Deep Learning for Abstract Argumentation Semantics

Dennis Craandijk

6002986

A thesis presented for the degree of
Master of Science

Supervisor: Floris Bex

Master Artificial Intelligence
Faculty of Science
Utrecht University
The Netherlands
2020

Contents

1 Introduction **1**

2 Preliminaries **3**
2.1 Computational argumentation 3
2.1.1 Abstract argumentation 4
2.1.2 Reasoning problems 6
2.2 Deep learning on graphs 9
2.2.1 Graph neural networks 9
2.2.2 Applications 12

3 Deep Learning for Abstract Argumentation Semantics **15**
3.1 Problem Setup 15
3.2 Argumentation Graph Neural Network 15

4 Experiments **17**
4.1 Data 17
4.2 Training 18
4.3 Metrics 18
4.4 Results 20
4.5 Scaling 20

5 Analysing AGNN Behaviour **22**

6 Enumerating Extensions **24**
6.1 Experimental Setup and Results 25

7 Discussion **27**
7.1 Related Work 27
7.2 Neural-symbolic computing 28
7.3 Conclusion 29
7.4 Acknowledgements 30

References **31**

A Guided tree search algorithm **36**

1 Introduction

Over the past few years an increasing amount of research effort has been directed towards designing deep learning methods that learn on problems from symbolic domains [21]. Deep learning methods have revolutionised areas such as natural language processing, computer vision and game playing due to the ability to discover features and patterns with little prior knowledge [36]. However, deep learning methods have until recently shown to be incapable of handling the discrete nature of symbolic reasoning. Since both symbolic-based and learning-based methods have complementary strengths and weaknesses, there has been a growing plea for combining these two pillars of artificial intelligence [41, 59].

Recent progress has sparked interest in graph neural networks (GNNs), a novel class of neural networks capable of performing computations over graphs. Graph-based problems are normally solved symbolically by defining the relations in a graph using a symbolic formalism, e.g. logic or mathematics. Such formalisms provide methods, such as deduction or arithmetic, which enable reasoning about the relations in a graph [34]. The design of symbolic algorithms however, requires significant specialised knowledge and trial-and-error. The appeal of using deep learning is that graph computations can be learned from data, rather than specifying the computations in advance. Whereas conventional machine learning methods struggle to capture the relational structures of graphs, GNNs have shown to perform well on a range of graph-based problems [61, 62] due to a strong relational inductive bias [5]. GNNs can for instance be trained to solve constraint satisfaction problems which could previously only be solved by symbolic solvers, such as boolean satisfiability [53] and Sudoku puzzles [45].

One symbolic domain of artificial intelligence that is relatively unexplored with respect to GNNs is *computational argumentation*. Argumentation is an important reasoning capacity which is present in many aspects of human reasoning and interaction. The aim of computational argumentation is to understand and represent the mechanism of argumentation, with a focus on the computational methods to determine the validity of a claim based on the interactions between arguments and counterarguments. The defeasible nature of computational argumentation allows the validity of a claim to change based on new information. This property enables reasoning in environments where conflicting or incomplete information exists. With applications in multi-agent systems, decision-making tools, medical and legal reasoning, argumentation has become a major subfield of artificial intelligence [2].

Much of the theory in computational argumentation is built on Dung's [16] pioneering work on abstract argumentation frameworks. Dung's formalism allows to arbitrate between conflicting arguments based on the attack relations between arguments. Through a process of conflict resolution it is possible to determine which arguments 'win' and can thus be accepted. Dung introduced several acceptability semantics that define which sets of arguments (*extensions*) can be reasonably accepted given an argumentation framework (AF) of arguments and attacks between these arguments, often represented as a directed graph. Thus, it can be determined if an argument can be accepted given an AF by looking at whether it is contained in some extensions (credulous acceptance) or all extensions (sceptical acceptance) under a given semantics.

The process of determining which arguments can be accepted can suffer from high computational complexity. Notably, finding the accepted arguments is shown to be intractable in various settings [18]. Due to the computational complexity of determining which arguments can be accepted, the design of efficient methods for computing extensions and acceptability constitutes an active research direction within the argumentation community. Most current approaches solve acceptance problems by translating the problem to a symbolic formalism for which a dedicated solver exists, such as constraint-satisfaction problems, propositional logic or answer-set programming [20, 12].

Since graphs are the basic representation of one of the most well-established formal models of argumentation, it is natural to ask whether GNNs can be applied to solve some of the computational problems of argumentation. In this thesis, an argumentation graph neural network (AGNN) is proposed that learns to predict credulous and sceptical acceptance of arguments under 4 well-known argumentation semantics. Furthermore, a method is introduced to predict (multiple) extensions given an AF by using AGNN to guide a search procedure. The learning-based approach to determining argument acceptance described in this thesis shows that sub-symbolic deep learning techniques can accurately solve a problem that could previously only be solved by sophisticated symbolic solvers. Moreover, by inspecting AGNN's behaviour, it is observed that AGNN learns to adhere to basic principles of argument semantics as identified in the literature [4].

The rest of this thesis is structured as follows. Chapter 2 discusses Dung's [16] abstract argumentation frameworks and graph neural networks [5]. Chapter 3 discusses the problem setup and introduces the AGNN model. In Chapter 4 the experimental setup (metrics, data and training) and results, also with respect to scalability, are discussed. Chapter 5 discusses the behaviour of the AGNN and how it relates to symbolic algorithms for abstract argumentation. Chapter 6 discusses the experiments focusing on finding multiple extensions. Finally, Chapter 7 discusses related research and concludes the thesis.

The results of this research have been published as in the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020) [14]. Compared to the IJCAI paper, this thesis includes extended preliminaries with more examples (Chapter 2); a detailed description of the experiments (Chapter 4) and an elaborate discussion (Chapter 7). The code to reproduce this research is published at https://github.com/DennisCraandijk/DL-Abstract-Argumentation.

2 Preliminaries

In this chapter some concepts which are central to this thesis are discussed. First, an overview is given of some concepts from computational argumentation, with a focus on Dung's abstract argumentation frameworks and semantics. Additionally, some central reasoning problems from computational argumentation and how they are currently solved are reviewed. Finally, this chapter discusses the basic concept of a graph neural network and mention some current applications.

2.1 Computational argumentation

Approaches to computational argumentation focus on the representational and computational aspects of argumentation and can be divided into *abstract* and *structured* argumentation. The abstract argumentation formalism was introduced by Dung's [16] pioneering work in which he defined a set of formal properties to represent arguments as *argumentation frameworks* (AFs). An AF consists of arguments and attacks between these arguments and is generally represented as a directed graph where arguments are nodes and the attack relations directed edges (see Figure 1).

In abstract argumentation there is no internal structure to an argument. Structured argumentation provides a more detailed formalism by assuming a formal logical language and specifying rules for the construction of arguments and counterarguments. In structured argumentation the relationship between a claim and the reasons for this claim are made explicit. This relationship can, for instance, be defined using logical entailment. Multiple approaches to structured argumentation have been proposed, such as Assumption-based argumentation (ABA) [17] and ASPIC$^+$ [49].

Although both structured and abstract argumentation are relevant argumentation formalisms, this research only covers abstract argumentation for two reasons. First, structured argumentation is structurally more complex compared to abstract argumentation. Whereas arguments are only connected through attack relations in abstract argumentation, in structured argumentation multiple types of relations exists. Due to the simplicity of its graph structure, abstract argumentation can therefore more easily be incorporated into existing GNN approaches without the need for much extra engineering effort. Secondly, since Dung's formalism is the most well-established approach, there exists a wide range of literature on reasoning problems and solvers relevant to this research. Moreover, since structured argumentation approaches (such as ABA and ASPIC$^+$) are an instantiation of Dung's abstract argumentation framework, the acceptability of ABA or ASPIC$^+$ arguments can be determined with abstract argumentation solvers. Therefore, abstract argumentation is a natural candidate for this research.

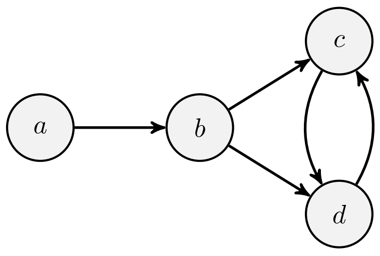


Figure 1: Graph representations of the AF F_e.

2.1.1 Abstract argumentation

Central to abstract argumentation framework as defined by Dung [16] are notions of argument, attack and defence.

Definition 1. *An* argumentation framework *(AF) is a pair (A, R) where A is a (finite) set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that a attacks b. An argument $a \in A$ is defended by c if, for b such that $(b, a) \in R$, there exists a c such that $(c, b) \in R$. A set $S \subseteq A$ is said to attack b if there is an $a \in S$, such that $(a, b) \in R$. An argument $a \in A$ is defended by $S \subseteq A$ iff, for each $b \in A$ such that $(b, a) \in R$, S attacks b.*

An argumentation framework is intuitively represented as a directed graph where arguments are nodes and attacks are directed edges.

Example 1. *Figure 1 illustrates the AF $F_e = (\{a, b, c, d\}, \{(a, b), (b, c), (b, d), (c, d), (d, c)\})$, which serves as a running example.*

Given an AF, it is possible to arbitrate between conflicting arguments. Arbitration in abstract argumentation is based on the notion that 'winning' arguments are those that collectively and adequately respond to all counterarguments [4]. This concept is expressed through different *semantics*. Semantics define the properties which are expected to be satisfied by a set of arguments in order to accept them as a single point of view. The interplay between attack and defence relations determines which subset of arguments can be accepted with respect to different semantics. For example, a basic property of all semantics is that sets of arguments should be *conflict free* in order to be accepted. This implies that no argument in the set is attacked by another argument in the set. Two well-established approaches to evaluate argumentation semantics are the extension-based approach and the labelling-based approach.

2.1.1.1 Extension-based semantics

In his original work, Dung [16] proposed to evaluate different argumentation semantics based on *extensions*. Extensions are sets of arguments that can jointly be accepted and represent some coherent view on the underlying AF. A semantics for argumentation frameworks is defined as a function σ which assigns a set of extensions to each AF. This research considers the admissible sets and the complete, preferred grounded and stable extensions which are assigned by the following functions respectively adm, com, prf, grd, stb.

Definition 2. *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is* conflict-free *(in AF), if there are no $a, b \in S$, such that $(a, b) \in R$. The collection of sets which are conflict-free is denoted by $\mathsf{cf}(F)$. For a conflict-free set $S \in \mathsf{cf}(F)$, it holds that:*

- *$S \in \mathsf{adm}(F)$, if each $a \in S$ is defended by S;*
- *$S \in \mathsf{com}(F)$, if $S \in \mathsf{adm}(F)$ and for each $a \in A$ defended by S it holds that $a \in S$;*
- *$S \in \mathsf{grd}(F)$, if $S \in \mathsf{com}(F)$ and for each $T \in \mathsf{com}(F)$, $T \not\subset S$;*
- *$S \in \mathsf{prf}(F)$, if $S \in \mathsf{adm}(F)$ and for each $T \in \mathsf{adm}(F)$, $S \not\subset T$;*
- *$S \in \mathsf{stb}(F)$, if for each $a \in A \setminus S$, S attacks a.*

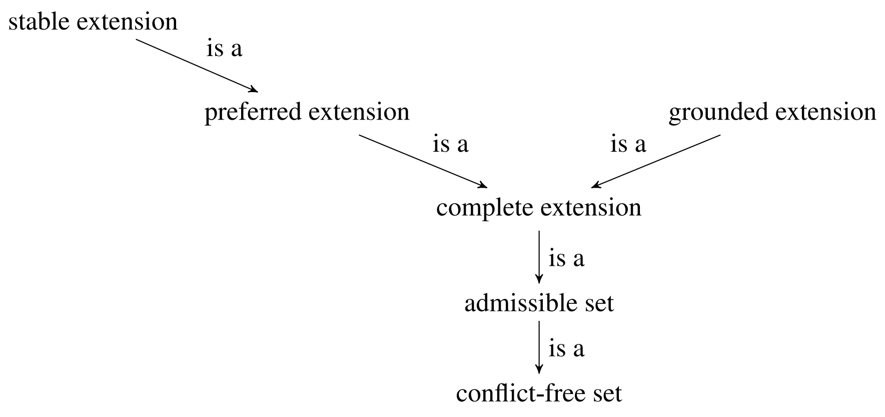


Figure 2: An overview of different extension-based semantics and their relations [4]

In other words, a set of arguments is said to be conflict-free if no argument in the set is attacked by another argument in the set. A set of arguments is admissible if it is conflict-free and can defend itself from all attacks. A set arguments is a complete extension if it is admissible and includes each argument that it defends. The grounded extension is the subset-minimal complete extension. A preferred extensions is a subset-maximal admissible set. A set of arguments is stable if it attacks each argument which does not belong to it. The different extension-based semantics are related as shown in Figure 2.

Example 2. *The extensions of F_e under the complete, grounded, preferred and stable semantics are:* $\mathsf{com}(F) = \{\{a\}, \{a, c\}, \{a, d\}\}$; $\mathsf{grd}(F) = \{a\}$; $\mathsf{prf}(F) = \{\{a, c\}, \{a, d\}\}$; $\mathsf{stb}(F) = \{\{a, c\}, \{a, d\}\}$.

2.1.1.2 Labelling-based semantics

Caminada [9] introduced an alternative way to evaluate the acceptance of arguments based on labels. In order to determine argument acceptance with respect to a semantics, each argument in an AF is assigned a label from the set $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UNDEC}\}$.

Definition 3. *Let $F = (AR)$ be an AF. A labelling is a total function $Lab : A \to \{\mathsf{IN}, \mathsf{OUT}, \mathsf{UNDEC}\}$. The set of arguments which are* IN*,* OUT *or* UNDEC *with respect to a semantics σ is denoted by* $\mathsf{in}_\sigma(F), \mathsf{out}_\sigma(F), \mathsf{undec}_\sigma(F)$

An argument is labelled IN if the argument is accepted and it is labelled OUT if it is rejected. Whereas extension-based semantics only define if an argument is accepted or not, labelling-based semantics permit the assignment of an intermediate status to arguments by assigning the UNDEC label. An argument is undecided if it cannot be accepted while there are also not enough reasons to reject it.

Example 3. *The labelling of F_e under the grounded semantics is:* $\mathsf{in}_{\mathsf{grd}}(F) = \{a\}$, $\mathsf{out}_{\mathsf{grd}}(F) = \{b\}$, $\mathsf{undec}_{\mathsf{grd}}(F) = \{c, d\}$.

For all semantics used in this work there exists a one-to-one mapping from labellings to extensions. An advantage of a labelling-based approach is that the label of an argument has consequences for the label of its neighbours. An argument can for example only be labelled IN if every attacker is labelled OUT. Similarly, when an argument is labelled IN, all arguments it attacks should be labelled OUT. These pairwise relations and interactions make labelling-based approaches a convenient method to compute the extensions of an AF. Modgil and Caminada [43] define what it is for an argument to be assigned a legal labelling:

Definition 4. *Let $F = (AR)$ be an AF and L be a labelling for F. An argument $a \in A$ is said to be:*

- *legally labelled IN iff a is labelled IN and every b, such that $(b, a) \in R$, is labelled OUT;*
- *legally labelled OUT iff a is labelled OUT and there exists at least one b, such that $(b, a) \in R$, that is labelled IN;*
- *legally labelled UNDEC iff a is labelled UNDEC and not every b, such that $(b, a) \in R$, is labelled OUT and there exists no b that is labelled IN.*

2.1.2 Reasoning problems

Given an argumentation framework F and a semantics σ, there are a number of different approaches to evaluate the acceptance of arguments. Evaluating the acceptance of arguments requires computing the answer to some reasoning problem (i.e. computational problem) based on the given AF. $\mathsf{Enum}_\sigma(F)$ enumerates all extensions of semantics σ and $\mathsf{Count}_\sigma(F)$ counts the number of extensions. Additionally since semantics define which arguments can be accepted in a single point of view, one can be interested whether there exists a position where in which a specific argument is accepted [43]. An argument a is said to be *credulously* accepted with respect to a semantics σ if it is contained in some σ-extension. An argument is *sceptically* accepted if it is contained in all σ-extensions. $\mathsf{Cred}_\sigma(a, F)$ and $\mathsf{Scept}_\sigma(a, F)$ denote whether an argument a is respectively credulously or sceptically accepted with respect to σ. Finally, $\mathsf{Ver}_\sigma(S, F)$ verifies whether a given set S is a σ-extension of F.

Definition 5. *Given an AF $F = (A, R)$, a semantics σ and an argument $a \in A$ then:*

- $\mathsf{Enum}_\sigma(F) = \sigma(F)$
- $\mathsf{Count}_\sigma(F) = |\sigma(F)|$
- $\mathsf{Cred}_\sigma(a, F) = \begin{cases} \textit{Yes} & \textit{if } a \in \bigcup_{\mathcal{E} \in \sigma(F)} \mathcal{E} \\ \textit{No} & \textit{otherwise} \end{cases}$
- $\mathsf{Scept}_\sigma(a, F) = \begin{cases} \textit{Yes} & \textit{if } a \in \bigcap_{\mathcal{E} \in \sigma(F)} \mathcal{E} \\ \textit{No} & \textit{otherwise} \end{cases}$
- $\mathsf{Ver}_\sigma(S, F) = \begin{cases} \textit{Yes} & \textit{if } S \in \sigma(F) \\ \textit{No} & \textit{otherwise} \end{cases}$.

Example 4. *The solutions to common reasoning problems in AF F_e with respect to the preferred semantics are:* $\mathsf{Enum}_{prf}(F) = \{\{a, c\}, \{a, d\}\}$; $\mathsf{Count}_{prf}(F) = 2$; $\mathsf{Cred}_{prf}(c, F) = \textit{Yes}$; $\mathsf{Scept}_{prf}(c, F) = \textit{No}$; $\mathsf{Ver}_{prf}(\{a, b\}, F) = \textit{No}$.

σ	Cred_σ	Scept_σ	Ver_σ
adm	NP-c	trivial	in P
com	NP-c	in P	in P
grd	in P	in P	in P
prf	NP-c	Π_2^P-c	coNP-c
stb	NP-c	coNP-c	in P

Table 1: Computational complexity of decision problems in abstract argumentation [18]

2.1.2.1 Complexity

For a number of semantics, computing the acceptance status of an argument is no trivial task. We recall some basic concepts of computational complexity. Decision problems (problems where the output is either Yes or No) can belong to one of several computational complexity classes. All problems solvable by a deterministic Turing machine in polynomial time belong to the class P. The problems where the Yes (respectively No) instances can be decided in polynomial time by a non-deterministic Turing machine belong to complexity class NP (resp. coNP). The problems NP (resp. coNP) in NP to which all NP problems can be reduced to in polynomial time are NP-complete (resp coNP-complete). Finally the class Π_2^P-c contains all problems verifiable in polynomial time using an coNP-oracle. For a more extensive description of the different complexity classes see Papadimitriou [46].

The reasoning problems described in definition 5 include three decision problems: $\mathsf{Cred}_\sigma(a, F)$, $\mathsf{Scept}_\sigma(a, F)$ and $\mathsf{Ver}_\sigma(S, F)$. Different complexity studies have been done on these decision problems [18]. Table 1 shows the complexity of the problems under different semantics. Most notably, deciding credulous acceptance under the admissible, complete, preferred and stable semantics is shown to be NP-complete (NP-c).

2.1.2.2 Direct computation

Algorithmic implementations for computational argumentation can be divided into *reduction-based* and *direct* approaches. Direct implementations use a dedicated algorithm for a specific reasoning problem. Most dedicated algorithms employ labellings to determine the extensions of an AF [12]. Generally, labelling-based algorithms start with assigning labels to uncontroversial arguments, and subsequently search for correct assignments of the remaining arguments. Based on different semantics, different labelling rules are used. When all arguments are labelled, without any labelling conflict, a correct labelling-based semantics is found.

For example, Modgil and Caminada [43] describe how the grounded labelling can be found by first labelling all arguments which are unattacked, or only attacked by OUT arguments, as IN. Subsequently all arguments attacked by those labelled IN are labelled OUT. These two steps are iterated until no new arguments are labelled IN or OUT, after which all remaining unlabelled arguments a labelled UNDEC. In the resulting grounded labelling all arguments labelled IN are the grounded extension.

Enumerating all preferred extensions with labelling-based algorithm is slightly more complex. Modgil and Caminada [43] also provide a labelling-based algorithm for the preferred extensions. The core idea behind this algorithm is to first label all arguments as IN and subsequently re-label arguments as either OUT or UNDEC until the set of IN arguments becomes admissible. The goal is to minimise the number of re-labelled

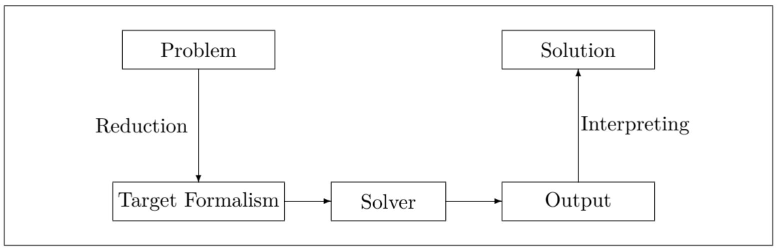


Figure 3: Reduction-based computation (source: Cerutti et al. [10])

arguments. By minimising the arguments which are OUT or UNDEC, the set of IN arguments will be a subset-maximal admissible extension (i.e. a preferred extension).

2.1.2.3 Reduction-based computation

In a reduction-based approach, the problem is first reduced to a target formalism for which a dedicated solvers exists. The solver is used to find a solution in target formalism, and afterwards this solution is decoded back to the original problem as illustrated in Figure 3. The overview by Charwat et al. [12] shows there exists a range of reduction-based solvers using different target formalisms, such as constraint-satisfaction problems, propositional logic or answer-set programming. Among these solvers, some prominent approaches encode the problem to a Boolean expression (i.e. a propositional logic formula) and use a Boolean satisfiability (SAT) solver to find a solution. A Boolean expression is built from Boolean variables, parentheses and the following operators: $\wedge$ (conjunction), $\vee$ (disjunction) and $\neg$ (negation). A SAT solver tries to find satisfiable assignment of a Boolean expression. The formula is said to be satisfiable if the variables in the expression can consistently be replaced by Boolean values in such a way that the expression evaluates to TRUE. By encoding an reasoning problem as a Boolean expression, advanced SAT solvers can be used to solve argumentation reasoning problems.

Consider an argumentation framework $F = (A, R)$, and a set of arguments $S \subseteq A$. In order to know if S is conflict-free, one should check that for every argument $a \in S$, the attacking arguments b are is not in S. Thus, if the propositional logic formula

$$\bigwedge_{a \in S} (a \wedge \bigwedge_{\langle b,a \rangle \in R} \neg b) \tag{1}$$

is satisfiable, S is a conflict-free set. The logical formula essentially encodes the properties of a conflict-free set as satisfiability constraints. The formula for a conflict-free set can be extended to encode the properties of admissible sets. This is done by adding a conjunction which states that in order to accept a, for each attacker b, some defender c must be accepted as well. Thus, if the propositional logic formula

$$\bigwedge_{a \in S} ((a \wedge \bigwedge_{\langle b,a \rangle \in R} \neg b) \wedge (a \bigwedge_{\langle b,a \rangle \in R} (\bigvee_{\langle c,b \rangle \in R} c))) \tag{2}$$

is satisfiable, S is an admissible extension. For a more extensive description of different reduction-based methods see Charwat et al. [12].

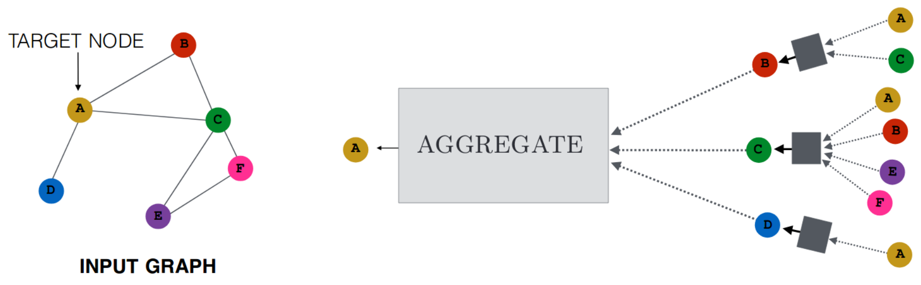


Figure 4: Neighbourhood aggregation through message passing. To generate the embedding for node A, the messages from A's graph neighbours are aggregated. In turn, the messages coming from these neighbours are based on the aggregated messages from their respective neighbours, and so on. (source: Hamilton, Ying, and Leskovec [25])

2.2 Deep learning on graphs

Graphs – being the basic representation of argumentation frameworks – are known to have complicated structures which contain rich underlying value [3]. The design of algorithms to solve problems on graphs generally requires specialised knowledge and trial-and-error. In recent years a considerable amount of research has been directed towards designing deep learning algorithms for graph-based problems. The appeal of using deep learning is that graph computations can be learned rather than hand-crafted. Deep learning has revolutionised the field of machine learning. Deep learning methods (such as convolutional neural networks (CNNs) [37], recurrent neural networks [50] (RRN) and auto-encoders [28]) have shown to be able to extract complex latent representations from data. This has resulted in superior performance in many machine learning tasks, which once relied on handcrafted features to extract useful information [36]. Until recently, existing machine learning methods have shown to be incapable of handling the complexity of graph data. The non-Euclidean and complex structure of graphs have caused some important deep learning operations to fail on graphs. Inspired by the success of deep learning-based methods on images and text, a substantial amount of research effort has been directed to apply ideas from CNNs and RNNs to graphs. This has led to a wide range of literature on graph-based learning systems called *graph neural networks* (GNNs) [61, 62]. Many different types of architectures have been proposed ranging from graph convolutional networks and recurrent graph neural networks to graph auto-encoders. In general, these architectures take a graph structure with node and edge attributes as input and output values for a specific task. The output values can typically occur on either the node, edge, or global level. A GNN can for instance be used classify nodes, predict missing edges or obtain a compact representation of a complete graph. While deep learning methods for graphs are relatively young, the overview by Zhou et al. [62] shows promising results on a range of graph-based problems.

2.2.1 Graph neural networks

The basic concept of a GNN is that nodes aggregate information from their neighbours using neural networks. Nodes are assigned a multidimensional embedding which represent some information about the node (e.g. a state, numerical value or some other attribute). The representations are updated in an iterative fashion by propagating messages between the connected nodes in a process called *message passing* [24]. At each message passing step, nodes send their embeddings to all connected nodes. Receiving nodes aggregate the messages of their neighbours and compress these messages into a new embedding using a neural network. The resulting embedding then becomes the new representation of each node. With each iteration nodes thereby aggregate information from further in the graph. Figure 4 shows a visual representation of this process.

Numerous variants on this message passing principle have been proposed. Whereas most approaches pass messages between nodes, some operate on an edge or global level. Battaglia et al. [5] unify a great number of variants into a single framework called a *graph network*. The authors leave out the predicate 'neural' to emphasise that a graph network can be implemented with other functions than neural networks. However, since the aim of this thesis is to use neural networks, the term graph neural network is used here. Battaglia et al. [5] describe the basic building blocks which are used in various graph neural networks. The basic idea is that elements in a graph are assigned multidimensional embeddings which are updated in computational steps.

2.2.1.1 Initialisation

Within a GNN, a graph is described as a tuple $G^t = (u^t, V^t, E^t)$, at computational step t. u^t is the global embedding, representing some global information. $V^t = \{v_i^t\}$ is a set of nodes, where v_i^t is node i embedding at step t. $E^t = \{e_{ij}^t\}$ is a set of edges, where each e_{ij} denotes an edge embedding between nodes i and j at step t. A GNN is initialised such that relevant embeddings represent relevant information. Embeddings are real-valued multidimensional vectors that can represent some node, edge or global information contained in the graph. A node embedding can for instance represent a node label, an edge embedding the distance between two nodes and a global embedding a property of the graph. At $t = 0$ the GNN is initialised by mapping the inputs to their respective embeddings with input functions on the edge $x_{e_{ij}}$, node x_{v_i} or graph x_u level such that

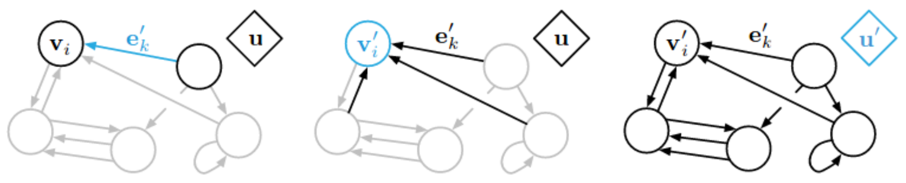
$$\begin{aligned} e_{ij}^0 &= x_{e_{ij}} \\ v_i^0 &= x_{v_i} \\ u^0 &= x_u \end{aligned} \tag{3}$$

2.2.1.2 Computational steps

After the initialisation the embeddings are updated by propagating information between connected elements. A computational step typically involves aggregating information from connected elements and subsequently updating the embeddings based on the aggregated information. These computational steps can be applied in an iterative fashion, allowing information to propagate through the graph.

At each computational step t information propagates through the graph and the attributes at each level are updated through the update functions (θ^e, θ^v, θ^u) and aggregation functions ($p^{e\to v}$, $p^{e\to u}$, $p^{v\to u}$) according to

$$\begin{aligned} e_{ij}^{t+1} &= \theta^e(e_{ij}^t, v_i^t, v_j^t, u^t) & m_i^t &= p^{e\to v}(E_{N(i)}^t) \\ v_i^{t+1} &= \theta^v(m_i^{t+1}, v_i^t, u^t) & \overline{e}^t &= p^{e\to u}(E^t) \\ u^{t+1} &= \theta^u(\overline{e}^{t+1}, \overline{v}^{t+1}, u^t) & \overline{v}^t &= p^{v\to u}(V^t) \end{aligned} \tag{4}$$



(a) Edge update (b) Node update (c) Global update

Figure 5: Update steps in a graph neural network. Blue indicates that an element is being updated and black indicates elements which are involved in the update (source: Battaglia et al. [5])

where E_i^t denotes all edges which have a connection into node i. The update and aggregation functions are the central elements of a graph neural network. The aggregation functions define how to aggregate incoming information from multiple sources. Typically aggregation operations which are employed are *summation*, *mean* or *maximum*. The update functions are learned differential functions that define how incoming information should be used to update the attributes of interest. All functions are reused on their respective embeddings, allowing GNNs to operate on graph of different sizes and shapes. With these functions a GNN can be structured into three distinct updates steps.

- Edge update
  - θ^e maps the edge embedding, the pair of nodes connected by the edge and the global to new edge embedding.
- Node update
  - $p^{e \to v}$ aggregates the incoming edge embeddings (or 'messages') for all receiving nodes.
  - θ^v takes the aggregated edge embeddings, along with the receiving node embedding and a global embedding and computes an updated node embedding.
- Global update
  - $p^{e \to u}$ and $p^{n \to u}$ aggregate all edge and node embeddings.
  - θ^v computes a global embedding based on the aggregated edge and node embeddings.

The edge, node and global update steps are the building blocks which can be used to build a GNN. Figure 5 shows which elements are involved at each update step. For most tasks only some of the building blocks are necessary to build a GNN. The update and aggregation functions used in the building blocks form a neural message passing algorithm which determines how information propagates through the graph. Since the update functions are learned differential functions (such as neural networks), the parameters of the neural message passing algorithm can be optimised to perform the desired task, thereby enabling learning capability on graph-based problems.

2.2.1.3 Readout

After each computational step the embeddings can be read out with a readout function. A readout function maps an embedding to a desired output, such as a label or a

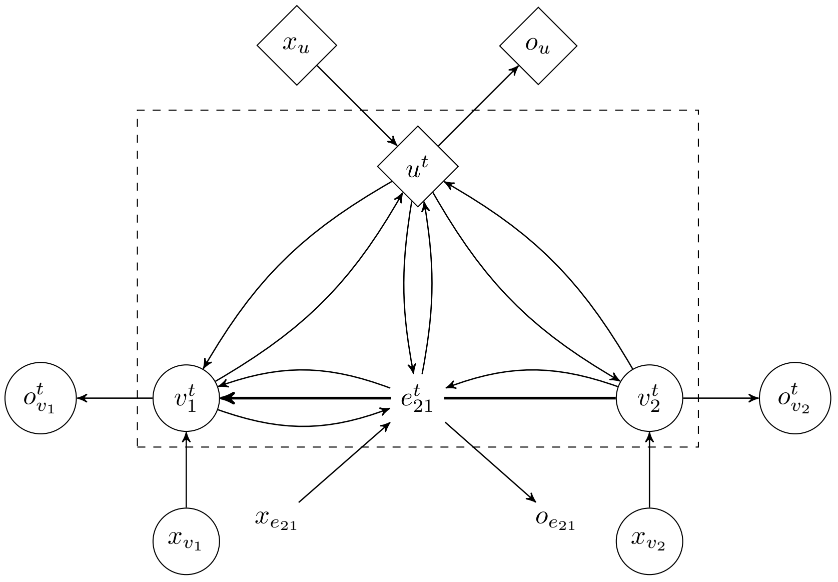


Figure 6: A graph neural network with edge, node and graph embeddings, on a two node graph with a directed edge from node 2 to node 1. The thick line indicates the edge between the nodes. The thin lines show how information flows between the variables as described by the update, aggregation, input and output functions (see Equations 4, 3 and 5). All variables within the dotted box are effected by the update and aggregation functions. The variables outside the box are determined by the input and readout functions and are used as a mapping between the GNN and the task at hand.

numerical value. The readout functions can be *edge-focused* (r^e), *node-focused* (r^v), *graph-focused* (r^u) or a custom mixture depending on the task at hand.

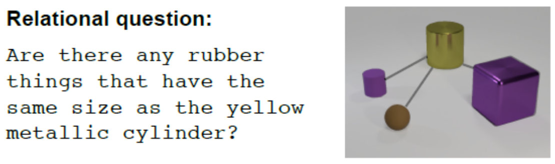
$$\begin{aligned} o^t_{e_{ij}} &= r^e(e^t_{ij}) \\ o^t_{v_i} &= r^v(v^t_i) \\ o^t_u &= r^u(u^t) \end{aligned} \tag{5}$$

The readout functions are also learned differential functions in order to learn a mapping between the embeddings and their respective outputs.

Figure 6 shows how all the variables are interconnected through the functions described in update and aggregation functions (Equation 4) and the input (Equation 3) and readout functions (Equation 5). In summary, the input and readout functions provide a mapping between the task and the GNN embedding, while the update and aggregation functions define how information should propagate between those embeddings in order to solve a task.

2.2.2 Applications

The recent developments in GNNs has enabled the use deep learning on graphs. This has opened up the possibility to use learning-based methods on problems which could previously only be solved with symbolic methods. This approach to graph-based problems has led to some learning-based breakthroughs in domains relevant to this research, namely combinatorial optimisation and relational reasoning [22]. The performance of


Relational question:

```
Are there any rubber
things that have the
same size as the yellow
metallic cylinder?
```

Figure 7: An illustrative example from the CLEVR dataset of relational reasoning (source: Santoro et al. [51])

most learning-based approaches is generally below that of symbolic handcrafted algorithms (in terms of run time as well as accuracy). However, the goal of most authors is not primarily to outperform symbolic algorithms but rather to show the potential for learning-based methods in solving symbolic reasoning problems.

2.2.2.1 Combinatorial optimisation

A prominent graph-based combinatorial optimisation problem researchers have been trying to solve with a GNN is the travelling salesman problem (TSP). The problem is to find a cycle of minimum length where every node is visited only once. Most approaches try to build a solution on the graph by sequentially adding unvisited nodes to the tour. However, given the hard nature of determining the optimal tour, determining the best node can involve algorithmic decisions that either require too much computing time or are not mathematically well defined [6]. Li, Chen, and Koltun [38] therefore proposes to use a GNN to estimate the likelihood whether a node is part of the optimal solution. The GNN takes in the graph along with the current tour and tries to extract features in the graph structure which are predictive for whether a node should be added tot the tour. Experimental results show this method to perform on par with highly optimised state-of-the-art heuristic solvers. Khalil et al. [31] use a similar architecture as a deep reinforcement learning algorithm to learn a policy for selecting the next node in the graph. Although the performance of this method is considerably lower compared to Li, Chen, and Koltun [38], it shows the possibility of learning to solve graph problems end-to-end. For an overview of the use of deep learning methods on other combinatorial optimisation problems see Bengio, Lodi, and Prouvost [6]

2.2.2.2 Relational reasoning and constraint satisfaction

Graph neural networks have been used on a number of relational reasoning tasks, where relational reasoning entails reasoning about interaction between entities [34, 26]. For instance Santoro et al. [51] models images from the CLEVR visual question answering dataset [30] (which contains relational questions as shown in Figure 7) as a graph by representing the objects as nodes and the relations as edges. The authors use a GNN to learn about the pair-wise relations between objects and show this method performs well on the CLEVR dataset.

Palm, Paquet, and Winther [45] enhance the GNN by Santoro et al. [51] to enable multi-step relational reasoning. This allows information to propagate through the graph in multiple computational steps, allowing complex chains of interactions. Palm, Paquet, and Winther [45] show this enables a GNN to solve Sudoku puzzles, a typical multi-step relational reasoning task. Solving a Sudoku involves recursively filling cells with a digit and checking how this influences other cells. The authors model this principle by representing cells of a Sudoku grid as nodes with edges to and from each other cell in the same row, column and box. Through the principle of message passing, a cell can send its value and inform the connected cells not to take the same value. The proposed GNN is effectively trained to develop a message passing strategy which models this behaviour. This model was able to solve 96.6% of the hardest 9-by-9 Sudoku puzzles, which can sometimes require op to 64 relational reasoning steps. For an overview of other uses of GNN's in relational reasoning tasks see Lamb et al. [35].

3 Deep Learning for Abstract Argumentation Semantics

Given the recent progress in using GNN on graph-based problems, this thesis attempts to show that it is possible to learn to reason about an argumentation framework with a graph neural network. Reasoning about AFs requires reasoning about the relations between arguments and satisfying the constraints of argumentation semantics. Section 2.2.1 discussed how GNN's exhibit both relational reasoning and constraint satisfaction capabilities on different types of problems. The main goal of this research is to explore the capabilities and limits of a GNN when used on reasoning problems from computational argumentation. Specifically, this thesis examines whether a GNN is able to learn general procedures which enable it to reason about the interactions between arguments in a fashion similar to symbolic algorithms. In order to test the main hypothesis, a number of prominent reasoning problems is selected and a GNN architecture to solve these problems is proposed.

3.1 Problem Setup

Deciding the sceptical and credulous acceptance of arguments are among the most prominent reasoning problems in abstract argumentation [10]. These problems have been studied widely and as a result many solvers exist [12]. As shown in Section 2.1 most of these problems suffer from high computational complexity, making them challenging to solve within a limited amount of computational steps. While such reasoning problems generally demand discrete and exact solutions, neural networks can only output continuous approximations. In order to solve argument acceptability problems with a GNN the reasoning problems are posed as classification problems. Consider an AF $F = (A, R)$ and a semantics σ. The goal is to approximate a function f_σ mapping the input F to a binary labelling $f_\sigma(F)$ denoting the acceptability of all arguments in A under semantics σ. The function is approximated by producing a value for each argument in the interval $[0, 1]$ - representing the likelihood whether an argument can be accepted - which is rounded to produce a binary answer (accept or reject).

3.2 Argumentation Graph Neural Network

In this section the *argumentation graph neural network* (AGNN) model is introduced. An AGNN maps an AF to a graph representation with arguments as nodes and attacks as directed edges and AGNN assigns a multidimensional embedding to all nodes. These embeddings are then iteratively updated by performing a number of message passing steps. At each iteration nodes broadcast their embeddings by exchanging messages with their neighbours and subsequently update their embedding based on the incoming messages. After each iteration those embeddings can be read out to produce the predicted likelihood of the respective argument being accepted.

More formally, G is an AF graph representation in which arguments are nodes and attacks are directed edges. Each node i is assigned an embedding, denoted by v_i^t at step t. The node embedding represents the initial state of an argument in the argumentation framework. Each node is initialised by a learned embedding x_i such that

$$v_i^0 = x_i \tag{6}$$

After initialising all node attributes, the attributes are updated recurrently in $\mathcal{T}$ message passing steps. Each message passing step the embeddings are updated according to two steps. First the messages are computed according to:

$$\begin{aligned} m_i^{t+1} = & \sum_{j \in N^s(i)} M^s(v_i^t, v_j^t) + \\ & \sum_{k \in N^t(i)} M^t(v_i^t, v_k^t) \end{aligned} \tag{7}$$

where $N^s(i)$ and $N^t(i)$ denote all nodes which have a connection with node i and for which i is the source or target node respectively. The message functions M^s and M^t are multilayer perceptrons (MLPs) which learn to compute a message to send along edges based on the embeddings of the nodes it connects. M^s computes a message from the source node to the target node and M^t vice versa. Messages from all neighbours are subsequently aggregated through summation to form the incoming message m_i^t. Two message functions are used to handle the directed nature of edges in an AF graph representation. This enables nodes to discriminate between messages from neighbours with which it has an incoming or outgoing connection. After computing the messages, the node embeddings are updated according to:

$$(v_i^{t+1}, h_i^{t+1}) = U(h_i^t, m_i^{t+1}, x_i) \tag{8}$$

where the update function U is a Recurrent Neural Network (RNN) which learns how to update a node given the incoming message and the node's input feature, where h_i^t is the RNNs hidden state. By updating the node embeddings recurrently while also accounting for the input features, AGNN is able to iteratively refine embeddings without forgetting any potentially relevant information. The message and update functions form the core of the AGNN model. Together, the functions yield a neural message passing algorithm whose parameters can be optimised.

After each iteration the embeddings can be read out with the readout function R. R is an MLP that learns to map a node's embedding v_i^t to a logit probability

$$o_{v_i}^t = R(v_i^t) \tag{9}$$

representing the likelihood of the respective argument being accepted. The logit probability can subsequently be converted to a likelihood in the interval $[0, 1]$ using a sigmoid function. Together, the readout and sigmoid function provide a mapping between the multidimensional node embeddings and the desired output.

With respect to the general GNN architecture described in Section 2.2.1, AGNN is a node-focused GNN consisting of a node update building block specifically designed to handle the directed nature of attacks in an AF. In terms of argumentation AGNN learns how to initialise arguments with an embedding; recurrently update these embeddings by exchanging messages between arguments over the attack relations; and map the argument embeddings to a likelihood of that argument being accepted. By using neural networks for the message and update functions, AGNN learns how arguments should exchange information in order to iteratively work towards a solution.

4 Experiments

In this chapter the reasoning capabilities of an AGNN are evaluated on problems in argumentation through empirical experimentation. A dataset of argumentation frameworks is generated and AGNN is trained to solve various argument acceptance problems. Afterwards the trained model is evaluated and assessed on whether the learned message passing algorithm has learned a procedure that scales to larger argumentation frameworks. The code to reproduce the experiments is published at https://github.com/DennisCraandijk/DL-Abstract-Argumentation.

4.1 Data

A variety of challenging argumentation frameworks is generated by sampling from the following AF generators from the International Competition on Computational Models of Argumentation [20]:

- *AFBenchGen2*. A random AF generator using Barabasi-Albert [1], Erdös-Rényi [19] or Watts-Strogatz [60] graph generation methods [11].
- *AFGen Benchmark Generator*. A random AF generator with a 'variety of algorithmic and probability behaviours' [56].
- *GroundedGenerator*. A random AF generator that generates AFs that (likely) possess a large grounded extension [58].
- *SccGenerator*. A random AF generator that generates AFs that (likely) posses many strongly connected arguments (i.e. arguments that are connected to every other argument) [58].
- *StableGenerator* . A random AF generator that generates that (likely) possess many stable, preferred, and complete extensions [58].

Since two isomorphic argumentation frameworks contain identical extensions, each AF is checked for isomorphism with *Nauty* [42] and duplicates are removed. Ground-truth labels are determined based on extensions obtained with the sound and complete μ*-toksia* solver [44]. A test and validation dataset are generated, consisting of size 1000 with AFs containing $|A| = 25$ arguments. A training dataset is generated consisting of a million AFs where the number of arguments per AF is sampled randomly between $5 \leq |A| \leq 25$. AFs of different sizes are included in the training dataset to accelerate the learning. Table 2 shows characteristics of the AFs in the test dataset under different semantics.

Characteristic	grd	prf	stb	com
Extensions per AF	1.0	2.1	1.6	6.3
Arguments per extension	4.7	9.5	11.8	8.0
Scept. accepted arguments per AF	4.8	5.9	5.8	4.8
Cred. accepted arguments per AF	4.8	8.0	7.9	8.0

Table 2: AF characteristics averaged over all AFs the test dataset.

4.2 Training

The AGNN model is instantiated with one hidden layer and a rectified linear unit for non-linearity for the MLPs M^s, M^t and R, a Long Short-Term Memory [29] for U and a shared random embedding x_i for all nodes. The dimensions of the embedding and all hidden neural layers are $d = 128$. The model is run for $\mathcal{T} = 32$ message passing steps. The approach described by [54] is used to determine the learning rate and ℓ_2 regularisation. The AGNN model is trained in batches containing 50 graphs (approximately 750 nodes) using the AdamW optimiser [39] with a cyclical learning rate [55] between 2e^{-4} and 1e^{-7}, ℓ_2 and regularisation of 1e^{-9}. The model is trained by minimising the binary cross entropy loss between the predicted likelihood and the ground-truth binary label. Given he set of target values $Y = \{y_i\}$ and all corresponding nodes in the graph $V^t = \{v_i\}$, the loss at message passing step t is

$$loss^t = \frac{1}{|V^t|} \sum_{v_i^t \in V^t} y_i \cdot log(o^t_{v_i}) + (1 - y_i) \cdot log(1 - (o^t_{v_i})) \tag{10}$$

The loss is minimised at every step t (rather than only on the final step) for two reasons. First by accounting for the loss at every step, the model is encouraged to learn an efficient and convergent message passing strategy. Each message passing step should bring the network closer to the solution. In addition, accounting for every step prevents the risk of vanishing gradients [45]. Due to the recurrent nature of an AGNN, training an AGNN may cause its gradients to vanish or explode [47]. Since node attributes are updated recurrently over multiple steps, information passes through the same neural layers multiple times. Updating these recurrent neural networks through backpropagation is known to cause some of the gradients to become vanishingly small or extremely large. This can result in an unstable learning process, or even stop the gradients from updating completely. By minimising the loss over every step, the gradients are update more directly rather than indirectly through a chain of recurrent connections, preventing gradients to vanish. Exploding gradients are prevented by clipping the gradients by global norm with a 0.5 clipping ratio [47].

4.3 Metrics

A metric is needed to test these reasoning capabilities of an AGNN. Accuracy and F1 are metrics which are often used for classification problems. Although accuracy an easy to interpret metric, it can be also be misleading when used on problems with imbalanced classes. As showed in Table 2, an AF generally contains more rejected arguments than accepted arguments. In other words there is an imbalance between the accepted and rejected class. Consider classifying the sceptically accepted arguments under the grounded semantics. Since on average 4.8 of the 25 arguments are accepted, a classifier can achieve a 80.8% accuracy simply by predicting all arguments to be rejected. This shows it is possible to achieve a high accuracy on imbalanced problems, without actually solving the problem.

A better approach is to use a metric based on a confusion matrix. A confusion matrix gives insights in the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) as shown in Figure 8. Prominent metrics in this context are

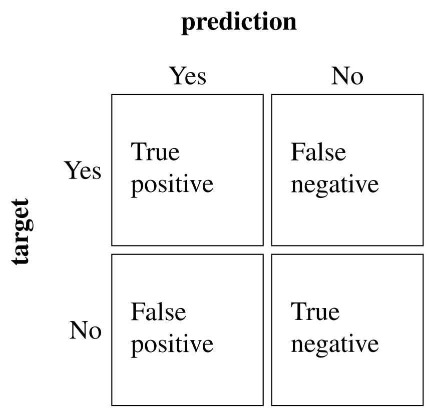


Figure 8: Confusion matrix for decision problems

precision p, recall r and the F1-score

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN} \qquad \text{F1} = 2 \cdot \frac{p \cdot r}{p + r} \tag{11}$$

F1 score is often used in Information Retrieval to measure the performance of classifying a specific active class of interest (i.e. the class that should be retrieved). In this work however, there is no active class since argument acceptance and rejection are equally important. A more encompassing metric for unbalanced binary classification with no active class of interest is the *Matthews Correlation Coefficient* (MCC)

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{12}$$

This metric is regarded as one of the most balanced metrics for evaluating binary classification performance [48].The MCC is a correlation coefficient between the predicted and actual binary classifications. This correlation coefficient is expressed as a value between -1 and 1. A value of -1 represents a perfect disagreement between predicted and target values, while 1 represents a perfect agreement. A value of 0 indicates the classification results are no better than random. The MCC metric accounts for both classes equally, independent from the class balance. Even if one class is disproportionately over-represented, a high MCC value means that both classes are predicted well [48].

For this research, time is not used as an evaluation metric. The goal is of this research is not to design a solver which operates faster than existing symbolic approaches. Rather, the goal is to test how well a graph neural network can learn to solve argumentative reasoning tasks. In addition, it is not straightforward how to compare performance between neural and symbolic approaches. Neural networks operate most efficiently on a graphic processing unit (GPU), while symbolic algorithms operate most efficiently on a central processing unit (CPU). Thus even if the aim of this research would be to compare run times, there is no fair or straightforward method to compare both approaches on common hardware.

		Scept_σ				Cred_σ			
Metric	Model	grd	prf	stb	com	grd	prf	stb	com
MCC	GCN	0.17	0.18	0.20	0.16	0.17	0.17	0.39	0.36
	FM2	0.64	0.54	0.55	0.64	0.63	0.57	0.55	0.57
	AGNN	**1.00**	**0.997**	**0.997**	**1.00**	**1.00**	**0.998**	**0.998**	**0.999**
MAE	AGNN	3e^{-8}	5e^{-4}	9e^{-4}	3e^{-8}	3e^{-8}	6e^{-4}	4e^{-4}	3e^{-4}

Table 3: Argument acceptance results on the test dataset.

4.4 Results

AGNN is trained on the AFs in the training dataset for each acceptance problem and semantics described in Section 2.1 and use the Matthews Correlation Coefficient to evaluate the binary classification performance on the AFs in the test dataset. The AGNN model is compared to a graph convolutional network (GCN) [32] baseline and a reproduction of the FM2 model of Kuhlmann and Thimm [33]. Both are single forward pass classifiers, where FM2 is a GCN with the number of incoming and outgoing attacks per argument added as input features (see Section 7.1 for more information on FM2). Table 3 reports the MCC scores for all models on deciding sceptical and credulous acceptance. AGNN performs considerably better than GCN or FM2, and achieves a perfect score on all problems which belong to complexity class P. On all other problems (belonging to NP or surpassing) AGNN is able to correctly predict the acceptance of arguments almost perfectly. The problem complexity thus seems to impact the performance of the model. The outputs of AGNN is inspected on AFs where some arguments were classified incorrectly in order to find a common pattern in the errors, but none were found.

In addition to the classification performance of the model, the confidence of its predictions are of interest for this research. As a measure of prediction confidence the mean absolute error (MAE) between the predicted likelihoods and the binary ground-truth labels is used. An MAE of 0.01 implies that on average the predicted likelihood deviates 1 percentage point from the ground-truth label. A low MAE thus indicates predictions are made correctly and with high confidence. Table 3 shows that predictions are overall made with high confidence. Most notably AGNN predictions deviate only $3e^{-6}$ percentage points from the true label on all problems for which it achieved a perfect classification score.

4.5 Scaling

Even though AGNN is trained on AFs of size $5 \leq |A| \leq 25$, it is able to determine acceptability in much larger AFs. In order to test how well the trained model scales to larger instances extra test datasets are generated with 1000 AFs containing $|A|$ arguments for each $|A| \in \{50, 100, 200\}$. Figure 9 illustrates the MCC scores for predicting the credulous acceptance under the preferred semantics on different sized AFs as a function of the number of message passing steps $\mathcal{T}$. The figure shows AGNN continues to improve its predictions on large AFs by running for more iterations. Notably the performance on $|A| = 200$ AFs still improves after hundreds of iterations (which is not surprising considering that those AFs on average contain 5e^3 attacks and up to 9e^5

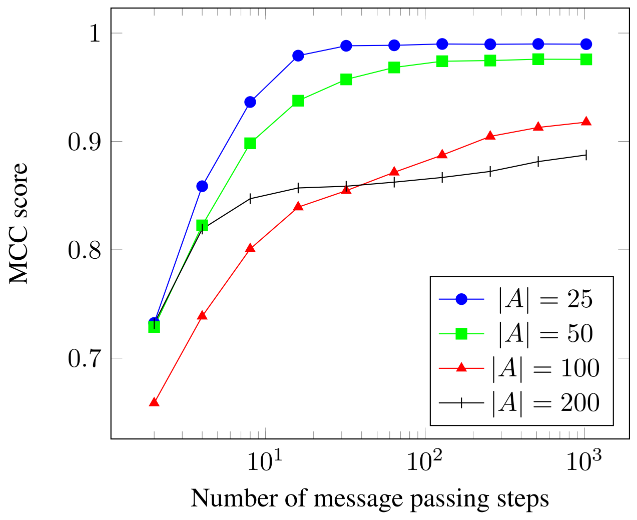


Figure 9: MCC score for $\mathsf{Cred}_{\mathsf{prf}}$ on AFs of different size as a function of the number of message passing steps $\mathcal{T}$. The performance for the first few message passing steps mainly reflects how well AGNN is able to anticipate the status of arguments before convergence. Because AFs are randomly generated the performance during this anticipation phase may vary between datasets, hence the crossing lines for $|A| = 100$ and $|A| = 200$.

extensions). This indicates that, while only being trained to perform 32 message passing steps, AGNN has learned some general and convergent message passing procedure which scales to larger AFs by performing more iterations.

AGNN exhibits similar behaviour on all problems of the same complexity as $\mathsf{Cred}_{\mathsf{prf}}$. On all problems belonging to complexity class P, AGNN is able to correctly classify all arguments in AFs with $|A| = 200$ when run for 32 message passing iterations. All together the results are supportive to the hypothesis that AGNN is able to learn some general procedure which enable it to reason about argument acceptance in an argumentation framework based on the interactions between arguments.

5 Analysing AGNN Behaviour

Chapter 4 showed the AGNN model learns a message passing algorithm which enables it to predict the acceptance status of arguments in an AF. The process of iteratively updating arguments by exchanging messages can be understood as performing a sequence of relational inferences between connected arguments. Since the computations underlying those inferences are learned by neural networks, it is hard to interpret 'how' those inferences enable the model to predict acceptance. In this sections the outputs of each iteration are inspected in order to infer how the model works towards a solution.

AGNN exhibits similar behaviour on all AFs in the test dataset. Arguments are initialised with a low confidence prediction. At each iteration the likelihoods change based on the incoming messages, until arguments 'decide' on their status by *converging* to a high confidence likelihood close to 0 or 1. Generally, the convergence of an argument directly affects the prediction of adjacent arguments in the next iteration. As information is exchanged between arguments, convergence propagates through the graph until the model stops evolving and the likelihoods stay more or less constant.

Focusing on on acceptance under the grounded semantics allows to gain a better understanding of this behaviour. As shown in Table 3 AGNN is able to correctly predict the acceptance of all arguments with extremely high confidence. Inspecting how arguments in an AF converge over consecutive iterations (as shown in Figure 10) reveals three consistent patterns:

1. unattacked arguments converge to *accept*;
2. any argument attacked by an argument which is converged to *accept*, converges to *reject*;
3. any argument which is only attacked by arguments which are converged to *reject*, converges to *accept*.

Any argument which is not affected by these procedures converges to *reject* over the course of multiple iterations. Interestingly, each procedural pattern seems to encode some principle of the grounded semantics. Pattern 1 corresponds to the notion that the defence of arguments included in the grounded extension is 'rooted' in unattacked arguments [4]; pattern 2 corresponds with the principle of conflict-freeness; and pattern 3 corresponds with the principle of defence. Since AGNN exhibits these patterns with extremely high confidence predictions (MAE of $7e^{-7}$) on every AF, it seems the model has learned to encode these principles as procedural rules into its message passing algorithm. The procedural rules also correspond with those used in a well-established symbolic labelling algorithm which can be used to find the grounded extension [43] (as described in Section 2.1.2.2). This algorithm applies the same principles as described in the three observed patterns. It seems AGNN has learned to encode the principles of the grounded semantics and applies these as procedural rules to determine which arguments are contained in the grounded extension.

IT is also observed that AGNN tries to anticipate the acceptance status of arguments in an opportunistic fashion. Arguments which have not yet converged try to anticipate their status based on information about their neighbourhood. At the first message passing step, only unattacked arguments have enough information to converge. For all other arguments a negative correlation between the degree of incoming attacks and the predicted likelihood of being accepted is observed. Statistically seen, the more incoming

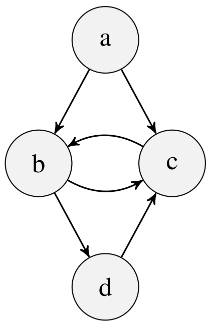
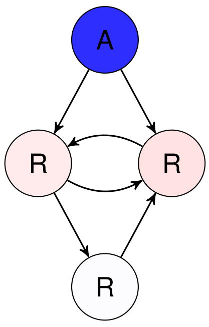
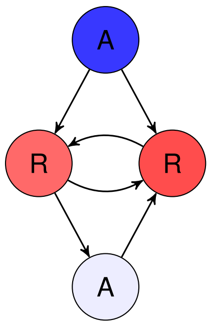
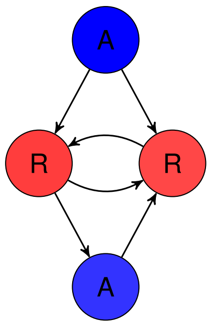
	t		
AF F	1	2	3
			

Figure 10: The acceptance predictions AGNN makes after the first three message passing iterations on the AF $F = (\{a, b, c, d\}, \{(a, b), (a, c), (b, c), (b, d), (c, b), (d, c)\}$ with respect to the grounded semantics. The label and colour of each arguments denote whether the argument is predicted to be A accepted (blue) or R rejected (red) where a darker colour indicates a higher confidence prediction. At $t = 1$ argument a converges to *accept* while the other arguments anticipate *reject* with a confidence that positively correlates with the amount of incoming attacks. At $t = 2$ arguments b and c converge to *reject* while d adjusts its anticipation to *accept* since all its neighbours anticipated *reject*. At $t = 3$ d converges to *accept* after which the model stops evolving.

attacks an argument has, the higher the chance that it is not defended against one of those attacks[1]. It seems that AGNN has learned to infer the amount of incoming attacks from the incoming message and uses this information to anticipate the predicted likelihoods. Additionally, anticipating arguments affect unconverged neighbouring arguments according to the same procedure as mentioned above. An argument attacked by arguments which anticipate *reject* will for instance anticipate *accept* or lower its confidence in anticipating *reject*.

[1]This encodes the ideas behind ranking-based semantics, in which the numbers of attackers and defenders are used to rank arguments [8].

6 Enumerating Extensions

Motivated by AGNN's ability to predict argument acceptance almost perfectly, in this chapter the scope of this research is extended to the extension enumeration problem. The observations from Chapter 5 showed that AGNN is able to predict the acceptance of arguments under grounded semantics by learning a procedure to enumerate the grounded extension. Since an AF always has one grounded extension, there is a one-to-one mapping between enumerating the extension and deciding acceptance. It seems plausible that AGNN is able to learn a similar procedure under the preferred, stable and complete semantics. However, under these semantics an argument can be contained in multiple extensions and as AGNN can only output a single value per argument there is no straightforward way to directly use AGNN to enumerate all extensions.

To facilitate enumeration under multi-extension semantics enumeration is posed as a search problem and use AGNN to guide a basic search. Starting from an empty set of arguments S a search tree is constructed by incrementally adding arguments to S that *extend* S into becoming an extension. When no argument can extend S any further the search procedure backtracks, selects a new argument to extend S and continues the search. Finding extensions with this procedure requires iteratively solving which arguments can extend S into becoming an extension and verifying when S becomes an extension. To address these problems with AGNN the *constructive acceptance* task Constr_σ is proposed.

Definition 6. *Given an AF $F = (A, R)$, a semantics σ, a set of arguments $S \subseteq A$ and an argument $a \in A$, a is said to be* constructively accepted *w.r.t. S if $S \cup \{a\} \subseteq \bigcup_{\mathcal{E}\in\sigma(F)} \mathcal{E}$*

An argument can only be constructively accepted w.r.t. a set which is subset equal to an extension. Given such a set S, an argument a is constructively accepted if it is either contained in S or if adding a to S yields a larger set which is also subset equal to an extension.

Example 5. *Given the set of arguments $S = \{a\}$ in F_e, arguments a, c and d are constructively accepted w.r.t. S under preferred, complete and stable semantics while only argument a is constructively accepted under grounded semantics.*

Consider AF $F = (A, R)$ and semantics σ. Starting from the empty set S is extended S into an extension by recursively computing which arguments are constructively accepted w.r.t. S and adding one of these arguments to S. AGNN is used to approximate the function f_σ mapping F and S to a binary labelling $f_\sigma(F, S)$ indicating for each argument in A if it is constructively accepted w.r.t. S. AGNN is informed which arguments are currently in S by initialising the corresponding nodes with a separate embedding x_i. The computed likelihoods are rounded for each argument to a binary answer.

Each time a constructively accepted argument is added, S is extended into a larger subset of an extension until at some point it becomes equal to an extension. Verifying when S becomes equal to an extension is straightforward under the grounded, preferred and stable semantics. Under these semantics no extension can be a subset of another extension. Therefore S is an extension when all constructively accepted arguments are included in S and no argument can extend it any further. Those extensions are thus found in the leaf nodes of the search tree (as shown in Figure 11). Under the complete semantics this principle does not hold for all extensions. Since a complete extension

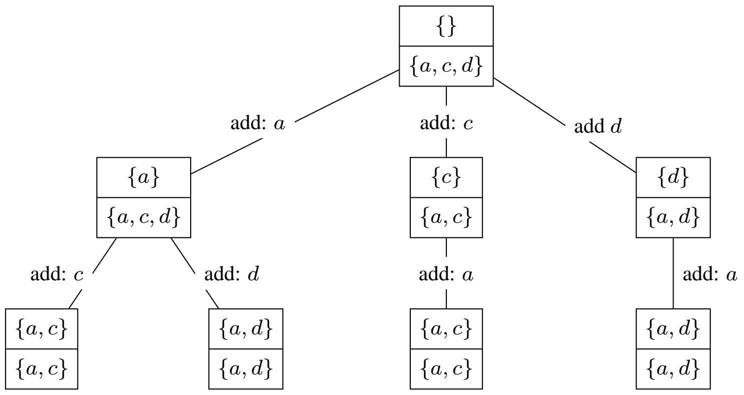


Figure 11: The search tree for enumerating the preferred extensions of the AF F_e. Each tree node illustrates a set S in the top half and the set of arguments which are constructively accepted w.r.t. S in the bottom half. At each step down the tree a constructively accepted argument is included in S until S becomes an extension (i.e. no constructively accepted argument can extend S any further).

can also be a subset of another complete extension, exhaustively extending S until it becomes an extension will find some, but not all extensions.

Since AGNN provides an approximation, somewhere in the tree search an argument a might falsely be labelled as constructively accepted w.r.t. S, yielding the *illegal* set $S \cup \{a\}$. An illegal set is not subset equal to any extension and therefore cannot be extended into a extension. Due to the branching nature of a tree search, a single mislabelled argument early in the search procedure may spawn many illegal sets thereby increasing the risk of an incorrectly enumerated extension. To mitigate this risk while constructing the search tree, the search procedure stops extending any set S when AGNN's output indicates that S contains a constructively rejected argument (since as long as S is subset equal to an extension all argument in S should be constructively accepted by Definition 6). Appendix A shows the complete search procedure in pseudo-code.

6.1 Experimental Setup and Results

The training dataset described in Section 4.1 is altered to supervise AGNN in learning to predict which arguments are constructively accepted w.r.t. a set of arguments. For each AF a set of arguments S is generated, such that S is subset equal to a randomly selected extension to serve as input feature. The ground-truth labels are determined by taking the union of all extensions which contain S and label each argument as *accepted*. To train AGNN in recognising illegal sets, in addition a set of arguments is generated which is not subset equal to any extension. The ground-truth label of each argument in this set are set to *reject*. Table 4 shows exemplary training examples for an AF. During training AGNN thus gets an AF and a set of arguments S (which are either subset equal to an extension or an illegal set) and is asked to predict which arguments are constructively accepted with respect to S. AGNN is trained with the same procedure and parameters as described in Section 4.2.

For each AF in the test dataset AGNN is used to construct a search tree and return the sets found in the leaf nodes. Table 5 shows AGNN is able to enumerate extensions

	S	$\mathsf{Constr}_{\mathsf{prf}}(F, S)$
1	$\{\}$	$\{a, c, d\}$
2	$\{a\}$	$\{a, c, d\}$
3	$\{c\}$	$\{a, c\}$
4	$\{a, c\}$	$\{a, c\}$
5	$\{a, d\}$	$\{a, d\}$
6	$\{b\}$	$\{\}$

Table 4: Exemplary training examples generated for deciding constructive acceptance in F_e under the preferred semantics, where S denotes a set of arguments and $\mathsf{Constr}_{\mathsf{prf}}(F, S)$ the set of arguments that are constructively accepted w.r.t. S. Note that in the fourth and fifth example the sets S are extensions, since all constructively accepted arguments are included in S. Also note that in the final entry S is an illegal set it contains an argument which is not constructively accepted.

	Enum_σ			
Metric	grd	prf	stb	com
Precision	1.00	0.999	1.00	1.00
Recall	1.00	0.998	0.999	0.41

Table 5: Extension enumeration results on the test dataset.

almost perfectly under most semantics. As anticipated the recall under the complete semantics is relatively low since the search procedure cannot find extensions which are a subset of another extension. However when a verification algorithm [7] is included to enable the identification of such extensions, the recall increases to 0.91. The verification algorithm is employed to verify at which steps in the search tree the set S becomes an complete extension. The resulting increased recall indicates AGNN has indeed learned the principles of constructing complete extensions but many are not identified as such due to the nature of the search procedure.

7 Discussion

This chapter discusses related research and compare it to the approach of this research. Additionally, this chapter reflects on the difference between symbolic and learning-based approaches to artificial intelligence and how this thesis fits in the broader attempt to bridge the two paradigms. Finally the thesis is concluded.

7.1 Related Work

Existing research on (deep) learning based approaches to argumentation focus mainly on argument mining – that is, extracting arguments or attacks from natural language text [13] – rather than solving acceptability problems. The exception is recent work by Kuhlmann and Thimm [33], who carried out a feasibility study on the use of a graph convolutional neural network to approximate the credulous acceptance of arguments under the preferred semantics. Kuhlmann and Thimm [33] report achieving 80% overall accuracy on predicting credulous argument acceptance under the preferred semantics. Closer inspection of their results however, show this metric to not accurately reflect the performance of their model. The authors report around 96% accuracy on predicting rejected arguments and only 25% on accepted arguments. Since on average an AF contains more rejected than accepted arguments, it is possible to achieve a high accuracy simply by predicting most arguments to belong to the majority class (i.e. rejected) as discussed in Section 4.3. However, the low accuracy on predicting accepted arguments shows how the reported 80% accuracy can be misleading.

For this thesis, the FM2 model is reproduced in order to compare FM2 to AGNN on a more representative metric. The experiments described in Section 4.4 showed that AGNN greatly outperforms FM2 on all tasks. This result may be explained by the fact that AGNN learns a general message passing procedure while FM2 only learns to perform a statistical inference. First, the proposed FM2 model operates as a conventional single forward pass classifier. In other words the authors map the input to the output in a single computational step. In contrast, AGNN learns to perform a sequence of relational inferences, enabling it to reason about multi-step interactions and long range dependencies between arguments. Additionally, Kuhlmann and Thimm [33] incorporate handcrafted input features to improve the classification performance. In FM2 the number of incoming and outgoing attacks are added to each argument as input features. Adding these features encourages the model find a correlation between the amount of incoming and outgoing attacks of an argument's neighbourhood and the likelihood of it being accepted. Although this might enhance the predictive power of the model (since a node with a high number of incoming attacks will most likely have a higher probability of being rejected) it does not represent the interactions between arguments in an AF. In constrast, AGNN predicts the acceptance of an argument solely based on the attack structure of an AF.

Besides Kuhlmann and Thimm [33], Malmqvist [40] reports on using a deep reinforcement learning algorithm for his submission in the Third International Competition on Computational Models of Argumentation [20]. Malmqvist [40] states his model is able to learn to solve argument enumeration problem under several semantics. However, since the actual model is not described it could not be reproduced, preventing comparison to the the current study.

From a machine learning perspective the AGNN model is close to Palm, Paquet, and Winther [45] and Gilmer et al. [24]. Both describe GNNs that learn neural message passing algorithms on problems from symbolic domains. The model from the current study differs since two message functions are employed in order to distinguish between messages sent from attack source to target or vice versa. A thorough search of relevant literature has not yielded any other GNN implementation using this method. In addition this research shows how GNNs can be used to guide a basic search on problems for which multiple solutions exist. Li, Chen, and Koltun [38] combines GNN with tree search in order to solve the travelling salesman problem. The authors use the tree search to explore multiple tours, before selecting the optimal tour. In contrast, in this study the tree search is used for an enumeration problem where multiple solutions exist. To the best of this authors knowledge this study is the first to combine a tree search with a GNN for an enumeration problem.

7.2 Neural-symbolic computing

This thesis fits in a broader attempt to bridge symbolic-based and learning-based methods in a paradigm called *neural-symbolic computing* [23, 21, 22]. The aim of neural-symbolic computing is to integrate the ability to learn from experience and the ability to reason from what has been learned [59]. Symbolic-based and learning-based approaches have shaped the field of artificial intelligence, but have been largely developed by distinct research communities [22]. Symbolic approaches to reasoning have been dominant during most of the history of AI. Symbolic approaches attempts to capture reasoning in a systematic fashion by representing information such as facts, objects, rules and relations as abstract symbols. By representing knowledge and reasoning in a structured symbolic fashion, symbols can be manipulated to express or infer knowledge. Such manipulations are typically expressed through formalisms such as logic or mathematics, which provide various operators which describe how symbols can be manipulated. Through proof methods it is possible to provide theoretical guarantees about symbolic operations (such as soundness and completeness). Moreover, due to the language like, propositional character these statements are interpretable and explainable to humans. In addition, the abstract symbolic representation allows for generalisation and re-use, while also allowing elementary symbols to be combined through connectives to form higher order abstractions.

The drawback to symbolic approaches is that they are dependent on explicit on handcrafted knowledge. In the history of artificial intelligence it has proven to be very difficult to represent knowledge explicitly through symbols[2]. In addition, critics have questioned where symbolic representations should come from in the first place. Harnad [26] describes this as the symbol grounding problem. Symbolic representations should be grounded in something to have a meaning. After all they are only representations of something. Symbols cannot simply be grounded in other symbols, since this would lead to a circular meaningless regress. According to Harnad [26], grounding symbols in other symbols would be similar to learning Chinese from a Chinese/Chinese dictionary (analogical to Searle's Chinese Room argument [52]). Therefore symbols have to be grounded in something extrinsic, thus a mere symbolic approach to reasoning is insufficient from a philosophical perspective.

A natural candidate for the grounding representations in experience is connectionism [26]. The strength of connectionism, specifically neural networks, is to learn representations from data with little or no prior knowledge. Neural networks are able to

[2]Notably, the symbolic *expert systems* (in 1980 seen as a promising AI approach) did not live up to their expectations due to this problem [27].

discover features in high-dimensional real-valued noisy data. As a result, deep learning methods have revolutionised areas such as natural language processing, computer vision and game playing [36]. Despite these breakthroughs, deep learning has attracted a lot of criticism with respect to its reasoning capabilities. The main criticism on deep learning is that it cannot learn through explication, but needs thousands of examples to learn something [41]. Additionally, neural networks have a limited capacity to transfer concepts. Neural networks are able to perform well on a single problem, but can hardly transfer learned concept to other problems. The learned representations thus lack the general and reusable character of symbolic representations. Finally, neural networks are not transparent nor exact. Neural networks are black-box models which provide approximation. As a result, both the model and the output are generally hard to interpret for humans.

The symbolic and connectionist approaches to artificial intelligence thus seem to have complementary strengths and weaknesses. A hybrid system could benefit from the interpretability and structure of symbolism combined with the learning capabilities and flexible representations of connectionism. The learning-based approach to determining argument acceptance described in this research fits in the broader attempt to integrate symbolic and neural models into a single framework. This thesis shows that sub-symbolic deep learning techniques can accurately solve a problem from computational argumentation that could previously only be solved by sophisticated symbolic solvers. Specifically, this research demonstrates that a GNN is able to represent and reason about the relations between arguments in an AF in a sub-symbolic fashion.

There are still some limit to the performance of the proposed AGNN model. AGNN does not provide the same theoretical guarantees as a symbolic algorithm and can thus make erroneous predictions. The appeal is that it is able to learn a message passing algorithm, which exhibits behaviour similar to a symbolic algorithm, without the expert knowledge of a human designer. Currently, the approach in this study is still dependent on the availability of labelled training data however. The labelled data can only be obtained with a sound and complete symbolic solver. In other words, a solver has to exist in order to train AGNN to be a solver. This dependency could be removed by using AGNN in a reinforcement learning (RL) setting [57]. In an RL environment, the model is not trained with labelled data but explores the solution space until a correct solution is found. Learning to solve reasoning problems in an RL environment thus only requires a method to verify when a correct solution is found (which is trivial in most cases as described in Section 2.1.2.1), thereby removing the dependency on preexisting solvers. This would enable the use of an AGNN on yet unsolved reasoning problems. This direction is left for future research.

7.3 Conclusion

This thesis presents a learning-based approach to determining acceptance of arguments under abstract argumentation semantics, proposing AGNN, which learns a message passing algorithm to predict the likelihood of an argument being accepted. AGNN can almost perfectly predict the acceptability under different semantics and scales well for larger argumentation frameworks. Furthermore, AGNN can also enumerate all extensions under different semantics very well - for multi-extension semantics, AGNN is used to extend a set of arguments such that it becomes an extension. Analysis of AGNN's behaviour shows that it learns to adhere to basic principles of (ranked) argument semantics as identified in the literature [4, 8], and behaves similarly to a well-known symbolic labelling algorithm for grounded semantics [43].

Although AGNN does not provide the same theoretical guarantees as a symbolic algorithm, the appeal of a learning-based approach is that it generalises to different problems without needing the expert knowledge of human algorithm designers [38]. AGNN is a single architecture that can solve different argumentation problems (Cred, Scept, Constr) for different semantics (grd, prf, stb, com) and on AFs larger than seen during training in constant time, simply by running for more iterations. Additionally, by solving the constructive acceptance problem, AGNN can guide a basic tree search which enumerates extensions.

In short this thesis showed it is possible to learn to reason about the interactions between arguments in an argumentation framework with a graph neural network. This research opens up new research directions on neural-symbolic computing with respect to computational argumentation. For future work, the author aims to look at employing AGNN for dynamic argumentation [15], looking at whether AGNN can learn, for example, which arguments or attacks should be added or removed to enforce a certain argument's acceptability.

7.4 Acknowledgements

I would like to thank my supervisor, Florix Bex, for his guidance through each stage of the process; Annemarie Borg for her feedback; SURFsara for the support in using the Lisa Compute Cluster; Utrecht University for their support in using the Gemini Compute Cluster; Weights and Biases for using their experiment tracking tool. Finally I would like to pay my special regards to my girlfriend and mother, who have been supportive and patient throughout all of my academic endeavours.

References

[1] Réka Albert and Albert-László Barabási. “Statistical mechanics of complex networks”. In: *Reviews of Modern Physics* 74.1 (2002), pp. 47–97.

[2] Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Ricardo Simari, Matthias Thimm, and Serena Villata. “Towards Artificial Argumentation”. In: *AI Magazine* 38.3 (2017), pp. 25–36.

[3] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge: Cambridge University Press, 2016.

[4] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. “An introduction to argumentation semantics”. In: *Knowledge Engineering Review* 26.4 (2011), pp. 365–410.

[5] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. *Relational inductive biases, deep learning, and graph networks*. 2018. arXiv: 1806.01261 `[cs.LG]`.

[6] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. *Machine Learning for Combinatorial Optimization: a Methodological Tour d’Horizon*. 2018. arXiv: 1811.06128.

[7] Philippe Besnard and Sylvie Doutre. “Checking the acceptability of a set of arguments”. In: *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR)*. 2004, pp. 59–64.

[8] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. “A comparative study of ranking-based semantics for abstract argumentation”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2016, pp. 914–920.

[9] Martin Caminada. “On the Issue of Reinstatement in Argumentation”. In: *Proceedings of the 10th European Conference Logics in Artificial Intelligence (JELIA)*. 2006, pp. 111–123.

[10] Federico Cerutti, Sarah Alice Gaggl, Matthias Thimm, and Johannes Peter Wallner. “Foundations of Implementations for Formal Argumentation”. In: *FLAP* 4.8 (2017).

[11] Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati. “Generating Structured Argumentation Frameworks: AFBenchGen2”. In: *Proceedings of Computational Models of Argumentation (COMMA)*. 2016, pp. 467–468.

[12] Günther Charwat, Wolfgang Dvorák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran. “Methods for solving reasoning problems in abstract argumentation - A survey”. In: *Artificial Intelligence* 220 (2015), pp. 28–63.

[13] Oana Cocarascu and Francesca Toni. “Identifying attack and support argumentative relations using deep learning”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017, pp. 1374–1379.

[14] Dennis Craandijk and Floris Bex. “Deep Learning for Abstract Argumentation Semantics”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020 [scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic]*. ijcai.org, 2020, pp. 1667–1673.

[15] Sylvie Doutre and Jean-Guy Mailly. “Constraints and changes: A survey of abstract argumentation dynamics”. In: *Argument & Computation* 9.3 (2018), pp. 223–248.

[16] Phan Minh Dung. “On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games”. In: *Artificial Intelligence* 77.2 (1995), pp. 321–358.

[17] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. “Assumption-Based Argumentation”. In: *Argumentation in Artificial Intelligence*. 2009, pp. 199–218.

[18] Wolfgang Dvorák and Paul E. Dunne. “Computational Problems in Formal Argumentation and their Complexity”. In: *FLAP* 4.8 (2017).

[19] P. Erdős and A. Rényi. “On the evolution of random graphs”. In: *The Structure and Dynamics of Networks*. Princeton University Press, Oct. 2011, pp. 38–82.

[20] Sarah Alice Gaggl, Thomas Linsbichler, Marco Maratea, and Stefan Woltran. “Design and results of the Second International Competition on Computational Models of Argumentation”. In: *Artificial Intelligence* 279 (2020).

[21] Artur S. d’Avila Garcez, Tarek R. Besold, Luc De Raedt, Peter Földiák, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luıs C. Lamb, Risto Miikkulainen, and Daniel L. Silver. “Neural-Symbolic Learning and Reasoning: Contributions and Challenges”. In: *Proceedings of the 2015 AAAI Spring Symposia*. 2015.

[22] Artur d’Avila Garcez, Marco Gori, Luis C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. *Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning*. 2019. arXiv: 1905.06088 [cs.AI].

[23] Marta Garnelo and Murray Shanahan. “Reconciling deep learning with symbolic artificial intelligence: representing objects and relations”. In: *Current Opinion in Behavioral Sciences* 29 (2019), pp. 17–23.

[24] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 1263–1272.

[25] William L. Hamilton, Rex Ying, and Jure Leskovec. “Representation Learning on Graphs: Methods and Applications”. In: *IEEE Data Engineering Bulletin* 40.3 (2017), pp. 52–74.

[26] Stevan Harnad. “The symbol grounding problem”. In: *Physica D: Nonlinear Phenomena* 42.1-3 (1990), pp. 335–346.

[27] James Hendler. “Avoiding another AI winter”. In: *IEEE Intelligent Systems* 2 (2008), pp. 2–4.

[28] Geoffrey E. Hinton and Richard S. Zemel. “Autoencoders, Minimum Description Length and Helmholtz Free Energy”. In: *Advances in Neural Information Processing Systems 6 (NIPS)*. 1993, pp. 3–10.

[29] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[30] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1988–1997.

[31] Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. "Learning Combinatorial Optimization Algorithms over Graphs". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017, pp. 6348–6358.

[32] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 2017.

[33] Isabelle Kuhlmann and Matthias Thimm. "Using Graph Convolutional Networks for Approximate Reasoning with Abstract Argumentation Frameworks: A Feasibility Study". In: *Proceedings of the 13th international conference on Scalable Uncertainty Management (SUM)*. 2019, pp. 24–37.

[34] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. "Building machines that learn and think like people". In: *Behavioral and Brain Sciences* 40 (2017), p. 253.

[35] Luis C. Lamb, Artur Garcez, Marco Gori, Marcelo Prates, Pedro Avelar, and Moshe Vardi. *Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective*. 2020. arXiv: 2003.00330 `[cs.AI]`.

[36] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[37] Yann LeCun, Léon Bottou, Yoshua Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[38] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. "Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*. 2018, pp. 537–546.

[39] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. 2019.

[40] Lars Malmqvist. "Yonas: An Experimental Neural Argumentation Solver". In: *International Competition on Computational Models of Argumentation (ICCMA)*. 2019.

[41] Gary Marcus. *Deep Learning: A Critical Appraisal*. 2018. arXiv: 1801.00631 `[cs.AI]`.

[42] Brendan D. McKay and Adolfo Piperno. "Practical graph isomorphism, II". In: *Journal of Symbolic Computation* 60 (2014), pp. 94–112.

[43] Sanjay Modgil and Martin Caminada. "Proof Theories and Algorithms for Abstract Argumentation Frameworks". In: *Argumentation in Artificial Intelligence*. Ed. by Iyad Rahwan and Guillermo R Simari. Springer, 2009, pp. 105–129.

[44] Andreas Niskanen and Matti Järvisalo. *μ-toksia: SAT-based Solver for Static and Dynamic Argumentation Frameworks*. 2019. URL: https://bitbucket.org/andreasniskanen/mu-toksia.

[45] Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. "Recurrent Relational Networks". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*. 2018, pp. 3372–3382.

[46] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 2005.

[47] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. 2013, pp. 1310–1318.

[48] David M. Powers. "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation". In: *Journal of Machice Learning Technology* 2 (2011), pp. 2229–3981.

[49] Henry Prakken. "An abstract framework for argumentation with structured arguments". In: *Argument & Computation* 1.2 (2010), pp. 93–124.

[50] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 1986, pp. 318–362.

[51] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Tim Lillicrap. "A simple neural network module for relational reasoning". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017, pp. 4967–4976.

[52] John R. Searle. "Minds, Brains, and Programs". In: *The Philosophy of Artificial Intelligence*. Oxford readings in philosophy. Oxford University Press, 1990, pp. 67–88.

[53] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. "Learning a SAT Solver from Single-Bit Supervision". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. 2019.

[54] Leslie N. Smith. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. arXiv: 1803.09820 [cs.LG].

[55] Leslie N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017.

[56] Billy Spelchan and Yong. Gao. "The AFGen Benchmark Generator". In: *International Competition on Computational Models of Argumentation (ICCMA)*. 2019.

[57] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[58] Matthias Thimm, Serena Villata, Federico Cerutti, Nir Oren, Hannes Strass, and Mauro Vallati. "Summary Report of The First International Competition on Computational Models of Argumentation". In: *AI Magazine* 37.1 (2016), p. 102.

[59] Leslie G. Valiant. “Three problems in computer science”. In: *Journal of the ACM* 50.1 (2003), pp. 96–99.

[60] Duncan J Watts and Steven H Strogatz. “Collective dynamics of ‘small-world’ networks”. In: *Nature* 393.6684 (1998), p. 440.

[61] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. *A Comprehensive Survey on Graph Neural Networks*. 2019. arXiv: 1901.00596 `[cs.LG]`.

[62] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. *Graph Neural Networks: A Review of Methods and Applications*. 2018. arXiv: 1812.08434 `[cs.LG]`.

A Guided tree search algorithm

Algorithm 1 AGNN guided tree search

Input: An AF F, a set of arguments S and a semantics σ
Output: The found extensions E

```
 1: $S \longleftarrow$ initialise with $\emptyset$
 2: procedure ExtendRecursively($F, S$)
 3:     $I \longleftarrow$ set of constructively accepted arguments according to $\mathsf{AGNN}_\sigma(F, S)$
 4:     if $S \not\subseteq I$ then
 5:         return ILLEGAL
 6:     else if $S = I$ then
 7:         Add $S$ to found extensions $E$
 8:     else
 9:         for each argument $a \in I$ and $a \notin S$ do
10:             $S' \longleftarrow S \cup \{a\}$
11:             ExtendRecursively($F, S'$)
12:         end for
13:     end if
14: end procedure
```