query_db(cnx, query):
    cursor = cnx.cursor()
    cursor.execute(query)
    result = cursor.fetchall()
    cursor.close()
    return result

def multi_query_db(cnx,queries):
    result = {}
    for key, query in queries.items():
        result[key] = query_db(cnx,query)
    return result
### End database connection classes ###
```

Figure 43. Python module for database connections between the STRIP Assistant Database and the DRL Implementation

```python
from strip import dbc as sdbc
from strip import MHE_vectorify as smhe
from queries import query_dict

import tensorflow as tf

import numpy as np
import pandas as pd

from sklearn.preprocessing import MultiLabelBinarizer
```

```
11
12  ### Get all the data from the database
13  cnx = sdbc.connect_to_db()
14  db = sdbc.multi_query_db(cnx, query_dict)
15
16  (patients, medications, complications, measurements, startstop, _,_,_) = db.values()
17  all_measurements = [x[0] for x in db["all_measurements"]]
18  all_medications = [x[0] for x in db["all_medications"]]
19  all_complications = [x[0] for x in db["all_complications"]]
20
21  cnx.close()
22
23  ### Empty (MHE)vectors for measurements and medications:
24  empty_medications = np.zeros(len(all_medications), dtype=int)
25  empty_measurements = np.zeros(len(all_measurements), dtype=int)
26
27  ### Empty vector for complications
28  mlb = MultiLabelBinarizer()
29  temp_comps = dict(smhe.listCombiner(complications))
30  df = pd.DataFrame(mlb.fit_transform(temp_comps.values()), columns = mlb.classes_, index =
    ↪  temp_comps.keys())
31  c_empty = list(np.zeros(len(np.array(df.iloc[[0]].values).flatten())))
32
33  ### Dictionary containing all medications per patient
34  medications_dict = smhe.createSTRIPdict(medications, all_medications)
35
36  ### Function to build a list of result data
37  def getResultDict(data):
38      resultdict = {}
39      for r in data:
40          (k, start, stopp) = r
41          if k not in resultdict:
42              if start > 1:
43                  resultdict[k] = ['START']
44              elif stopp > 1:
45                  resultdict[k] = ['STOPP']
46              elif start < 2 and stopp < 2:
47                  resultdict[k] = ['NONE']
48          if k in resultdict:
49              if start > 1:
50                  resultdict[k] = resultdict[k] + ['START']
51              elif stopp > 1:
52                  resultdict[k] = resultdict[k] + ['STOPP']
53              elif start < 2 and stopp < 2:
54                  resultdict[k] = resultdict[k] + ['NONE']
55
56      for k, v in resultdict.items():
57          u = list(set(v))
58          if 'START' in u and 'STOPP' in u:
59              resultdict[k] = 3
60          elif 'START' not in u and 'STOPP' in u:
61              resultdict[k] = 2
62          elif 'START' in u and 'STOPP' not in u:
63              resultdict[k] = 1
64          elif 'START' not in u and 'STOPP' not in u and 'NONE' in u:
```

```
            resultdict[k] = 0
        else:
            resultdict[k] = 0
    return resultdict

### Function to get all the measurement values
def getMeasurements(pID):
    if pID not in measurements[0]:
        return empty_measurements
    else:
        result = ()
        rowindex = 0
        patientMeasures = [row for row in measurements if row[0] == pID]
        for index, item in enumerate(all_measurements):
            if all_measurements[index] in patientMeasures[1]:
                result = result + (int(patientMeasures[rowindex][2]),)
                rowindex += 1
            else:
                result = result + (0,)
        return result

def generateIO():
    input, complications, medications, output = [],[],[],[]
    result_dict = getResultDict(startstop)
    for row in patients:
        (pID,_,_,_) = row
        input.append(list(row[1:]) + list(getMeasurements(pID)))
        try:
            c = np.asarray(df.loc[[pID]].values)
            c_flat = [item for sublist in c for item in sublist]
            complications.append(c_flat)
        except KeyError:
            complications.append(c_empty)
        if pID in medications_dict:
            medications.append(medications_dict[pID])
        else:
            medications.append(empty_medications)
        if pID in result_dict:
            output.append([result_dict[pID]])
        else:
            output.append([0])

    return np.asarray(input),np.asarray(complications), np.asarray(medications),
     ↪  np.asarray(output)
```

Figure 44. Python module for preprocessing the data entering the STRIP Assistant DRL Implementation

```
import numpy as np
def totuple(a):
    try:
        return tuple(totuple(i) for i in a)
```

```
    except TypeError:
        return a

def listCombiner(inputList):
    firstrow = 1
    previousPatient = 0
    targetList, totalList = [], []
    for row in inputList:
        pID, tID = row
        if firstrow == 1:
            targetList.append(tID)
            firstrow = 0
        else:
            if pID == previousPatient:
                targetList.append(tID)
            else:
                totalList.append((previousPatient, targetList))
                targetList = []
                targetList.append(tID)
        previousPatient = pID
    totalList.append((previousPatient, targetList))
    return totalList

### Function to transform a query into a multi hot encoded vector
def toMHEvector(query, totalSet):
    MHEvector = []
    for x in totalSet:
        if x in query:
            MHEvector.append(1)
        else:
            MHEvector.append(0)
    return MHEvector

def massMHEvector(input, superset):
    result = []
    vec = []
    for row in input:
        patientID, y = row
        vec = toMHEvector(y, superset)
        result.append((patientID, vec))
    return result

def createSTRIPdict(x, x_set):
    return dict(massMHEvector(listCombiner(x), x_set))
```

Figure 45. Python module aiding in preprocessing of the data entering the STRIP Assistant DRL Implementation

D Network Statistics

This appendix contains additional metrics for the networks that have been constructed in Chapter 7. While the main results for the networks are already presented in Section 7.4, these metrics are meant as supporting data for these results. The following sections will house the metrics and supporting tables for each subsection from Section 7.4.

D.1 Personalia Network - Unaggregated

Table 23
Confusion matrix of the validation results on the trained personalia network, showing the confusion between the zero-class and the remainder class

		Predicted		
		0	∼	Total
Actual	**0**	0	18	18
	∼	0	84	84
	Total	0	102	102

Table 24
Metrics for the confusion matrix shown in Table 23

Metric	**Formula**	**Value**
Accuracy	(TP + TN) / total	0.8235
Error rate	(FP + FN) / total	0.1765
Sensitivity	TP / (FN + TP)	0.0000
Specificity	TN / (TN + FP)	1.0000
Precision	TP / (FN + TP)	***Error***
False Positive Rate	FP / (TN + FP)	0.0000
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	***Error***

Table 25
Confusion matrix of the validation results on the trained personalia network

		Predicted				
		0	**1**	**2**	**3**	Total
Actual	**0**	0	0	0	18	18
	1	0	0	0	16	16
	2	0	0	0	1	1
	3	0	0	0	67	67
	Total	0	0	0	102	102

Table 26
Metrics for the confusion matrix shown in Table 29

Metric	**Acronym**	**Class 0**	**Class 1**	**Class 2**	**Class 3**
True Positive	TP	0	0	0	67
False Positive	FP	18	16	1	0
False Negative	FN	0	0	0	35
True Negative	TN	84	86	101	0
Metric	**Formula**	**Class 0**	**Class 1**	**Class 2**	**Class 3**
Accuracy	(TP + TN) / total	0.8235	0.8431	0.9902	0.6569
Error rate	(FP + FN) / total	0.1765	0.1569	0.0098	0.3431
Sensitivity	TP / (FN + TP)	***Error***	***Error***	***Error***	0.6569
Specificity	TN / (TN + FP)	1.0000	1.0000	1.0000	***Error***
Precision	TP / (FP + TP)	0.0000	0.0000	0.0000	1.0000
False Positive Rate	FP / (TN + FP)	0.1765	0.1569	0.0098	***Error***
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	***Error***	***Error***	***Error***	0.7929

D.2 Complications Network - Unaggregated

Table 27
Confusion matrix of the validation results on the trained complications network, showing the confusion between the zero-class and the remainder class

		Predicted		
		0	∼	Total
Actual	**0**	0	18	18
	∼	0	84	84
	Total	0	102	102

Table 28
Metrics for the confusion matrix shown in Table 27

Metric	**Formula**	**Value**
Accuracy	(TP + TN) / total	0.8235
Error rate	(FP + FN) / total	0.1765
Sensitivity	TP / (FN + TP)	0.0000
Specificity	TN / (TN + FP)	1.0000
Precision	TP / (FP + TP)	***Error***
False Positive Rate	FP / (TN + FP)	0.0000
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	***Error***

Table 29
Confusion matrix of the validation results on the trained complications network

		Predicted				
		0	**1**	**2**	**3**	Total
Actual	**0**	0	0	0	18	18
	1	0	0	0	16	16
	2	0	0	0	1	1
	3	0	0	0	67	67
	Total	0	0	0	102	102

Table 30
Metrics for the confusion matrix shown in Table 29

Metric	**Acronym**	**Class 0**	**Class 1**	**Class 2**	**Class 3**
True Positive	TP	0	0	0	67
False Positive	FP	18	16	1	0
False Negative	FN	0	0	0	35
True Negative	TN	84	86	101	0
Metric	**Formula**	**Class 0**	**Class 1**	**Class 2**	**Class 3**
Accuracy	(TP + TN) / total	0.8235	0.8431	0.9902	0.6569
Error rate	(FP + FN) / total	0.1765	0.1569	0.0098	0.3431
Sensitivity	TP / (FN + TP)	***Error***	***Error***	***Error***	0.6569
Specificity	TN / (TN + FP)	1.0000	1.0000	1.0000	***Error***
Precision	TP / (FP + TP)	0.0000	0.0000	0.0000	1.0000
False Positive Rate	FP / (TN + FP)	0.1765	0.1569	0.0098	***Error***
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	***Error***	***Error***	***Error***	0.7929

D.3 Medications Network - Unaggregated

Table 31
Confusion matrix of the validation results on the trained medications network, showing the confusion between the zero-class and the remainder class

		Predicted		
		0	∼	Total
Actual	**0**	8	10	18
	∼	6	78	84
	Total	14	88	102

Table 32
Metrics for the confusion matrix shown in Table 31

Metric	**Formula**	**Value**
Accuracy	(TP + TN) / total	0.8431
Error rate	(FP + FN) / total	0.1569
Sensitivity	TP / (FN + TP)	0.4444
Specificity	TN / (TN + FP)	0.9286
Precision	TP / (FN + TP)	0.5714
False Positive Rate	FP / (TN + FP)	0.0714
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	0.5000

Table 33
Confusion matrix of the validation results on the trained medications network

		Predicted				
		0	**1**	**2**	**3**	Total
Actual	**0**	8	0	0	10	18
	1	3	0	0	13	16
	2	0	0	0	1	1
	3	3	0	0	64	67
	Total	14	0	0	88	102

Table 34
Metrics for the confusion matrix shown in Table 33

Metric	**Acronym**	**Class 0**	**Class 1**	**Class 2**	**Class 3**
True Positive	TP	8	0	0	64
False Positive	FP	6	0	0	24
False Negative	FN	10	16	1	3
True Negative	TN	78	86	101	11
Metric	**Formula**	**Class 0**	**Class 1**	**Class 2**	**Class 3**
Accuracy	(TP + TN) / total	0.8431	0.8431	0.9902	0.7353
Error rate	(FP + FN) / total	0.1569	0.1569	0.0098	0.2647
Sensitivity	TP / (FN + TP)	0.4444	0.0000	0.0000	0.8533
Specificity	TN / (TN + FP)	0.9286	1.0000	1.0000	0.3143
Precision	TP / (FP + TP)	0.5714	***Error***	***Error***	0.7273
False Positive Rate	FP / (TN + FP)	0.0714	0.0000	1.0000	0.6857
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	0.5000	***Error***	***Error***	0.7853

D.4 Personalia Network - Aggregated

Table 35
Confusion matrix of the validation results on the trained personalia network, trained with aggregated data, showing the confusion between the two classes

		Predicted		
		0	**1**	Total
Actual	**0**	9	9	18
	1	1	83	84
	Total	10	92	102

Table 36
Metrics for the confusion matrix shown in Table 35

Metric	**Formula**	**Value**
Accuracy	(TP + TN) / total	0.9010
Error rate	(FP + FN) / total	0.0990
Sensitivity	TP / (FN + TP)	0.9000
Specificity	TN / (TN + FP)	0.9011
Precision	TP / (FN + TP)	0.5000
False Positive Rate	FP / (TN + FP)	0.0989
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	0.6429

D.5 Complications Network - Aggregated

Table 37
Confusion matrix of the validation results on the trained complications network, trained with aggregated data, showing the confusion between the two classes

		Predicted		
		0	**1**	Total
Actual	**0**	14	4	18
	1	21	63	84
	Total	35	77	102

Table 38
Metrics for the confusion matrix shown in Table 37

Metric	**Formula**	**Value**
Accuracy	(TP + TN) / total	0.7549
Error rate	(FP + FN) / total	0.2451
Sensitivity	TP / (FN + TP)	0.4000
Specificity	TN / (TN + FP)	0.9403
Precision	TP / (FN + TP)	0.7778
False Positive Rate	FP / (TN + FP)	0.0597
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	0.5283

D.6 Medications Network - Aggregated

Table 39
Confusion matrix of the validation results on the trained medications network, trained with aggregated data, showing the confusion between the two classes

		Predicted		
		0	**1**	Total
Actual	**0**	10	8	18
	1	29	55	84
	Total	39	63	102

Table 40
Metrics for the confusion matrix shown in Table 39

Metric	**Formula**	**Value**
Accuracy	(TP + TN) / total	0.6373
Error rate	(FP + FN) / total	0.3627
Sensitivity	TP / (FN + TP)	0.2564
Specificity	TN / (TN + FP)	0.8730
Precision	TP / (FN + TP)	0.5556
False Positive Rate	FP / (TN + FP)	0.1270
f1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$	0.3509

E Improved 4VATT Question List

This appendix contains an improved version of the 4VATT Suitability Model from Section 4.2. Like the version of the model presented in Appendix A, the questions are sorted by their respective factors in the leftmost column. Identification keys are noted prior the questions, in order to be able to refer to specific questions in other appendices or from within the main body of this thesis. The other 5 columns indicate the answer on a five-point Likert Scale and its corresponding score for that question.
When spoken about an application, there should be referred to a chosen example application within the domain the 4VATT Suitability Model is filled in for. When spoken about an implementation, there should be referred to the same application, but with Deep Reinforcement Learning built into it.

Question	**Score**				
	1	**2**	**3**	**4**	**5**
Volume					
(Vol.1) What is the data volume the application in this domain is capable of handling?	Very high	High	Moderate	Low	Very low
(Vol.2) What is the data volume the application in this domain receives as input?	Very high	High	Moderate	Low	Very low
(Vol.3) What is the amount of features within the data an average application in this domain receives as an input?	Very high	High	Moderate	Low	Very low
(Vol.4) What is the amount of time-based data that an average application within this domain currently uses?	Very low	Low	Moderate	High	Very high
Variety					
(Var.1) What is the current level of data variety in an the application in this domain?	Very high	High	Moderate	Low	Very low
(Var.2) What is the expected level of change in data variety after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low

Question	Score				
	1	2	3	4	5
Velocity					
(Vel.1) What is the number of data sources the application receives data from?	Very high	High	Moderate	Low	Very low
(Vel.2) What is the speed of the data that enters the application in this domain?	Very high	High	Moderate	Low	Very low
(Vel.3) What is the required speed that the data needs to travel through an application in this domain?	Very high	High	Moderate	Low	Very low
Veracity					
(Ver.1) What is the average data quality level within this domain?	Very low	Low	Moderate	High	Very high
(Ver.2) What is the level of variety in data quality the application within this domain receives as input?	Very high	High	Moderate	Low	Very low
(Ver.3) How major is it that the data fed into the application is completely correct (i.e. fault-free)?	Very major	Major	Moderate	Minor	Very minor
(Ver.4) What is the amount of data preparation needed in order to feed application data into a DRL solution for an average application in this domain?	Very high	High	Moderate	Low	Very low
Accountability					
(Acc.1) What is the expected seriousness level of an accountability issue after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low
(Acc.2) What is the expected amount of people involved in an accountability issue after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low
(Acc.3) What is the total expected amount of accountability issues after implementing DRL principles into the application in this domain?	Very high	High	Moderate	Low	Very low

Question	Score				
	1	**2**	**3**	**4**	**5**
Transparency					
(Trp.1) What amount of governance of the application in this domain is needed or will be used?	Very high	High	Moderate	Low	Very low
(Trp.2) How major is it to know the exact decision making process for the application in this domain?	Very major	Major	Moderate	Minor	Very minor
(Trp.3) What is the amount of sensitive data the application will be dealing with?	Very high	High	Moderate	Low	Very low
Transformability					
(Trf.1) What is the amount of similarity in structure between the implementation in this domain, and an implementation in the gaming domain?	Very low	Low	Moderate	High	Very high
(Trf.2) What is the expected difficulty level of transforming the application in this domain into one having roughly the same structure as a game?	Very high	High	Moderate	Low	Very low
(Trf.3) What is the level of connectedness of the application to other applications in this domain? (e.g what is the number of applications this application is embedded in, or connected to?)	Very high	High	Moderate	Low	Very low
(Trf.4) What is the application maturity level of an average application in this domain?	Very low	Low	Moderate	High	Very high
(Trf.5) What is the level of clarity with respect to determining what is a 'good' outcome of an average application within this domain?	Very low	Low	Moderate	High	Very high